



Submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the

Degree of Bachelor of Science
in Computer Science

and for the
Degree of Bachelor of Science
in Interactive Media and Game Development

and for the
Degree of Bachelor of Arts
in Interactive Media and Game Development

Authors:

Zachary Adams (IMGD BA)
Renee Cullman (IMGD BA)

Conor Dolan (IMGD BA)
Austin Hyatt (CS, IMGD BS)

Jessica Liano (IMGD BA)
Nelson Pires (CS, IMGD BS)

Faculty Advisors:

Farley Chery (IMGD)
Rodney DuPlessis (IMGD)

Gillian Smith (CS, IMGD)
Karen Stewart (IMGD)

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review.

Abstract

Clean Sweep is A 3D, comedic, character-driven, action-RPG taking place in the town of Cleanland, where characters are personifications of cleaning products. Play as a Janitor who has recently inherited a mansion that is infested with germs, Befriend the residents of Cleanland and team up with them to fight germs in character-oriented combat. Most importantly, connect with the characters and support them as they strive to become better versions of themselves.

Clean Sweep was created by a core team of six students with additional help from thirteen external workers consisting of WPI students and alumni, as well as four advisors. It was developed in Unreal Engine 5.3 and can be played on a personal computer (PC) with a controller or keyboard and mouse. The game contains themes of mental health, healthy relationships, and diversity. Our priority was ensuring significant and proper representation of marginalized groups that typically endure poor or negative portrayals in media and games.

Clean Sweep is a testament to the creativity, dedication, and collaboration of its development team and advisors. Through its unique concept, gameplay, and commitment to addressing meaningful themes of mental health, relationships, and representation, the game entertains and resonates with players. The challenges faced in its development were met with perseverance and innovation resulting in a project that exceeded management and content expectations. Throughout this year, we have stayed on track to release our game for the public to purchase and enjoy. Additionally, the valuable insights, tools, and workflows we developed during this project will be shared to enrich the Interactive Media and Game Development program at WPI.

Team Breakdown

Production

Nelson Pires – Co-Producer

Renee Cullman – Co-Producer

Jessica Liano – Creative Director

Art

3D Characters

Jessica Liano – Character Lead

Conor Dolan – 3D Character Modeler

Renee Cullman – 3D Character Modeler

3D Environments

Conor Dolan – Environmental Lead

Daniel Baranov – Environmental Artist (C24)

Ethan Lockhart – Environmental Artist (C24, D24)

2D Art

Jessica Liano – Lead 2D Artist

Technical Art

Zachary Adams – Lead Technical Artist

Animation and Mocap

Zachary Adams – Lead 3D Animator, Mocap Artist, Rigger

Mia Bourguignon – 3D Animator, Mocap Artist (C24), VFX (D24)

Victoria Rindeiko – Motion Capture Consultant

Hanwen Xu – Tool Consultant

Character Optimization

Zachary Adams – Optimization Lead

Caleb Prouty – Character Optimizer, Character Clothing Optimizer (C24)

Camille Prats – Character Optimizer, Character Clothing Optimizer (D24)

Keenan Jones – Character Clothing Optimizer (D24)

Sarah Bodkin – Character Clothing Optimizer (D24)

Tech

Nelson Pires – Lead Programmer

Nicholas Frangie – Programmer (B23, C24)

Justin Ignatowski – Programmer (B23, C24)

Austin Hyatt – Combat Lead

Tate Donnelly – Programmer (D24)

Writing

Renee Cullman – Lead Writer

Ethan Chau – Writer (C24, D24)

Jessica Liano – Writer

Nelson Pires – Writer

Design

Austin Hyatt – Lead Designer

Nelson Pires – Designer

Jessica Liano – Designer

Renee Cullman – Level Design

Audio

Jessica Liano – Audio Lead

Braden Arnold – Implementation (C24, D24)

Nelson Pires – Implementation

Music and Soundtrack

Jessica Liano – Composer, Producer, Vocalist

Braden Arnold – Consultant (C24, D24)

Sound Effects

Jessica Liano – Sound Designer

Braden Arnold – Primary Sound Designer (C24, D24)

Andrew Simonini – Sound Designer (C24, D24)

Voice Acting

Jessica Liano – Performance Director

Nelson Pires – Quality Controller

Character Actors

Jessica Liano – Roxi

Nelson Pires – Shawn

Trevor Parks – Beau, Vic, Van, Mr. Tidy

Hannah Belan – Charming

Sofia Encarnação -- Windy

Adam Ferriotti – Cain

Alexander Hayden - Soap

Jessica Rhodes – Lysa

Acknowledgements

The *Clean Sweep* team would like to thank our advisors, Rodney DuPlessis, Farley Chery, Gillian Smith, and Karen Stewart, for their guidance throughout production. Additionally, we would like to thank everyone who contributed to *Clean Sweep*: Braden Arnold, Daniel Baranov, Sarah Bodkin, Mia Bourguignon, Ethan Chau, Tate Donnelly, Nicholas Frangie, Justin Ignatowski, Keenan Jones, Ethan Lockhart, Camille Prats, Caleb Prouty, and Andrew Simonini. We could not have achieved what we have without you. Thank you to Hannah Belan, Sofia Encarnaçao, Adam Ferriotti, Alexander Hayden, Trevor Parks, and Jessica Rhodes for lending their voices to our characters. Thank you to Arnold Lane and the Office of Diversity and Inclusion along with Mathew Berry and the Student Development and Counseling Center for their support and guidance. Thank you to the play testers and focus group participants who provided invaluable feedback to improve our game. We also thank the plugin creators, business consultants, and lawyers for sharing their knowledge. Finally, a special thanks to Zach's cat (especially for ~~deleting~~ helping us edit the paper), Austin's dog, Renee's Grandmother, and the best janitor there is (who just so happens to be our biggest fan), Gregory P. Aubin.

Table of Contents

Abstract.....	ii
Team Breakdown	iii
Acknowledgements.....	vii
Table of Contents.....	viii
List of Figures/Tables	xvii
1 Introduction	1
1.1 Design Pillars	1
1.2 Accomplishments.....	1
2 Project Management	5
2.1 Communication.....	5
2.2 Meetings	5
2.3 Project Management Tools and Methods	5
2.4 ISPs	6
2.5 Events, Deadlines, and Goals	7
2.6 Recommendations	9
3 Writing	12
3.1 Themes.....	12
3.2 Consultation.....	13

3.3 The World of <i>Clean Sweep</i>	14
3.4 Structure	16
3.4.1 Main Story.....	16
3.4.2 Character Routes.....	21
3.4.3 Hangouts.....	22
3.4.4 Quests	23
3.4.5 Pooled Dialogue	24
3.5 Setting and Locations.....	24
3.6 Characters	27
3.6.1 Main Story Characters.....	27
3.6.2 Character Route Characters.....	33
3.6.3 Additional characters.....	51
3.7 <i>Clean Sweep</i> Wiki.....	53
3.8 Recommendations	54
4 Game Design	57
4.1 Design Within the Team.....	57
4.2 Blending Genres.....	57
4.3 Genres and Inspirations	58
4.4 Mapping Out the Game	59
4.5 Integrating Narrative and Combat.....	60

4.5.1 Character Routes.....	61
4.5.2 Hangouts.....	61
4.5.3 Quests.....	62
4.5.4 Mansion Storyline.....	62
4.5.5 Janitor Jobs.....	63
4.6 Phone UI.....	63
4.7 Combat.....	64
4.7.1 Combat Vision.....	64
4.7.2 Controls and Game Feel.....	65
4.7.3 Swapping.....	67
4.7.4 Attack Options and Moveset Design.....	68
4.7.5 Character Movesets.....	70
4.7.6 Items.....	74
4.7.7 Enemies.....	75
4.8 Recommendations.....	76
5 Art.....	77
5.1 Art Style.....	77
5.2 2D Art.....	77
5.2.1 Concept to Model.....	78
5.2.2 UI Design.....	78

5.2.3 Sprites	80
5.3 Characters	83
5.3.1 Humans	83
5.3.2 Germs	85
5.4.2 The Social Implications of Jiggle Physics	86
5.4 Environment.....	86
5.4.1 Overall Style	86
5.4.2 Main World	89
5.4.3 Mansion	92
5.4.4 Market.....	93
5.4.5 Beach.....	95
5.5 3D Character Art Pipeline	96
5.5.0 Pipeline Overview	96
5.5.1 Concept	97
5.5.2 Modeling	97
5.5.3 Texturing	99
5.6 3D Environmental Pipeline	102
5.6.1 Layouts and Concept Art.....	102
5.6.2 Grayboxing	103
5.6.3 Detail Modeling.....	104

5.6.4 UV Mapping	106
5.6.5 Texturing	107
5.6.6 Polish and Juice	108
5.7 Recommendations	110
5.7.1 2D	110
5.7.2 Characters	110
5.7.3 Environments	112
6 Character Technical Art Pipeline	113
6.1 Developing a Pipeline	113
6.1.1 Process	114
6.2 Project Management	114
6.2.1 Version Control	115
6.2.2 Remote Access Software	115
6.2.3 Setting Projects	116
6.2.4 File References and Namespaces	117
6.3 Pipeline Development	119
6.3.1 Exploring Programs and Tools	119
6.3.2 Tool Development	121
6.3.3 Asset Development	122
6.3.4 Miscellaneous Asset Development	131

6.3.5 Tutorials and Training	131
6.4 <i>Clean Sweep</i> Technical Art Pipeline	132
6.4.1 Optimization and Retopology	132
6.4.2 Postprocessing	134
6.4.3 Rigging.....	135
6.4.4 Animation.....	137
6.4.5 Implementation	141
6.4.6 Recommendations	141
7 Technical Implementation	143
7.1 Game Engines.....	143
7.2 Source Control	144
7.3 Plugins.....	146
7.3.1 Blueprint Assist	146
7.3.2 Auto Size Comments	146
7.3.3 Global Event System	147
7.3.4 TMW Event Aggregator.....	149
7.3.5 Dialogue System X and Inkpot	150
7.3.6 LE Extended Standard Library	151
7.3.7 Runtime Audio Importer	152
7.4 Levels.....	152

7.5 Combat.....	155
7.5.1 Hitbox System	155
7.5.2 Basic Character Function	157
7.5.3 Playable Characters.....	158
7.5.4 Inputs	159
7.6 Enemies.....	160
7.6.1 Enemy AI and Token System.....	160
7.6.2 Enemy Types and Mechanics	164
7.7 NPCs	165
7.7.1 Schedules and Spawning.....	165
7.7.2 Shops.....	166
7.8 UI.....	168
7.8.1 Common UI	168
7.9 Dialogue System.....	170
7.9.1 Dialogue Widget and Logic	171
7.9.2 Perfecting the Dialogue Pipeline.....	172
7.10 Quest System	173
7.10.1 Sending, Receiving, and Interpreting Events	174
7.10.2 Quest Creation and Expansion.....	178
7.10.3 Scripting Flags	180

7.11 Recommendations	181
8 Audio	183
8.1 Soundtrack Design	183
8.1.1 Overworld	183
8.1.2 Combat and Character Themes	184
8.1.3 Worldbuilding and Miscellaneous	185
8.2 Sound Effects	186
8.3 Voice Acting	188
8.3 Implementation	190
8.4 Recommendations	193
9 Marketing.....	195
9.1 Social media	195
9.2 Events.....	195
9.3 Handling Copyright	195
9.4 Recommendations	196
10 Results and Analysis	197
11 Conclusion.....	198
11.1 Accomplishments.....	198
11.2 Setbacks and Failures.....	199
11.3 Advice for Future Projects	200

11.4 Future Development	201
11.5 Final Remarks	202
Works Cited	203
Appendix A: Asset Production Excel Sheet	208
Appendix B: Playtesting Feedback Form	209
Appendix C: Art Interview Feedback Questions	210
Appendix D: Focus Group Structure and Questions	211
Appendix E: Character Bible	212
Appendix F: Example Character Route Event One	231
Appendix G: Character Voice Descriptions	240
Appendix H: Rough Concept Sketch to Character Model	242
Appendix I: Character Sprite Expressions	244
Appendix J: Technical Art Pipelines and Tools	246
Appendix K: Common UI Documentation	247
Appendix L: <i>BURST UR BUBBLE</i> Lyrics	256
Appendix M: <i>Clean Sweep</i> Computer Science Project Presentation Poster	259

List of Figures/Tables

Figure 2.1: The state of combat at Protofest (top) versus Alphafest (bottom).....	8
Figure 3.1: Initial mansion layout with 6 levels	19
Figure 3.2: Revised mansion layout with 5 levels	20
Figure 3.3: Cleanland location descriptions and their current status.....	26
Figure 3.4: Table of non-playable characters (NPCs) featured in the main story and character routes. ...	52
Figure 3.5: <i>Clean Sweep</i> Wiki home page	54
Figure 4.1: First iteration of game concept map	60
Figure 4.2: Second iteration of game concept map.....	60
Figure 4.3: early mockup of phone UI with messages, contacts, items, and to-do list	64
Figure 4.4: Inputs for the game using controller	66
Figure 4.5: Prototype for swapping button radial menu	68
Figure 5.1: A dialogue UI first pass with a placeholder janitor sprite.....	79
Figure 5.2: A rough mockup of an interactable item shop in the market.....	80
Figure 5.3: A side by side of Charming’s sprite reference screenshot and her first pass neutral sprite. ...	82
Figure 5.4: Sprites for the Cosmic Broomie (left) and the Moppy Joe (right).....	82
Figure 5.5: Olive’s first design, her finalized design, and her model.	84
Figure 5.6: From left to right, the Melee, Ranged, and Exploding Germs.	86
Figure 5.7: A pot asset before and after applying the slope blur filter.....	87
Figure 5.8: A pile of rocks without the outline shader.	88
Figure 5.9: A pile of rocks with the outline shader.....	89
Figure 5.10: Opacity maps used to make the color variation for the grass (left) and sand (right).....	90

Figure 5.11: A demonstration on how RVTs allow for assets to take the color of the landscape underneath them.....	91
Figure 5.12: The Mansion with the Custom Skybox in the Background	92
Figure 5.13 The Mansion’s Dining Room.....	93
Figure 5.14 The Mansion’s Kitchen.....	93
Figure 5.15: Cleanland Market.....	94
Figure 5.16: Wave’s stall at the Market.....	94
Figure 5.17: Olive’s stall at the Market.....	95
Figure 5.18: The Beach.....	96
Figure 5.19: A blackout of Beau using different subtools to build the different forms of muscle groups. 97	
Figure 5.20: Beau’s full model with smoothed features, clothing, and hair.....	98
Figure 5.21: The differences in subtool subdivision levels; getting a low poly surface and subdividing it to sculpt will help technical art create retopology easier.....	99
Figure 5.22: A texture breakdown for Windy’s face, focusing on the skin’s base color.....	101
Figure 5.23: A texture breakdown of Lysa’s pants, showcasing the procedural application of the tartan pattern.	102
Figure 5.24: Concept Art and Reference Board for the Squeaki Tiki	103
Figure 5.25: Graybox for the Squeaki Tiki.....	104
Figure 5.26: Low-Poly (left) and High-Poly (right) versions of a tiki mask asset.....	106
Figure 5.27: Examples of the Substance Painter Templates.....	108
Figure 5.28: Polished version of the Squeaki Tiki, with baked lighting and simple VFX.....	110
Figure 5.29: Vic’s first, second, and third iterations through development, sans shoes.	112
Figure 6.1: The development phases of a pipeline.....	114
Figure 6.2: location of workspace file in the file system	117

Figure 6.3: File references in Maya.....	118
Figure 6.4: Namespace editor and resulting outliner	119
Figure 6.5: Advanced skeleton rig.....	120
Figure 6.6: Topology transfer using Wrap	121
Figure 6.7: Developing tools with Charcoal Editor.....	121
Figure 6.8: Base meshes with female and male anatomy	123
Figure 6.9: Quad, Triangle, N-gon, Non-manifold.....	124
Figure 6.10: Pole Examples	124
Figure 6.11: Pole Redirection Patterns	125
Figure 6.12: Using 8 edges to connect a round form to a grid	125
Figure 6.13: Inset reduction and diamond reduction	126
Figure 6.14: Localization of the ear geometry preventing extra edges throughout the face.....	126
Figure 6.15: Mesh lacking density in deformation area (left). Additional edges in deformation areas (right)	127
Figure 6.16: Using poles to create anatomy support loops for secondary anatomical forms on Scapula and Elbow.....	128
Figure 6.17: Diamond Reduction Pattern used to increase density between fingers	129
Figure 6.18: 3D sculpted head (left). Retopologized head with face bag (right)	129
Figure 6.19: High-Poly head (left) and retopologized head (right).....	130
Figure 6.20: Facial Topology following facial anatomy	130
Figure 6.21: Sergi Caballer guide on pole placement	131
Configuration files were exported from DCCs to speed up processes. For example, <i>Human IK</i> definition presets and skin weights were exported from <i>Maya</i> and spline data was exported from <i>Wrap</i> . These assets were imported by the technical artists when an update was created.	131

Figure 6.22: Training videos provided to technical art team	132
Figure 6.23: Pipeline workflow steps	132
Figure 6.24: UV and Vertex Order Transfer allowing for skin weights transferred using NgSkinTools. ...	133
Figure 6.25: Wrap node network	134
Figure 6.26: Optimized assets after assigning materials and vertex colors	135
Figure 6.27: Iterative rigging workflow	136
Figure 6.28: Windex Katana weapon rig	137
Figure 6.29: Rigging props with the Node Editor	137
Figure 6.30: Using animation layers to improve silhouette	140
Figure 6.31: Fine tuning action for stylized animation within <i>Maya's</i> time editor	140
Figure 6.32: Model import settings in Unreal Engine	141
Figure 6.33: <i>Clean Sweep</i> female base mesh 02	142
Figure 7.1: The bind function that bound new gameplay tag events to their respective functions	148
Figure 7.2: The functions each gameplay tag event is bound to.	148
Figure 7.3: <i>TMW Event Aggregator's</i> event binding function.	149
Figure 7.4: The <i>TWM Event Aggregator</i> class used to send data between actors.	150
Figure 7.5: An early prototype map of the mansion that utilized level streaming with each room being its own sublevel.	154
Figure 7.6: The level streaming of the prototype mansion with levels being loaded based on what rooms the Janitor had previously entered from.	155
Figure 7.7: Capsule hitbox attached to Charming's palm socket and drawn with a debug capsule	157
Figure 7.8: Group of multiple hitboxes using Coexist IDs to make the shape of a broom	157
Figure 7.9: Custom velocity timeline for janitor standard attack	159
Figure 7.10: An early prototype of our AI Blackboard.	161

Figure 7.11: Melee germ behavior tree in Unreal.	164
Figure 7.12: The widget designer of shops with Wave's shop data loaded into it.	167
Figure 7.13: The shop purchase item pop-up in the widget designer.	167
Figure 7.14: CommonInputActions being used to change the interact key for controllers and keyboard. On the left is the controller being the active input and on the right is keyboard and mouse being the active input.	170
Figure 7.15: The widget designer of our dialogue widget, CAW_Dialogue.	171
Figure 7.16: The Shop Manager binding to the <i>Inkpot</i> subsystem's On Story Begin event to a function that binds the function "OpenShop" to <i>Ink</i> calling OpenShop	172
Figure 7.17: Our Gameplay Tag Manager with every gameplay tag we've used so far.	175
Figure 7.18: An example of creating a tag event, loading data into it, and broadcasting the message. This example is in CAW_Dialogue.	176
Figure 7.19: The Quest Manager processing the Gameplay Tag Event Class' data it received.	177
Figure 7.20: The Quest Data Asset utilizing the quest task type to process the Gameplay Tag Event's data and update the current progress.	178
Figure 7.21: The quest data asset's meta tags for quest UI and its activations requirements.	179
Figure 7.22: The quest data asset's phases utilizing quest task types, meta information, and dialogue associated with that phase.	179
Figure 7.23: The quest data asset's rewards.	179
Figure 7.24: An example of a tooltip in the quest data asset.	180
Figure 7.25: An example of a quest processing a message. This example is from QT_ScriptingFlag	180
Figure 7.26: The quest rock's code for deleting the rock when spawning if the scripting flag is higher than the required value.	181
Figure 8.1: The <i>Ableton</i> workspace for <i>BURST UR BUBBLE</i> , a combat song.	184

Figure 8.2: Adding different effects to folied audio to create unique UI sound effects.....	187
Figure 8.3: The line selection process in Reaper.	189
Figure 8.4: The proposed folder structure with <i>Runtime Audio Importer</i> that became the foundation for the current implementation of dialogue voice lines and sound effects.....	192
Figure 8.5: The lookup table for dialogue voice lines being implemented.....	193
Figure 10.1: Responses for combat pacing from 2 different playtests.....	197

1 Introduction

Clean Sweep is a 3D, comedic, character-driven action-RPG developed in Unreal Engine 5.3 where everyone in the town of Cleanland is a human personification of a real-world cleaning product. As the town janitor, befriend the residents of Cleanland and team up with them to fight germs. The original concept for *Clean Sweep* was a blend between a visual novel dating simulation and a turn-based RPG, referencing the likes of *Super Mario RPG* and *Fire Emblem*. As development progressed in pre-production *Clean Sweep* became an action-RPG and social simulation hybrid. The gameplay was inspired by our main comparable games, *Genshin Impact* and *Persona 5*. Our games content is focused on healthy relationships, mental health, positive representation, and diversity.

1.1 Design Pillars

The goal of this project was to create a game that balanced our genres and themes into a cohesive experience that would appeal to an audience of players in their late teens and early twenties. To guide this development, we established the following pillars of design: Goofy, character-focused, and thrilling. These pillars deeply influenced the world of *Clean Sweep*, including the town of Cleanland, the main story, and the diverse cast of dynamic characters, along with their integration with the gameplay mechanics. Additionally, they served as guides in the decision-making process, ensuring consistency in realizing the game's vision across all departments.

1.2 Accomplishments

We successfully managed a large design team comprised of students. We were able to manage a large team of students. As a 3D game, *Clean Sweep* had numerous departments. Within each department, additional help was brought on to keep production on track. Six core members and an additional thirteen students were brought on throughout development. Project management was vital to organizing all these contributors and the flow of production. Finding methods to increase communication and management tools that supported the development process of each department was vital in maintaining a productive workflow.

An accomplishment of the team was the optimizing production process using creative strategies to reduce development time. This allowed us to focus on the story we wanted to tell and get the most out of our team. These tools and workflows will be shared to better WPI's Interactive Media and Game Development program. One of our project's focuses was creating tools and workflows to minimize the workload of our team members. This included automatically implementing assets without the need to download Unreal Engine or the project, drag and drop dialogue implementation, and easy creation of custom data assets like quests for our quest system. We also developed robust managers that help govern gameplay including NPC, Item, Quest, and Dialogue Managers.

With the help of Professor Chery, we were able to get Wrap a 3D topology software for IMGD. Through ISPs and in class lectures, we shared our research to assist their projects and enrich the IMGD department.

Clean Sweep blends both 2D and 3D art. Every art asset in *Clean Sweep* was designed by hand. No store assets were used. The AAA game industry is primarily focused on 3D content and looking for developers with skills in 3D game development. We aimed to finish this project with a

strong portfolio that would make us stand out in the current job market. For this reason, we developed *Clean Sweep* in 3D. *Clean Sweep* challenged us to deepen our understanding of 3D Game Development. 3D games are leagues more graphically and technically involved than 2D games. Although we have the utmost respect for 2D media, 3D game art is a monumental task of coordination and problem solving a dimension more demanding than 2D artwork. Modeling, rigging, 3D animation and texturing are expansive and time demanding disciplines. 3D graphics are also more costly on computer hardware so it was necessary to optimize the entire development process. This included retopology, texture space optimization, rig optimization and GPU and CPU profilers to locate graphical and computational bottlenecks. The desire to ensure *Clean Sweep* could run on any computer regardless of hardware speed forced us to develop a deeper understanding of optimization and 3D graphics.

We are especially proud that we exceeded our initial project goals by producing a significant amount of original content for *Clean Sweep*. In numbers, the *Clean Sweep* team created:

- An explorable town consisting of eight modeled environments
- 312 3D props
- 32 diverse 3D characters
- 62 2D elements
- Over 776 original sound effects
- Seven original songs
- Voice acting
- 70 original animations

- Fifteen technical art mocap tools
- Nine seven-event character routes with branching dialogue options
- Nine playtesting sessions
- Too many cleaning puns to count!

In the chapters ahead, we will be discussing all facets of our game, including project management, contribution breakdowns by discipline (writing, design, art, technical art, programming, and audio), as well as the highs, lows, and key takeaway learned during the development of *Clean Sweep*.

2 Project Management

2.1 Communication

Communication was key in the development of *Clean Sweep* and our primary method of communication among team members and advisors was Discord. Discord is a messaging platform that allows users to create private or public servers where they can converse through messaging, voice chat, and video chat. We utilized a private server with text channels for general updates, as well as dedicated channels for each department and specific disciplines within those departments.

2.2 Meetings

Meetings were the foundation of *Clean Sweep's* production. We held regular meetings to facilitate communication, share updates, set goals, and accomplish work. The frequency and duration of meetings fluctuated depending on workload and the addition of team members. Initially, we scheduled meetings using *When2meet*, but later transitioned to *Vailable* as an easier platform to compare availabilities. *25live* was used to reserve rooms for entire terms, but late scheduling made it challenging to secure a consistent space. We learned the importance of gathering everyone's availability early to ensure convenient meeting times and better options when booking meeting locations. Meeting minutes were maintained to enhance productivity during meetings. These minutes outlined agendas, recorded decisions, and planned tasks, serving as a valuable resource to reference later in production.

2.3 Project Management Tools and Methods

Different teams required different management tools. The programming team was familiar with *Jira* for task management and found it to be an effective tool. *Jira* aligns with the scrum development cycle. How the programming team aimed to structure their pipeline. Scrum involves planning, testing, and iterating to create big software. This entailed the planning of weekly sprints, or cycles of development, to work towards. Every Monday new tasks would be created or retrieved from a backlog, put into the sprint on *Jira*, and assign to various members to complete that week.

The art team attempted to use *Jira* to maintain consistent management tools, but quickly realized it was not suitable for their workflow and team members. Consequently, they transitioned to Trello, creating discipline-specific boards, each organized to reflect their unique pipeline.

In addition to *Jira* and *Trello*, asset progress was maintained in a multi-tab excel spreadsheet (see Appendix A). Within these sheets, we utilized priority lists which spanned departments. This method allowed easy access and comparison of priorities between departments, keeping tasks aligned and enhancing communication within the team.

2.4 ISPs

With a project of this size, additional help was necessary to achieve our vision. We recruited students to work on our project through independent study projects (ISP)s. To find interested students, we reached out to friends, received recommendations from advisors, sent google forms to the WPI IMGD Discord, and advertised ISP opportunities at Showfest, a student showcase event. Person-to-person recruitment methods were the most successful in establishing communication with potential ISPs. Once a student showed interest, we scheduled a meeting with the entire team. It was important for the core team to agree on inviting someone to the project.

This reinforced unity, as the core members were aware of who was working on the project even if they were outside of their department. The same could be said for the ISPs; they got to meet the team that they were working with.

ISPs were managed within their department. Department leads oversaw work for their ISPs and provided updates to production during team meetings. This method was used to manage the high number of ISPs. We left core team meetings open to ISPs to attend if they wanted to voice questions, comments, or concerns. All ISPs were added to the *Clean Sweep* Discord server, which allowed for easy communication and a way for ISPs to stay informed of important announcements and general information.

2.5 Events, Deadlines, and Goals

Several events throughout the year drove the scheduling of the project: Protofest, Alphafest, PAX East, and Showfest. Protofest, Alphafest, and Showfest are WPI events where IMGD students present their games to the community. Pax East is a large gaming convention that would allow us to showcase our project to a larger audience outside of the WPI community. With an event taking place during each of the four terms, these events gave us milestones to orient production.

For Protofest we wanted to create a demo with combat for people to experience. We wanted to make sure that the gameplay mechanics were fun, and getting feedback on it early on was a goal for production and design. In addition to this we had a slide show with art and character sheets to gauge interest in these elements of the game. We found that people liked the visual design, signaling that we were headed in the right direction.

For alphafest, we expanded upon our protofest demo. For this version, or build, we had a more developed town square, more dialogue, a more fleshed out combat system, actual characters implemented into the game, and enemy AI. We also showcased art including 3D modeled characters and environments. Our build's quality had gone up exponentially since Protofest and people were more excited about the development of *Clean Sweep* than ever.



Figure 2.1: The state of combat at Protofest (top) versus Alphafest (bottom).

Due to a bug that crashed our game, we were not accepted into PAX East. This was disappointing for the team, but we quickly addressed this bug and pivoted to a new milestone: the Clark University IGDA showcase. At Clark we presented the first iteration of what would be our Showfest demo. We experienced several technical difficulties that were fixed at the event or noted to be addressed later. We received lots of positive feedback, made new connections, became aware of bugs to address, and took note of ways to enhance the gameplay experience.

Presenting a refined vertical slice at Showfest was our ultimate milestone during production. We realized the need to focus on developing a concise portion of the game after comparing the scope of the entire game with the amount of time Showfest attendees would spend playing. Our vision for this vertical slice changed quite a lot during the school year, but always maintained the core elements of gameplay including combat, visuals, audio, and narrative.

At each event, we used an anonymous playtesting feedback survey that stayed consistent throughout production, allowing us to receive honest feedback and see how our game evolved over the course of production (see Appendix B). For research involving human participants, the team utilized the International Review Board (IRB) to ensure quality research tools. These tools include playtesting and art survey questions, informed consent documents, along with the questions asked and materials provided to focus group participants (see Appendix B, C, D). After each event was completed, playtesting feedback was reviewed and utilized to direct production towards our next milestone.

2.6 Recommendations

Communication is key. Having lots of meetings is daunting, but often necessary. Make sure that information discussed within these meetings is shared with other departments to ensure the team is on the same page. Keeping everyone informed will keep the project in line and team synergy intact. On the topic of communication, remind people of events and goals. This not only ensures that tasks are not being forgotten, but can initiate conversations about said tasks, such as the feasibility of completion by a certain date, if something went wrong, or if a task should no longer be pursued.

Create multiple milestones to drive your project. Having one large goal is great, but creating smaller goals along the way is necessary to keep production on track. Scheduling playtesting sessions is a highly recommended example of a small milestone. Once people sign up for playtesting, it becomes a hard deadline that the team cannot put off. It is incredibly important to receive feedback early and often to ensure that you are achieving your desired experience goals and so that features can be altered it becomes too challenging to do so.

As you prepare for playtesting make sure you plan. Whether it's playtesting, focus group sessions, ISP recruitment, or submitting something to the IRB for approval, make sure you give adequate time for responses. On the topic of the IRB, establish direct communication. This will be helpful when it comes to updating submissions and can even speed up the approval process.

Reach out for assistance, whether it's to advisers, resources on campus, or people with experience outside of school, always be seeking information to understand your options to make informed decisions. Experiment with management techniques to find what works best for your team and individual departments.

Wanting to be involved in multiple departments is great, but when it comes to working on an ambitious group project, it is important to limit the roles people take on. Varying levels of priority will prevent work from getting accomplished and departments will get behind. The writing department was the main victim of this fault, as two thirds of the team were the leaders of other departments. Do not be afraid of ISPs, they can help pick up slack and contribute new features with their unique skills and interests. That said, make sure to find an effective method of ISP management and keep constant communication. If an ISP is not doing the work, reaching out to advisors for assistance is always an option.

Plan out meeting schedules and book rooms as soon as possible. Planning early ensures that people have the most availability, preventing meetings at inconvenient times. Having consistent meeting spaces is not only helpful for organization, but a familiar, comfortable environment can facilitate work. During these meetings, take note of decisions. This helps with organization and prevents people from forgetting past decisions.

Production plays a vital role in game production and should not be overlooked. It is a big responsibility and can make or break a game. Staying organized and facilitating team communion are only two of the many vital tasks production must perform to maintain a project.

3 Writing

In *Clean Sweep*, the story serves as a foundational element that not only enriches the gameplay experience but also conveys important messages and themes. While *Clean Sweep* could have prioritized combat, we believe that a compelling narrative adds depth and meaning to the player's journey. With that in mind, our story was crafted to resonate with our audience and promote themes of mental health, healthy relationships, and diverse representation. We undertook research and consultation to accurately portray these themes within our narrative. This section illustrates how these themes were integrated into *Clean Sweep*, enriching the player's journey and creating a more meaningful experience.

3.1 Themes

Within our story there were several themes we wanted to promote and believed would resonate with our audience: mental health, healthy relationships, and diverse representation. These themes reflect our commitment to promoting an inclusive and supportive gaming environment. With these themes, we aim to provide players with a relatable experience that mirrors the complexities and diversity of their own lives.

Certain themes and action within the game's narrative can be perceived as cliché, such as 'the power of friendship' however, the writing team embraces these clichés for several reasons. Firstly, it's challenging and often unnecessary to avoid clichés entirely, especially when they work well within the context of the narrative and its messaging. Additionally, *Clean Sweep's* inherently wacky premise utilizes familiar tropes to maintain the game's accessibility and prevent the world

and its characters from becoming too disjointed or chaotic. Finally, depicting cliches in a comedic context can be used to portray the goofy pillar of our game. Because cliches are inherently familiar, exaggerating or drawing attention to their elements effectively communicates comedy to a broad audience. This familiarity also allows for easier perception of the themes we are trying to convey.

3.2 Consultation

To properly address the topics of mental health and healthy relationships within our narrative, we undertook research to ensure that these themes were portrayed accurately and sensitively. This involved a combination of self-driven online research, where we gathered and analyzed materials relevant to each character's development, and guided research—facilitated through consultations with WPI Student Counseling Center (SDCC). The Counseling Center provided guidance on pertinent psychological topics that align with the overarching wellness themes of our game, such as Acceptance and Commitment Therapy (ACT). This form of therapy emphasizes accepting what is out of one's personal control while committing to actions that enrich one's life, making it a fitting therapeutic approach to embed within our game's storyline and character arcs. Through this research, we aimed to provide accurate representation and provide players with subtle tools and understandings that could positively impact their own personal growth.

For the theme of diversity, it was crucial to the narrative team to engage directly with members of the communities we were portraying to ensure authentic representation. To facilitate this, we collaborated with the Office of Diversity and Inclusion (ODIME) to learn practices in representing diverse groups and to develop a focus group strategy. With the guidance of ODIME

our approach to focus groups entailed reaching out to various clubs across the campus, presenting them with our diverse cast of characters, and inviting them to sign up to discuss the characters of their choice (see Appendix D). This method allowed participants to choose the characters they felt most connected to, which created the groundwork for a more personalized and insightful discussion.

The focus groups provided varied perspectives due to the participants' diverse backgrounds, resulting in valuable feedback. One such suggestion was to integrate Spanish words in the dialogue of our half Puerto Rican character, Cain. Additionally, we directly consulted with members of these communities. While these discussions lacked multiple viewpoints, they allowed for faster and more direct communication. This dual approach not only enriched our characters but also ensured that our portrayal was respectful and resonated with a wide audience.

3.3 The World of *Clean Sweep*

The world of *Clean Sweep* is an alternate version of our reality where popular cleaning products are personified, and janitors are akin to Ghost Busters. In this universe, janitors are called upon when germs appear, a phenomenon triggered by people experiencing intense negative emotions. Figuring out how germ summoning integrated with the world and its connection to combat was a key focus for the narrative team.

While exploring the relationship between combat and the narrative, we drew inspiration from various media sources. Zombie apocalypse media, for example, often presents a dangerous world with constant combat opportunities. The universe of *Scott Pilgrim vs. the World*, while maintaining a seemingly normal reality, features unexpected, over-the-top action sequences.

Unlike the dire scenarios often depicted in zombie media, *Scott Pilgrim* displays fights that are comedic and highly stylized. The world of *Clean Sweep* combines the lighter tone of *Scott Pilgrim* with the constant threat of combat found in zombie apocalypse narratives.

When discussing the concept of negative emotions summoning enemies, we were reminded of the children's show *Miraculous Ladybug*, where the antagonist empowers individuals with manifestations of their negative feelings to wreak havoc on the city of Paris. Despite the regularity of these attacks, the city's residents treat each incident with significance. This blend of commonplace yet serious combat scenarios is found in the dynamics of *Clean Sweep*.

While *Miraculous Ladybug* effectively utilizes negative emotions as a plot device to drive conflict and character development, it does not explicitly address the inherent value or unavoidable nature of these feelings. The show tends to frame negative emotions as tools for villainy without exploring their complexity or the fact that experiencing such feelings is a normal and healthy part of being human. Expanding on this inspiration, *Clean Sweep* incorporates the idea that negative emotions can summon challenges while also exploring their dual nature. In our game, negative emotions do trigger the appearance of germ enemies, but we emphasize that these emotions are not inherently bad or to be avoided. Instead, they are natural and unavoidable aspects of human experience.

To enhance this concept, *Clean Sweep* incorporates narrative elements that encourage players to acknowledge and manage their emotions rather than fearing or suppressing them. This is most evident in the final boss battle when the characters must use their negative emotions to defeat the main antagonist. By communicating this message, players can learn to accept and ultimately overcome the challenges spawned from these emotional states. This approach allows

us to address and critique the simplistic good vs. evil dichotomy often seen in children's media, and calls attention to the dangers of toxic positivity.

Toxic positivity refers to the constant pursuit of happiness and positivity in a way that is unrealistic and potentially harmful. To weave this understanding into our narrative, we drew inspiration from the show *Supernatural*, where celestial beings often exhibit morally ambiguous or even murderous behaviors comparable to the demons they oppose. This portrayal of heaven and angels inspired us to create a heaven-like realm called Sanitopia to serve as a space to discuss the concept of toxic positivity. This dimension differentiates *Clean Sweep* from its inspiration and provides a more rounded and realistic portrayal of emotional experiences.

3.4 Structure

The writing team worked closely with design to develop the narrative elements of *Clean Sweep*: The main story, character routes, hangouts, quests, and pooled dialogue. These features work together to drive a player's progression and communicate the themes of the game.

3.4.1 Main Story

The main story takes place in the player character's mansion, recently inherited from their deceased aunt. It is split into a prologue and four chapters. Situated on the mansion's first floor, chapter one centers on the retrieval of scattered research items belonging to the aunt. It concludes with a pivotal story beat revealing the purpose behind her research. Transitioning to the second chapter and second floor of the mansion, the focus shifts to Blanche, a character formerly aligned with the 'good guys' who is now working for the enemy germs. Throughout this

segment, the player and their companion Beau sways Blanche to join their side and she subsequently reveals the true villain and his motives. During the remaining two chapters which occur in the mansion's basement lab and the town sewer system, the player works towards defeating the final boss, 001.

Each chapter culminates in a boss fight against formidable germs that require significant negativity to summon, hinting towards and supporting narrative developments. Dialogue occurs before, during, and after these boss fights to develop character motivations. These battles serve as markers of progression and provide insight into the evolving storyline.

The levels within each chapter, grouped into thematic room settings such as the dining room and kitchen, offer opportunities for storytelling and character development. The writing team worked with design to determine which rooms to include in the mansion and how they would be grouped into levels. Players discover items scattered throughout these environments. The aunt's research, which is found in every level on the first and second floor of the mansion, gives insight into the aunt's work and relationship with the Q-Pid Corporation. The additional environmental elements further develop the aunt's character, her connections with the people of Cleanland, and provide information on the lore of the world.

In addition to boss fights, the narrative transitions between chapters with key moments marked by significant revelations and character dynamics. Notably, the introduction of communication with Q-Pid via a fax machine at the end of the first floor. This feature serves to allow Q-Pid to be a more active character and underscores the evolving narrative landscape.

As players progress to the second chapter, the narrative shifts its focus to the character of Blanche. While elements of the aunt's research remain present, the storytelling becomes

more dialogue-heavy, with the player's sidekick, Beau, offering insights into this new mystery, given that Blanche was his former coworker. This character-centric section aligns with the character-driven nature of *Clean Sweep*, emphasizing active character development and dialogue interactions. Players will have needed to progress through dialogue heavy and character-centric character routes to access this chapter of the narrative, suggesting that they are primed for storytelling that centers on character dialogue and active character development.

Upon completing this chapter, Blanche joins the player's side, revealing 001's plan to take over the town of Cleanland. By this point, players will have formed strong connections with the town and its residents, making this revelation particularly impactful.

After Blanche joins the players side, the third and fourth sections are focused on defeating 001. The third section is essentially a fetch quest and is relatively short to maintain a fast pace following revelations from the end of the previous chapter. Here, the storyline reconnects with the aunt's research, as players seek to recover the mysterious 100 percent formula, she developed to defeat 001. This section's setting shifts to a basement lab environment, marking a departure from the previous mansion rooms and signaling a shift towards actively saving the town.

Upon retrieving the formula, the fourth section sees players traversing through town sewers to thwart 001's plan. This setting adds depth to the town's lore and visuals, reinforcing the theme of stepping outside comfort zones to confront challenges. This exchange of support highlights the features of a healthy, supportive friendship

Because 001's development has since been confined to references from other characters, the final boss battle is preceded by dialogue from 001. Dialogue exchanges with 001

before and during the final battle provide depth to his character and contextualize his dynamic with Blanche.

The final boss fight sees the player use the 100 percent formula on 001 to no avail. The writing team initially planned to have the formula work, but as our themes developed, we wanted to emphasize the duality of negative emotions and the importance of embracing one's vulnerabilities. By utilizing negative emotions to thwart 001's plans, players learn valuable lessons about acceptance and resilience, concluding the narrative journey on a poignant note.

The main story underwent significant changes during the first half of production as themes were established and the writing team conducted research. Crafting coherent character motivations proved challenging, leading to multiple revisions of the storyline. Midway through production, the decision was made to reduce the number of levels on the first and second floors from six on each floor to five. While this adjustment streamlined the storytelling on the first floor, it posed challenges for the narrative arc on the second floor. Given that the second-floor storyline revolves around Blanche's allegiance transition from to the germs to the player's crew, we had to carefully adjust the pacing to ensure her decision felt realistic and well-developed

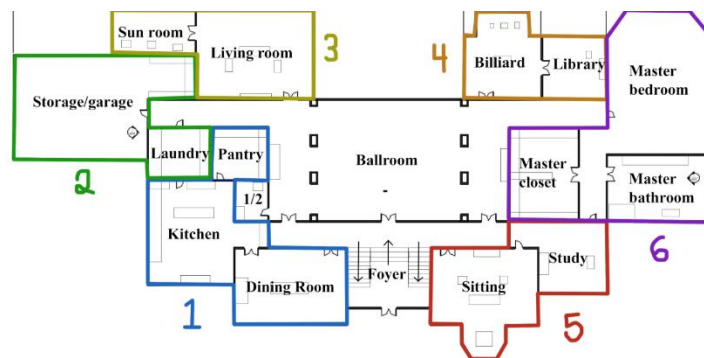


Figure 3.1: Initial mansion layout with 6 levels

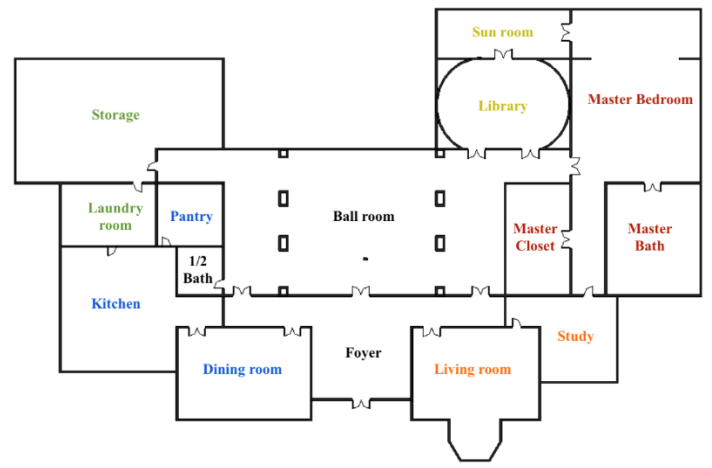


Figure 3.2: Revised mansion layout with 5 levels

While the main story has been developed it is not entirely written. As later production shifted towards creating a vertical slice of the game, the main story took a backseat in priority. We recognized that showcasing gameplay mechanics and focusing on the town characters were more crucial for demo purposes, aligning better with player expectations. Initially, having an incomplete main story felt like a failure in production, but upon reflection certain benefits emerged. When adjustments to levels were necessary, having the story in a flexible state allowed for easier adaptation. Allowing time for reflection on narrative decisions was also very helpful with development. There were a lot of great ideas, and time and reflection allow for evaluation of what works and what should be removed. Allowing time for reflection on narrative decisions was also very helpful with development. There were a lot of great ideas, and time and reflection allow you to realize what works and what should be removed.

Throughout production, adjustments in the main story illustrated the dynamic nature of narrative development. Themes were refined, character motivations were crafted, and the

narrative underwent revisions to ensure coherence. Despite challenges such as reworking level layouts and pacing, the team remained dedicated to delivering a compelling storyline that resonates with players.

3.4.2 Character Routes

Clean Sweep always had a strong focus on its cast of characters (see Appendix E). By following a character's route, players can deepen their relationship with that character, which translates to improved fighting abilities during combat. These routes are designed to highlight the power of friendship and positive support, unfolding linearly while incorporating choices to keep players engaged and offer fresh information and humor on subsequent playthroughs.

Each route consists of seven events where the player learns who a character is, their motivations, goals, insecurities, and conflicts. The initial three events focus on acquainting the player with the character (see Appendix F). The fourth event, referred to as “cleaning the house,” involves the player helping to clear the character's home of germs which are a manifestation of the character's negative emotions. The appearance of germs prompts the character to reveal more profound personal issues that influence the remaining events. The last three events have the characters acting on their “cleaning the house” revelation with support from their friend, the player. The final event serves as a grand conclusion to their storyline, but not necessarily to their personal development.

The structure of character routes shifted multiple times during development. One of the largest changes was cutting romance. The original idea for *Clean Sweep* leaned much more into its dating simulation influence. After the “cleaning the house” event, players would have been able

to choose to pursue a romantic relationship with the character. There were several reasons why this idea was cut. The first was scope. Including both romantic and platonic paths for each character proved daunting in terms of the required writing and research. Although we were excited by the thought of portraying positive romantic relationships, doing them justice in addition to the various topics touched upon in the friendship routes was not realistic. More critically, initiating a romance in the aftermath of such a vulnerable event raised ethical concerns, as it could be perceived as exploiting the character's emotional state.

After romance was removed, the length of character routes continued to fluctuate. Initially planned to include nine total events, the number of events occurring before and after “cleaning the house” changed several times, most notably becoming longer for a large part of production with the addition of intro quests. Upon revision, adjustments were made by cutting events both before and after the “cleaning the house” event to maintain balance, scope, and to create tighter narrative focus.

Character routes were also a place to explore the relation between narrative and combat. In addition to the “cleaning the house” event, encounters with germs occur at various points along a character's route. This not only unifies the combat and narrative but enriches the world, giving the writers another device in their storytelling, and reinforcing the rules of the *Clean Sweep* universe.

3.4.3 Hangouts

With friendship being such an important part of the narrative and gameplay, we wanted to include as many opportunities as possible for friendship development. To address this, we

introduced "hangouts," which are brief yet meaningful interactions with the main playable characters that are designed to deepen friendships. Initially conceived as to have lengths like route events, scope considerations necessitated trimming their duration and complexity. Embracing our core design principle of goofy, we transformed hangouts into humorous, lighthearted moments. These interactions not only offer players further insights into the characters' personalities but are used to reveal how characters within the town interact with one another, enriching the community of Cleanland.

3.4.4 Quests

The inclusion of quests was a collaborative decision between the writing and design team. During quests, players not only receive money and items to use for combat but are also encouraged to explore the town and interact with its inhabitants. As players may have developed favorites among the playable characters through their character arcs and combat styles, quests provide an opportunity to expand their network and get to know more characters. Beyond the playable characters, quests prompt players to engage with all the inhabitants of the town, enriching the environment and making it feel more lived-in.

Quests can also be seamlessly integrated not only with each other but also with the main story. As the main story primarily unfolds within the mansion, there is a risk of it feeling disconnected from the rest of the town. Incorporating quests into the main storyline encourages players to interact with the town's residents, ensuring the narrative components remain interconnected.

Three quests are featured in the *Clean Sweep* demo. For these quests, the entire team brainstormed which characters would be included in each quest based on their relationships and how we wanted to exhibit their personalities. The writer in charge of the character would write their dialogue. For quests that involved characters with different writers, these writers would write together to ensure that the story was coherent and goofy.

3.4.5 Pooled Dialogue

Pooled dialogue played a crucial role in creating the town of Cleanland. A town populated with diverse characters would feel incomplete if the people did not speak. Pooled dialogue not only provides a platform for individual character personalities to shine but was also a method to further immerse the player in the town.

Like other narrative features, writers responsible for individual characters were tasked with crafting their pooled dialogue. In the demo we featured pooled dialogue from characters such as Wave and Olive, owners of shops in the market, alongside the playable cast. For these characters, writers with a more dedicated focus on narrative work contributed to their lines, collaborating with the entire team to ensure the personality and voice was agreed on and that we had a unified understanding of these characters.

3.5 Setting and Locations

Cleanland is a vibrant suburban town in middle America where it is always sunny. As players venture through Cleanland, they will encounter various locations where residents gather for work

and recreation. The decision to include certain locations and their contents was guided by their relevance to the main narrative and the development of character arcs.

Location	Description	Status
Mansion	The inside of the mansion is where most of the main story occurs. It is comprised of a first floor, second floor, and basement. Rooms within the mansion are grouped to form levels. Each time the player clears a level, they can find items in the rooms that reveal new information to progress the main story.	In-game
Town square	Home of the town hall and a large soap tree, it acts as the access point to every other location in Cleanland. Within the Town Hall, players can access janitor jobs and often have opportunities to interact with Mayor S.C. Jenkins, Roxi, and Windy.	In-game
Plaza	The plaza contains three significant buildings: Ph Level Up, Scrub-a-Licious Boba, and The Periodic Table. At all three locations players can acquire food items to use in combat, interact with NPCs, and playout route events.	In-game
Market	Players can interact with characters and purchase items from four NPCs running stalls in a farmers' market setup.	In-game

Beach	The beach features a boardwalk with several shops, the most significant of which is the building that houses the Squeaki Tiki bar and the Bay Wash night club. On the beach, players will find an outdoor gym, volleyball court, and the home of the NPC Francisco.	In-game
Park	The park contains several locations including a skatepark, tennis court, and gazebo. Its primary function is to be a place where route events occur, and the player can interact with NPCs.	In development
Sewers	The sewer system running beneath Cleanland becomes accessible in the third act of the mains story. It juxtaposes the bright aesthetic of the town and is populated by germs.	In development

Figure 3.3: Cleanland location descriptions and their current status.

Once locations were established, the next step was to populate them with characters. Crafted schedules detailed where each character would be at every hour of each day of the week. Certain locations hold particular significance for specific characters and their story arcs. For instance, Windy spends a considerable amount of time at the Ph Level Up arcade, while Mr. Tidy is most often found at his restaurant, the Periodic Table. These schedules are not static; they evolve as players progress through character routes, reflecting the characters' developing behaviors and responses to the narrative. This dynamic scheduling adds a layer of realism and depth to the world, making Cleanland feel like a living, breathing community.

3.6 Characters

Characters are one of the primary points of focus in *Clean Sweep*. The writing team was tasked with developing characters and relationships that are dynamic, diverse, and reflect the overarching narrative themes. Within the cast there are three distinct character types: main story, character route and additional non playable characters (NPCs).

3.6.1 Main Story Characters

3.6.1.1 The Janitor

The player takes on the role of a janitor who has just arrived and Cleanland after inheriting a mansion from their recently deceased aunt. The Janitor acts as self-insert for the player. Because of this, the Janitor is gender-less and is addressed using they/them pronouns by the in-game characters. Although the Janitor is a self-insert, the writing team still includes moments of personality to maintain the goofy atmosphere of *Clean Sweep* and to prevent the friendship development aspect of the game from feeling stifled and unnatural.

3.6.1.2 Beau

The first significant character the player befriends is Beau, a cherub-like being from the Q-Pid Corporation who is only visible to the Janitor. Beau is the deuteragonist, acting as the player's companion, providing information about the world, and prompting actions. Beau is primarily involved in the main story but occasionally appears in other parts of the narrative to guide the player and provide moments of comedic relief. His involvement in the players' interactions with

other characters is limited as not to detract attention and meaning from the friendship development.

With a story so focused on defeating negative emotions, we wanted to recognize that it is not only healthy to feel negativity, but it is also unhealthy to strive for constant positivity. Beau's arch recognizes this as he becomes aware of the toxic positivity being promoted by his boss, Q-Pid at the Q-Pid Cooperation. At the end of the main story, we see Beau taking a stand against Q-Pid and embracing necessary negative emotions to defeat the main antagonist.

3.6.1.3 Q-Pid

Q-Pid runs the Q-Pid Cooperation located in the heaven-like plane of existence called Sanitopia. While not physically appearing in-game, communication is opened to Q-Pid at the end of the first chapter. It is during the second chapter when the player learns more about Q-Pid and his promotion of toxic positivity. While character archs and the fighting of germs support the process of working through emotions, having a clearer example of how being completely positive can be bad supports our messaging and provides an additional partial antagonist to enrich the main story.

When exploring how to portray Q-Pid, we researched toxic positivity to better understand what it is and how it can manifest in the workplace. Thinking positively is not a terrible thing, in fact, it is used in cognitive behavior therapy to help people with depression and other mental health issues (Sipe). Positivity becomes toxic when it is, “presented as a demand that we must be positive at the expense of other feelings” (Reynolds). In the context of *Clean Sweep*, Q-Pid does not fully acknowledge the feelings and concerns of others, instead creating unrealistic expectations for how they should act and feel. With this being a workplace dynamic, the purpose

of his toxic positivity is to promote productivity and happiness in workers. In the Forbs article, *Toxic Positivity At Work: Examples And How To Deal With It*, Erik Pham mentions several ways leaders can avoid toxic positivity in workplace, like creating a safe place for communication and being honest with employees (Pham). We took this information to inform us on what Q-Pid should not do. As characters like Beau start to identify these actions as negative behavior, the player begins to do the same.

3.6.1.4 Blanche

Like Beau, Blanche is a cherub-like being, but is a former employee of the Q-Pid cooperation. She first appears in the main story's second act as a tritagonist and foil character to Beau, but her presence can be seen in the first act. She starts as an antagonist working for the germs but becomes an ally in the second act. Gaining her friendship reveals more information about the germ world, which initiates the third act. As a foil to Beau, she exposes the faults of working for the Q-Pid Cooperation and its promotion of toxic positivity.

One of the primary themes in *Clean Sweep* is the depiction of healthy relationships. The writing team aimed to underscore this theme by incorporating illustrations of unhealthy dynamics, thereby raising awareness. A notable instance of this is evident in the complex relationship between Blanche and the game's primary antagonist, 001, which serves to portray various characteristics of an abusive relationship.

3.6.1.5 001

001, pronounced double O one, is the main antagonist of *Clean Sweep*. His name is derived from the James Bond code number 007 and the ability of many popular cleaning products to kill

99.9 percent of germs. 001 is the one percent that cannot be defeated by a janitor or any known cleaning product. He resides in the Bacterial Plane but appears in Cleanland for the third act of the main story as the main antagonist and final boss. 001, while extremely good looking, was also intentionally designed to not portray any specific ethnicity to ensure that no group felt villainized, especially in a game about positive representation and fighting a force of uncleanness.

His relationship with Blanche portrays elements of an abusive relationship as he is able to sense negative emotions and attempts to use Blanche's hatred towards Q-Pid and the corporation to control her. We hope that by including this relationship within the narrative we can bring awareness to unhealthy dynamics. To do so, we researched what toxic and abusive relationships are and how they may present themselves to write informed character motivations and dialogue.

In an abusive relationship, one partner is in control by using abusive behavior with awareness of their actions (Rahman and Troy). Manipulation is common in abusive relationships as a form of mental abuse; "non-physical behaviors that are meant to control, isolate, or frighten"(What Is). The common tactic of controlling a person's time and actions can lend itself to a boss-employee dynamic between Blanche and 001, but there are additional tactics implemented by 001 that play on Blanche's insecurities. Making Blanche question her reality and recollection of past events can be used to intensify her hatred towards the corporation, further securing her loyalty to 001. Love bombing, an over display of affection, in the form of exaggerated compliments would provide Blanche with the validation that she was not receiving from Q-Pid (Signs of). When Blanche is a loyal employee, additional elements of a toxic relationship can be used as a means of control, such as withholding praise, giving the silent treatment, and "moving the goal post" so that she is constantly working for the approval of 001 (What Is, Holland et al.).

In understanding how 001 would treat Blanche as an emotionally abusive employer, we learned the complexities of leaving an abusive relationship. In the main story, the player and other characters are involved in convincing Blanche to leave 001, but in an abusive relationship it is important that the person leaving is the one making that choice. According to an advocate from the National Domestic Violence Hotline, “Rather than trying to “save” them, consider how you might empower them to make their own decisions about how to proceed” (Kim). Making the choice for them takes away their autonomy. This must be considered when developing Blanches arc. The characters can present themselves as a means of support, but Blanche quitting must be a decision that she comes to on her own.

3.6.1.6 Germ Bosses

In addition to 001, there are three bosses that the player encounters as they reach the final fight against 001: The Poop Monster, Typhoid Mary, and Cerberlyssus – names still in development. Each boss is based on a disease with a connection to water, a foreshadowing to the location of the final boss in the sewer system.

3.6.1.6.1 The Poop Monster

The Poop Monster is the first boss in *Clean Sweep*, and takes inspiration from dysentery, particularly the diseases’ symptoms and common contraction through water. He appears at the end of the first floor from the toilet in the master bathroom and has two forms: constipation and diarrhea. In the first phase, constipation, its form is condensed to resemble that of a human but is overall goopy. He mimics the intonations of Elvis, as canonically, The Poop Monster killed Elvis on the toilet (which is unfortunate because he was a huge fan and just wanted an autograph, but the

fright resulted in Elvis' fatal heart attack). Parents often tell their young children to eat their fruits and vegetables to avoid having the same fate as the music star... but that is beside the point. He attacks using projectiles, and in this dense state, is harder to damage. In the second phase, diarrhea, he becomes more amorphous and blob-like. He is easier to damage but attacks quicker and with a wider range. We understand that poop is gross; the art team intends to heavily stylize the content to ensure it emphasizes being silly rather than disgusting.

3.6.1.6.2 Typhoid Mary

Typhoid Mary is the second-floor boss spawned by Blanche in the peristylum, and is directly inspired by typhoid fever and Mary Mallon, popularly referred to as Typhoid Mary. The typhoid fever itself traveled through water and was highly contagious. The symptoms include fever, stomach pain, muscle aches, delirium, fatigue, nausea, vomiting, and more. To incorporate these elements, she uses water attacks, and has special moves that let her inflict various status effects, such as slowness, poison, and the unique ability to invert the player's controls.

However, the story of Mary Mallon is not cut and dry. Mary was an Irish immigrant to the United States who worked for wealthy New York banker Charles Henry Warren as a cook for his family during the height of typhoid fever. She was infected and contaminated most of the environment around her as well as the families she currently and previously served. When prompted, Mary refused to cooperate with the Department of Health and denied being ill, which led to the direct infection of at least 122 people. However, the Department of Health failed to inform her properly of the importance of her treatment, performing test after test on her in the hopes that she would be able to cook again, and she was villainized by the media and medical dictionaries as a disease carrier (Marineli).

We do not want to capitalize off anyone's suffering in *Clean Sweep*. As we further develop this character idea, we plan to write with our research in mind. Mary faced great prejudice, and we want to illustrate that to the audience and not make a mockery of what she endured. If we are unsatisfied with our ability to do so, the writing team is prepared to scrap the character idea completely and design a new boss. The most essential element of *Clean Sweep* is that we provide good, accurate representation of all the communities and people present in the game.

3.6.1.6.3 Cerberlyssus

Cerberlyssus is a three headed germ dog inspired by the mythological Cerberus and rabies, a viral disease caused by Lyssavirus. Cerberlyssus is a singular beast composed of three different normal size dogs. It uses animalistic attacks that reflect the side effects of rabies, and the differentiates its moveset based on which dog is in control. Upon clearing the boss, the three germ dogs become friendly and live in the /janitor's mansion. The breed of dogs that converge to form Cerberlyssus is not yet decided, but current ideas for the three green dogs include a small cotton ball Pomeranian, an extremely muscular but secretly gentle Pitbull, the mop-esque Komondor, and an Afghan Hound designed to resemble a feather duster.

3.6.2 Character Route Characters

The other significant type of character are the people in town with character routes (see Appendix E). There are nine characters with routes, each with their own theme or dilemma. The problems these characters face was chosen to appeal to players in their late teens and early twenties, but many of the themes can be related to by people of all ages. As a route progresses, a character takes steps to improve their life with the player by their side. It was an intentional

decision to have the characters themselves make character developing changes in their lives, with the Janitor merely acting as the catalyst for critical thinking. This way, the Janitor would not be a “savior” who undermines the importance of the characters’ personal journeys, but instead a friend who is there to support them as they make choices for themselves.

Character route characters were split between members of the writing team. Initially this meant three characters for each member of the team, but after gaining a fourth member as an independent study, they were given one. Distributing entire characters allowed writers to become an expert in their personality, interests, and voice. When design decisions had to be made about a character, the writer in charge of their development was consulted. Having one point person helped with communication, decision making, and keeping a consistent vision. That said, character development was open to input from other members of the writing team.

For each character, we researched their background and route themes and consulted about their representation and the portrayal of their problems and improvement. Consulting was done with members of WPI’s Student Development and Counseling Center, Office of Diversity, Inclusion, and Multicultural Education, and a student focus group.

3.6.2.1 Charming

Charming is the first route character the player encounters in *Clean Sweep*. She and the player are childhood friends. Charming is an incredibly optimistic person with a bubbly personality; A people pleaser who works hard for approval of others. Charming’s route is about burnout, learning to put herself first, and realizing when people are taking her for granted.

We wanted to show through Charming's route and interactions with others that she is being taken advantage of heavily by others and is taking on too much work and stress. Charming

experiences other characters like Clara lying about her order to get a large drink at Scrub-a-licious. We find there and through that the mental tax it has on Charming and how quick she is to try and solve any issues or problems. Through her work, perfectionism, and striving to be very friendly and personable with everyone, she starts to experience burnout. Burnout is described as, "an affective reaction to prolonged exposure to stress at work; that is to situations in which job demands exceed individuals' adaptive resources" (Shirom 270). Shirom also continues by saying, "...the core content of this affective reaction is the feelings of emotional exhaustion, physical fatigue, and cognitive weariness that result from gradual depletion over time of individuals' intrinsic energetic resources that occurs at work" (Shirom 270).

As both a gag and big reason as to Charming overworking herself, she is the unknowing owner of Scrub-a-licious. She handles all the upkeep and stress that comes with running a business. The cycle of people using her, running the business, and trying to remain bubbly on top of everything contributes to her burnout. We wanted her role as a barista at Scrub-a-licious to inform her dedication to serving those around her. In many ways, she models the ideal Starbucks barista by putting the customer first, fixing any mistakes, and taking fault for any errors regardless of if they are her own. By the end of the route, we aim to still have her working but taking more time off, realizing when people in and out of work are using her, and finally standing up for herself against others.

3.6.2.2 Vic

Charismatic and cocky, Vic thrives in the spotlight as half of an identical twin heartthrob duo. The constant pressure of maintaining his image leaves Vic torn between his public persona

and authentic self. Vic's route is focused on exploring his individuality and finding the courage to pursue his interests.

Vic was one of the least developed characters when writing began. Based on Spic and Span, we knew he had an identical twin and that they were both rich, mischievous goofballs. Research was easily ingrained into Vic's character due to his underdevelopment and played a significant role in figuring out who he was.

Going off Vic being a twin, research began with a focus on individuality. Individuals' thoughts, emotions, and behaviors are significantly influenced by their family and social groups. The Bowen Center for the Study of the Family explains that the degree to which a person conforms to group norms depends on their level of self-differentiation, a psychological concept reflecting the development of a personal identity distinct from others (Differentiation). Individuals with lower self-differentiation are more susceptible to external influences and tend to either adapt excessively to gain approval or exert control over others to create conformity. This idea of adapting to gain approval inspired the idea that Vic had some level of fame and that he had to put on a persona to appease his fans. Further on this topic, a person's level of self-differentiation is formed during childhood and adolescence through interactions within the family, and it remains stable throughout life unless consciously altered (Differentiation). This led to the development that Vic and Van gained their fame at a young age playing the same character to bypass child labor laws, as seen with identical twins Mary Kate and Ashley Olson who both played the role of Michelle Tanner in the sitcom *Full House*.

Growing up as an identical twin under the early pressures of fame led Vic to struggle feeling like his own person and from with low self-differentiation. His journey highlights several indicators

of low differentiation, such as concealing feelings, talking to friends about relationship problems instead of your partner—or in his case, his twin Van—and demanding, directly or indirectly, compliments and praise (Taylor and Abulhosn). These behaviors were pivotal in shaping Vic’s character and actions. The demand of praise especially influenced his cocky persona (see Appendix F). To better understand how Vic could evolve over the course of his route, it was essential to consider strategies for increasing self-differentiation. According to Licensed psychologist Paul-Roy Taylor this can be achieved by showing your true self, especially to those who matter to you (Taylor and Abulhosn). This approach challenges individuals to contemplate whether they prefer others to appreciate a contrived persona or their genuine identity. In Vic's route, this theme unfolds as he embraces his passion for serious acting, initially hiding it from his brother but eventually performing publicly. This journey reflects Vic's progressive separation from his twin and signifies significant personal growth throughout his narrative arc.

3.6.2.3 Mr. Tidy

Mr. Tidy is the owner and head Chef of the high-end restaurant, The Periodic Table. Tidy is highly dedicated to his work and practically lives at his restaurant, sleeping most nights on a cot in his office. His route explores the pitfalls of a work-centric life and using work as an escape from reality, while emphasizing the importance of taking breaks.

Tidy stands apart from the other playable characters for a couple reasons. At fifty-two years old, he is much older than the other playable characters. Additionally, he is based on two existing characters; Mr. Clean and Walter White from the hit television drama Breaking Bad. Because of this, the development of his character and personality began with these characters.

There is lore behind Mr. Clean. According to the official Mr. Clean website, his adoptive parents instilled in him the belief that, “to be the best at what you do, you need to work harder than everybody else” (A MAN). This intense drive relates to the darker portrayal of Walter White in the later seasons of *Breaking Bad* when he embraces his drug-dealing kingpin persona, Heisenberg. Unlike Mr. Clean, however, Walter White’s pursuit of success proved destructive to himself and those around him. Drawing inspiration from Walter’s later traits, Mr. Tidy is depicted as a hardened and standoffish, setting him apart from many of the other playable characters.

Initially, there was the intention to mirror Walter White’s background in teaching and chemistry, but we decided to diverge from the direct reference, instead choosing to make Mr. Tidy a chef. The name of his restaurant, the Periodic Table, fits with the cleaning theme and pays homage to Walter White’s relation to chemistry. The high-pressure role of a chef was chosen for Mr. Tidy because it aligned with the relentless work ethic promoted by Mr. Clean’s which also includes the mantra: “if something’s worth doing, it’s worth doing right” (A MAN). The profession of a high-end chef is often depicted in the media as one that demands perfection and a significant personal sacrifice, reflecting the challenges faced by Mr. Tidy in balancing his professional and personal life.

While several character routes involve themes of work-life balance, Mr. Tidy’s story is focused on using work to avoid difficult parts of life. This focus allows for an exploration into the impacts of overwork, specifically how it can lead to work-related anxiety and eventual burnout. According to Licensed Clinical Mental Health Counselor and Supervisor, Hailey Shafir, burnout, “occurs when workplace stress has become chronic and begins to negatively impact a person’s performance” (Shafir). This understanding incentivized Mr. Tidy’s route to center around a lack of

inspiration to create new recipes. Workplace Strategies of Mental Health highlight that employees experiencing burnout may deny their suffering by telling themselves lies, such as convincing themselves that, “No one else can do this” (Ricketts). This self-deception supported the decision to include the sous chef character, Jaxy. Based on Jesse Pinkman from *Breaking Bad* and the cleaning product Ajax, Jaxy is used as a vehicle for Mr. Tidy to develop positive work relationships and take time off, actions Shafir claims mitigate work anxieties (Shafir). By the end of his route, Tidy takes steps to improve his work-life balance, such as entrusting Jaxy with more responsibility at the Periodic Table and taking time to disconnect from work. These changes lead to a happier and more inspired Mr. Tidy.

3.6.2.4 Soap

Soap N’Water is an entrepreneur with an unyielding pursuit of success at the expense of basic life experiences. During their route, Soap begins shifting away from a “live to work” mindset to explore the simple things they have missed out on. Despite their competence and intelligence in business dealings, Soap's lack of life experience presents challenges in everyday social interactions.

A significant aspect of Soap’s character is their identification as nonbinary. As none of our team members identify as nonbinary, we sought guidance from various sources to ensure accurate and respectful representation. Our research explored what gender identity is, what it means to be nonbinary, the different experiences of nonbinary individuals, and how to be supportive of people who identify as nonbinary. Sources such as the *National Center for Transgender Equality*, *Choosing Therapy*, and the *Trevor Project* provided valuable insights. In addition to research, we consulted with individuals from the nonbinary community to gather feedback and ensure that our portrayal

was authentic and positive. This collaborative approach was essential in ensuring that the character of Soap resonates with and respects those who identify as nonbinary.

Another significant aspect of Soap's character is their lack of social skills. The decision to depict Soap as socially inept was a part of their character from early development. During playtesting, the dialogue filled with business jargon and an exaggerated lack of social awareness elicited laughter. Aligning with the game's goofy design pillar, this feedback led to a deliberate emphasis on Soap's social awkwardness.

Following this decision, research to develop Soap's personality included social awkwardness and introversion. Because overcoming social awkwardness is not the focus of Soap's route, we did not want to portray this aspect of their personality in a negative light. In fact, research revealed how social awkwardness and introversion can be an asset. In his book, *Awkward: The Science of Why We are Socially Awkward and Why That's Awesome*, psychologist Ty Tashiro highlights the strengths of socially awkward individuals, emphasizing their attention to detail, depth of thought, and strong empathy. (Tashiro). These characteristics influenced Soap's personality to be caring and earnest. Licensed social worker Silvi Saxena builds on Tashiro's points by stating that, "introverts can be more insightful, good listeners, and better-decision makers than their extroverted colleagues" (Saxena). By portraying Soap, a successful entrepreneur, as socially awkward, we aimed to humanize their character and convey Tashiro's notion that success can be achieved by individuals who may not excel in traditional forms of communication. While Soap N'Water may hesitate or feel embarrassment due to their introversion and social awkwardness, they are not prevented from stepping out of their comfort zone and trying new things.

3.6.2.5 Windy

The player meets the introverted Windy through their mutual friend, Charming. Windy drowns out the constant pressure of life with gaming and writing. As a result, they exhibit signs of self-neglect and isolation. Once they realize the harm of their lifestyle, they begin taking care of themselves.

When created, Windy was initially an NPC and was not part of our main cast of characters. After creating the initial concept of the character, we realized the potential appeal of a gamer character like Windy, as our audience themselves are gamers. The entire writing team agreed to add them as our final "datable" option. Even when *Clean Sweep* removed its dating-sim elements, Windy remained as a playable character. Their route was set to be focused on self-care and gaining self-confidence and has remained faithful with a deeper dive into exploring more of their depression, anxiety, isolation, and self-neglect.

A lot of initial research was done on self-neglect. According to the Washington State Department of Social and Health Services' article, *Self Neglect*, self-neglect is, "Vulnerable adults who neglect themselves [and] are unwilling or unable to do needed self-care" ("Self Neglect"). This includes behaviors like, "not eating enough to the point of malnourishment, wearing clothes that are filthy, torn, or not suited for the weather, living in filthy, unsanitary, or hazardous conditions, [and] not getting needed medical care" ("Self Neglect"). The characteristics of self-neglect that we related to Windy included being assigned female at birth, being depressed, and having a history of poor hygiene and living conditions. Living alone was another big sign, but we ended up having Windy live with SC due to them being still in college. Having Windy live with SC also gave rise to the ability to show how people around others who are living in self-neglect should help and support them.

We wanted to convey the subtle signs of self-neglect through Windy's character to show the player what signs to look out for in the friends and family around them. Through Windy's route events, they don't wear their glasses, they exhibit signs of poor personal hygiene, they are incredibly dehydrated which we show through their extensive drinking of Clora-Cola, they spend too much time alone and isolated, and they exhibit a lack of interest and concern in their life. While wanting to convey to the player the signs to look out for, we also wanted to show how they can help and support people like Windy. Throughout Windy's that Janitor route helps to reduce that isolation through their hangouts. The Janitor also helps Windy accept that they do have a problem. Helping those in self-neglect sometimes just takes letting them talk and express themselves, helping them review options and make their own decisions, and respecting the choices and progress that Windy makes throughout route events.

3.6.2.6 Lysa

Lysa would describe herself as “a mystic of minds, a seeker of souls, shrouded in an aura of divine divination, treading the mortal plane to guide wayward souls through the unpredictable ebb and flow of the tides of energy”. The writing team would describe her as an enigmatic self-proclaimed clairvoyant and musician, obsessed with the bizarre workings of the universe, determined to captivate her audience through abstract instrumental from her soul. Despite her dreams of pursuing a music career, resurfacing memories of her ex have left her with a broken heart and a broken spirit.

Initially, Lysa's personality was wholly centered around her musical pursuits, lack of confidence, and lingering feelings for her ex. However, it didn't feel very compelling; something was missing from her character. Instead, her personality pivoted to embrace her goth, spiritual

side and comedically exaggerating her superstitiousness and enigmatic nature. Thus, her language became heavily saturated with New Age jargon and over-the-top spirituality, almost hinting that she has some supernatural powers to juice up the goofiness. Players had a very positive reception to this personality, and we decided to embrace her superstitiousness as the focal point of her identity while making her music and not letting the opinions of others control her life the focus of her route.

Keith Knight, author and illustrator of *The K Chronicles* comic strip, mentions in one of his comics that Soul Train was a “Saturday afternoon staple for households across America” as “a ‘Black American Bandstand’ that showcased the best in R&B, funk and soul,” disco, and hip-hop. He recounts that “the program was much more than that:” “it was the only place on T.V., ever” that showed Black Americans “simply having fun” (Knight). For this reason, including a reference to Soul Train in *Clean Sweep* as Soul Drain in Lysa’s upbringing was an important detail to developing her character as a young Black musician. Coming from a long line of creatives, Lysa’s life had always revolved around music. Her earliest memory was watching a VHS tape of her father’s featured appearance on an episode of Soul Drain where he performed live as the drummer in a special guest 5-piece band. Since that moment, she wanted to become a musician like her father, though he had long ago retired his drumsticks in hope of a more stable career for his family. Soul Train played a heavy influence in deciding her unique, trademark genre and style. Due to her parents placing strong emphasis on succeeding in school, they intentionally did not invest much into her creative development, leaving a passionate Lysa without direction. Thus, she taught herself to play by feel and created her own genre from components of music she grew up listening to, an experimental blend between jazz, funk, soul, and sprinkles of R&B and classical rock. Lysa

hopes to connect with listeners on a deeper level through her abstract instrumental, as it reminds her of the fonder moments with her family.

This is particularly impactful because at the time of the game's events, Lysa is estranged from her family and lives with her brother Ty "Wave" DePod and his fiancé Olive. This is due to the contrast between Lysa's sexuality and spirituality, and her family's religious beliefs and tolerance for those who deviate from their perceived palatable norm. This resulted in a massive familial schism and Lysa distancing herself from her parents. Journalist Lynn D Johnson remarks that in "the black church, the oldest institution and pillar of the black community, has historically dictated the community's stance on homosexuality – either you don't talk about it, or you condemn it" (Human Rights Campaign). While this is certainly not a belief that all religious Black communities hold, especially considering that churches have become more welcoming, it is still very prevalent and has lasting impacts on Black identity and personal expression (Human Rights Campaign). This "informal church dictum" has resulted in many LGBTQ+ African Americans to "find and create other places to exercise their spirituality" (Human Rights Campaign). For Lysa, this took the form of New Age beliefs and spirituality. She knew she believed in a higher power, but since she felt so unwelcome and unsure in her parents' current church, she turned to tarot, crystals, chakras, energies, and other New Age oddities. We also intentionally wrote Lysa as a lesbian Black character to provide an example of positive representation in the Black and LGBTQ+ communities to hopefully work towards greater acceptance of LGBTQ+ individuals within and auxiliary to the Black community. Regardless of Lysa's past, she has now found a welcoming community in Cleanland. She has a strong support system of characters, including Shawn, Olive, the Janitor, and Wave, who accept and love her for who she is.

3.6.2.7 Roxi

As the face of the town, Roxi has let it all get to her—objectively beautiful—head. In Cleanland, running the town is a family business, and Roxi is convinced she is next in line. I mean, how could she not be? She, like, always has it all—including others to do her work for her. However, a good mayor keeps an eye on everyone, and SC has not let this go unnoticed.

The Covid-19 pandemic resulted in extreme discrimination and violence towards Asian populations that is still perpetuated today, especially in the United States. Thus, it was important to ensure that our half White, half Chinese character was well written and represented. To properly represent Roxi's Chinese American background, research was conducted on the identity group. In the United States, Chinese Americans tend to either fully embrace their heritage, or heavily adopt Western culture. This stems from the American societal norm of "whiteness" being seen as the acceptable default, with deviations from the norm that celebrate cultural identity being frowned upon. In "You're So Pretty, You Must Be Half-White", an article that focuses on an anonymous Chinese teenage girl recounting her experiences, she mentions how several people had mentioned how whenever "people called [her] pretty, they often followed up by asking if [she] was half White" (Anonymous). She also recounts that "when [she] altered [her] appearance and became more conventionally attractive, people began to assume [she] must be part-White" (Anonymous). Chinese Americans and other people of color in the United States are met with the expectation to adapt to Western standards, which results in appealing to Western culture since "no matter how progressive this country may seem in some ways, White is still the standard, and beauty and status are tied to someone's proximity to Whiteness" (Anonymous). Emma Lovewell illustrates her experience in "As a White-Passing AAPI, My 'Asianness' Has Been Questioned and

Doubted,” and notes that while she has encountered racism that made her feel “shame and confusion about [her] identity,” she is “proud to be Asian.” She mentions that her “journey of finding [her] real identity began in college,” and the “many racist interactions” she experienced “lit a fire in [her] to learn more about [her] heritage” and her identity (Lowell).

From these experiences and others encountered during research, Roxi’s character background and relationship with her cultural identity were shaped. Roxi predominantly takes direct inspiration from Lowell and Anonymous, adopting to White standards in her predominantly White high school to be seen by others, resulting in her popularity. By reinforcing standards of Whiteness in and outside of school, Roxi strain on her relationship with her Chinese mother. Moving to college and encountering other Asian students with similar experiences made her realize that she can be both beautiful with her natural features and embrace her culture. Regardless, she remained the it-girl. When the game takes place, she has reconciled with her mother, resuming their close relationship, and is proud of her familial heritage, which is illustrated through her home decor and explored through her route four event.

On Roxi and her family, Roxi heavily benefits from entitlement nepotism. Pankit Gami states in “Nepotism in the Workplace: Concern, Types, Examples, and Cons” that entitlement nepotism is when “the person receiving favorable treatment believes they deserve it.” Roxi has always been Daddy’s Girl, which is exceedingly valuable due to her father’s wealth and influence as a day trader. In Roxi’s parents’ eyes, she was their perfect child and received everything she could ask for throughout her childhood and teen years. This also includes her job at the Town Hall reception desk, as Roxi’s father was responsible for wearing down SC’s resolve until he offered his unemployed niece a job. Roxi’s beauty, social popularity, and special treatment from her parents

allowed her to maneuver through life with a lack of consequences. This manifests as a lack of responsibility and accountability for her actions, only focusing on reaping the benefits and pawning off blame onto someone else. Business expert and startup cultivator Karie Kaufmann states that “nepotism reduces talent and innovation.” She continues, “when unqualified individuals are placed in positions of influence because of their connections, it can result in poor decision-making and inefficiency” which “hampers the growth of the organization” (Kaufmann). This is explored in Roxi’s character through her extremely low work ethic. She uses her social connections to do her generally simple secretary work for her while being convinced that SC will select her to be the next mayor, since running the town is and always has been “a family business.” Roxi often makes careless mistakes because she could not care less and does not consider others in her decision-making process. When things turn out poorly, she deflects blame onto others. Through it all, she still believes that her work exceeds Town Hall standards, and she is met with brutal surprise when SC reveals that he is not considering her to be the next mayor. This ultimately becomes her catalyst for change.

Toxicity is also explored in Roxi’s route through her relationship with her best friend, Clara. Licensed Social Worker, Certified Clinical Trauma Professional, and Certified Oncology Social Worker, Silvi Saxena, writes that, “toxic relationships can take on many forms,” which may include, “mutual disrespect, emotional manipulation, or feeling lonely,” while in each other’s company. Clara and Roxi exist in a mutually destructive friendship. They are passive aggressive towards each other, advocate for bad decision making, and “bring out the bad qualities” of each other (Saxena, Saleh). However, Clara is a more negative force in the friendship, secretly harboring jealousy towards Roxi and using her for her money, which is her primary motivation in her toxicity. While

Roxi is also toxic to Clara, her toxicity is more retaliatory. Yet, their friendship remains unquestioned by each other due to Roxi and Clara feeling alone in Cleanland. Though they claim to have many friends, they keep all relationships surface level out of a place of judgement and elitism. When Roxi eventually cuts Clara off after the events of route four, she realizes for the first time that she is also not a great person. In tandem with recognizing her use-use approach to relationships with others is flawed and harmful, she also begins to take responsibility for her actions. Roxi and Clara's arc shows that sometimes in the end, people are better off without each other.

An important point should be made that Roxi and Clara's relationship is different than the relationship between 001 and Blanche. While toxic and abusive relationships can overlap, toxic relationships require reciprocation of toxicity, such as Roxi and Clara, whereas abusive relationships involve an abuser and a victim, such as 001 and Blanche (Saxena, Saleh). In Roxi and Clara's friendship, no one party is in control of the other, and they do not exhibit manipulation to the degree of undermining each other and promises of reconciling. In fact, they avoid talking about the problems that the other causes and prefer to let them pile up from within instead of attempting to talk it out for risk of a fight, which is ultimately a very unhealthy dynamic. 001, in contrast, has a large amount of power over Blanche, and actively manipulates Blanche without regard for her wellbeing or goals. Both relationships are unideal situations that result in one or more parties getting hurt and illustrate the importance of leaving these relationships as fast as safely possible.

Overall, it was still extremely important to make Roxi a likeable character for players to engage with her. Thus, her personality became rooted in comical entitlement as Cleanland's it-girl

with a genuine interest in helping others, however poorly executed. This takes the form of “giving away” makeovers to help “improve” the people of Cleanland, distributing “constructive feedback” to others, and otherwise feeling like she is the Queen of Cleanland. It was also a key point of Roxi’s character to not change her ways completely by the end of her route. Roxi’s personality was influenced by reaping the benefits of those around her catering to her every need for her entire life; a summer’s worth of personal development would not result in an overhaul of her worldview. Realistically, change takes time, and personalities shape identities. Her flaws are essential parts of her character and make her more human and relatable.

3.6.2.8 Shawn

In a town settled by the rich and successful, Shawn feels like the odd one out. Running away from a personal crisis has left him out of college, work, and most importantly, cash. In need of a job, Shawn’s route is centered on finding motivation.

The character of Shawn, based on Dawn dish soap, underwent significant changes after the writing team brought on our ISP Ethan Chau and transferred to him the task of writing Shawn’s route. In his initial depiction, Shawn was a skater boy taking on odd jobs trying to support his sick mom and siblings whose arc focused on following his passions to become an artist. With the transfer, we wanted to give our ISP the opportunity to shape Shawn into a character based on his own experiences. Given the extensive writing required to develop a character route, it was crucial that the writers felt passionate about the characters and storylines they were portraying. With Shawn being Filipino, our ISP drew inspiration from a Filipino friend to evolve the character into a laid-back, stoner persona who left school due to conflicts with his parents over his lack of effort and direction.

Despite the change of hands, we wanted to preserve Shawn's financial struggles. This aspect of the character adds financial diversity to the upper-middle-class town of Cleanland and is this relatable to our target audience of college-age students who may find themselves in similar circumstances while trying to find their place in a seemingly successful world.

3.6.2.9 Cain

Cain spends his time modeling for Kelvin Clean and working out at the gym; the gym is practically a second home for him. For anyone who's met him, he's the sweetest, jolliest guy around. But, under that facade, he's excessively worrying about his body. He only got such a chiseled body after working out due to years of excessive bullying. Now he's desperately working to maintain that body out of fear of returning to the state he once was.

A lot of Cain's research revolved around body dysmorphia. Cain's route has a big emphasis on learning to love your body and not let outside pressures influence your own view of yourself. Cain was highly influenced by the sudden attention that his body started receiving. For him, this made him feel like his "new" body was perfect. Body dysmorphic disorder (BDD) is described by Neelam A. Vashi MD in her article, *Obsession with perfection: Body dysmorphia, as, "the obsession with perfection"* (Vashi). This was very influencing in how to write Cain. Another aspect of BDD that Vashi describes is that people "respond to attractive persons more favorably" (Vashi) thus leading to people who receive this attention and still have dissatisfaction to be diagnosed with BDD.

Vashi goes further talking about the favorable treatment beautiful people, citing how much it penetrates society. "Being beautiful has been equated with happiness, life satisfaction, higher salary, group and family formation, and social competence" (Vashi). Cain is supposed to be a direct representation of this fact. With him experiencing years of unattractiveness, he knows both sides of

society. He wants to stay on the side he's currently on to not fall victim to the jaws of society like he once was. Being beautiful now gives him the ability to support himself fully financially.

These aspects that Vashi describes influenced Cain's route, creating a character that is overly obsessed with this perfection and hate for false impurities. The Janitor's role in helping Cain involves an empathetic approach. We did not want to throw the role of psychiatrist onto the player, after all, people should seek professional help when dealing with serious mental health problems. The most the Janitor can do and should do is help Cain recognize the issues at hand and start Cain's journey of addressing them himself.

3.6.3 Additional characters

Cleanland is populated with many more characters. Our initial list of non-main character NPCs well exceeded forty. To manage scope, we focused our attention on developing characters who were featured in character routes and the main story.

Character	Cleaning Product	Background	Description
Ty "Wave" D'Pod	Tide Pods	Haitian He/Him	Older brother of Lysa and engaged to Olive. He is cool. He is fashion. He is <i>Wave</i> . He runs the shop Drip By Wave in the market.
Olive	Olive Branch	Haitian She/Her	Olive is a friendly and glamorous essential oils shop owner. She's into wellness, taking care of plants, reading, and is obsessed with cats—or in our case, Vroombas.

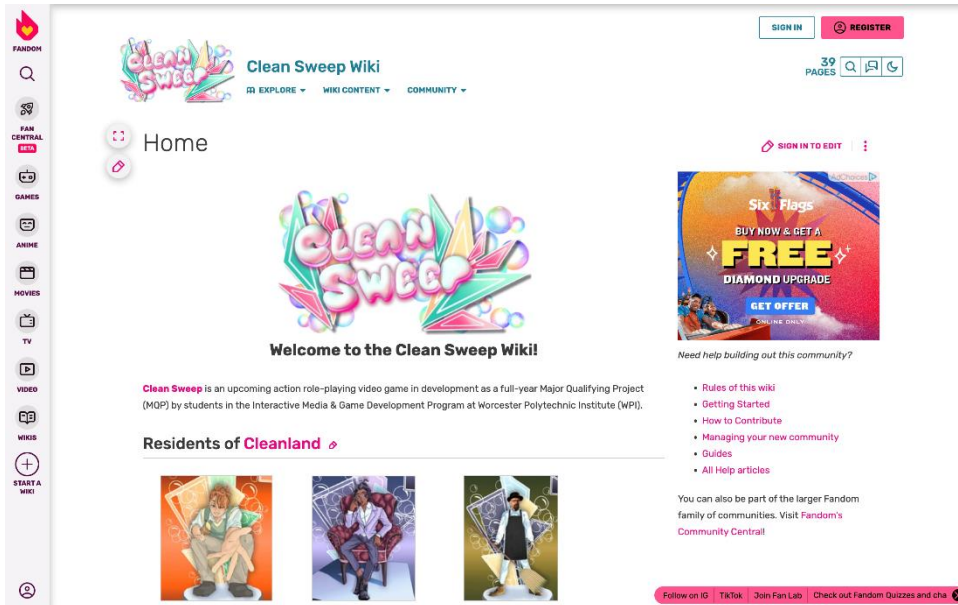
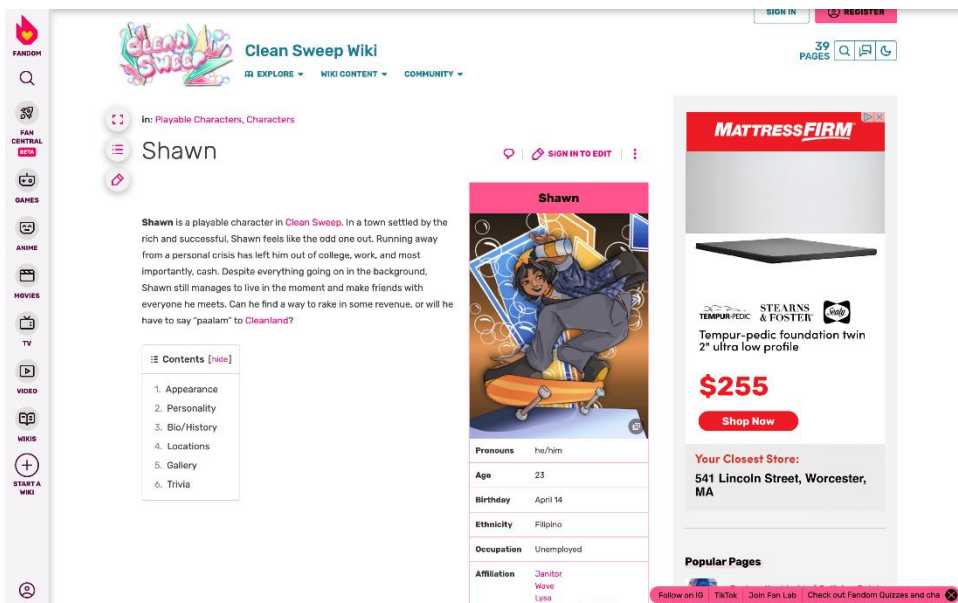
SC Jenkins	S.C. Johnson	White He/Him	SC Jenkins is the jolly mayor of Cleanland who knows the ins and outs of the town. SC is Windy's father, Roxi's uncle, and is very close friend with Mr. Tidy.
Clara	Clorox	Egyptian She/Her	Roxi's questionable confidant, number one source of bad advice, and overall toxic best friend.
Jaxy	Ajax	White He/Him	Sous chef at The Periodic Table. Inspired by the character of Jesse from <i>Breaking Bad</i> .
Scottie	Scotts paper towels	White He/Him	Local teen emo who works at Scrub-a-Licious Boba.
Amber "Brandy" Barr	Ambar	Dominican She/They	The lead singer in Cleanland's underground local punk band who doesn't let her disability stand in her way of rocking out. Works at the Squeaki Tiki, and is super supportive of upcoming talent.
Glaive	Glade	Japanese She/Her	The lead guitarist in Cleanland's underground local punk band. Believes in spirits, ghosts, ghouls, and other paranormal oddities. She has a crush on Lysa.
Draco	Drano	None	The bassist in Cleanland's underground local punk band. Collect stickers, cool rocks, pins, and other oddities.
O'Cassia "The Count"	O'Cedar	Brazilian He/They	The drummer in Cleanland's underground local punk band. Laid back and groovy, but socially reserved. The band jokes that he resembles Dracula, and since he keeps the time in the band, call him "The Count".

Figure 3.4: Table of non-playable characters (NPCs) featured in the main story and character routes.

In accordance with our commitment to diversity, the additional characters in *Clean Sweep* represent a broad spectrum of ethnicities and body types. By integrating this diversity into both the narrative and character design, we challenge the often stereotypical and superficial portrayals prevalent in mainstream media. Our goal is to create a more inclusive experience that enhances player engagement by reflecting the diversity of our real-world audience. Through considerate character development and interactions, we aim to enrich our narrative and create a more inclusive and relatable gaming environment.

3.7 *Clean Sweep* Wiki

The *Clean Sweep* Wiki is hosted on *Fandom*, a platform that allows fans to create and manage wikis for their favorite media. The wiki provides in-depth information about the characters, locations, and main story. This online repository helps players better understand the backstory and attributes of their favorite characters and explore the different settings they encounter in Cleanland. It was created not only to provide additional context and lore but to promote community engagement.

Figure 3.5: *Clean Sweep* Wiki home pageFigure 3.6: *Clean Sweep* Wiki page for the character Shawn.

3.8 Recommendations

If you are planning to make an ambitious, story-driven game, you should have that reflect in the priorities of your team members' disciplines. Do not put the team members who are most involved in other disciplines on the MQP as two thirds of the writers.

Have different types of writing meetings, such as ones for discussing goals and making decisions and ones dedicated to actual writing. Group writing sessions have several benefits. Because dialogue itself is a back and forth, writing with others will result in more juiced-up, organic writing. It also provides a casual space for feedback and revisions live as you write. Share ideas, research, and writing early and often and don't get married to an idea.

Feedback from focus groups was very useful on how to integrate representation of diverse groups that members of our team are not a part of. Having focus group participants from our target audience gave us additional insight into what made individual characters appealing, providing guidance on how to further development for relatability and engagement. Focus groups can also be useful for other aspects of a game, not just narrative and art.

Research can be useful when developing a character. Learning about specific aspects of a character's personality can reveal new characteristics that support and expand what you have. Another helpful method for character development is finding their voice. This means determining their tone, speech patterns, and voice comparable (see Appendix G). Doing this will force reflection on the character and keep dialogue consistent.

When writing with a team, agree on conventions to ensure unified writing. This will prevent instances where writers use different variations of the word "yeah," or having sections of the narrative vary in their use an Oxford comma. Effective communication about writing structure overall is essential for maintaining coherence throughout the narrative.

Try to write what you are passionate about. This not only makes the writing process easier and more enjoyable but also fuels motivation. Giving writers room for creative freedom is a way to achieve this. While not every production will allow for this, finding ways to prioritize passion in your work will help to alleviate the challenges of demanding writing tasks.

4 Game Design

4.1 Design Within the Team

Within the structure of the team, designers were responsible for the game's shape and mechanics, planning out large concepts like game loops and character stat progression, as well as smaller tasks such as deciding input methods and health indicators. There was design work for both the story and combat sections of the game, such as devising surrounding systems to enhance the story and designing and iterating on the game's combat systems. Design also acted as a space for ideas in their early stages across all departments, brainstorming SFX, VFX, and early UI mockups while ensuring ideas were feasible for implementation. We held numerous meetings over the course of the year—both internally and with other departments—to make sure the vision for the game remained cohesive and practicable.

4.2 Blending Genres

The *Clean Sweep* team decided early on to try our hand at blending genres, mainly because we didn't feel that any single existing genre interacted with our characters in the goofy yet personal way we were looking for. Blending genres was a big decision for our game: When developing with a single, established genre, you save tremendous time designing by inheriting the established conventions and design solutions of existing games. However, when blending genres, many of these solutions must be rethought or thrown out entirely, and lots of time must be spent solving new problems that arise at the genre's intersections. Despite this, we believed that the

additional time to create a unique blend of game mechanics would pay off. We felt a blend of genres would help us create a unique game loop that would be a better fit to our unique premise and tone. That said, we still took plenty of inspiration from other games and genres to see what worked for them and get a better feel for what we were looking for.

4.3 Genres and Inspirations

We examined several genres when looking for inspiration for our game, starting with character focused, turn-based RPGs like *Persona 5* and *Fire Emblem*. These games do an excellent job of fostering player attachment to characters by supplementing in-game relationships with game mechanics, and their social simulation aspects were a great fit for interacting with our games many characters. However, we felt their turn-based sections were too slow for the upbeat, goofy tone we were aiming for. To address this, we first looked at more interactive turn-based games like *Super Mario RPG* and *Bowser's Inside Story*, which better matched our tone. These games involve the player more constantly with minigame-like, action command sections to enhance the action happening on screen. Eventually, we decided that turn-based RPGs lacked a level of personal interaction, so we turned to exploring action games. We started with character action games like *Devil May Cry* and *Transformers: Devastation*. Their fast pace, over-the-top action, and exaggerated moves were exactly what we were looking for to capture the goofiness of our game. However, their focus on power fantasy rather than relatable characters meant that a slight pivot within the action genre was necessary. We finally landed on action-RPGs. Though the line between the two can be somewhat blurry, action RPGs possess the same high-octane action of their character action siblings, but with a greater emphasis on

storytelling and nuanced character development. These examples of social simulation and action RPG games would serve as guidelines for the vision of our game and stand as our chosen references for the core pillars of each genre that make them enjoyable.

4.4 Mapping Out the Game

Our first step in solidifying our game's vision was to construct a game concept map. These maps are created with the goal of visually exploring a game's key elements and identifying strengths that need to be focused on during development to make the most effective end product. We required multiple iterations of our map to nail down which aspects of our game were most important and to better categorize elements. Our original map successfully brainstormed the initial elements of our game but lacked organization or a hierarchy of importance. Without breaking down the categories further it was difficult to tell which sub elements were most important to the identity of a section, and what details made them significant. With a multi-genre game, it was also difficult to determine how each category was distinct within the game loop. For our second iteration, we remedied these issues by reorganizing and further breaking down categories, ensuring that each element shared connections only with its defining aspects. We also changed the size of each node to visually distinguish between their importance, with larger nodes being more important and thus placed higher in development priority. Using a game concept map gave us a greater insight into our game's defining elements and allowed us to move forward with a clearer picture in mind.

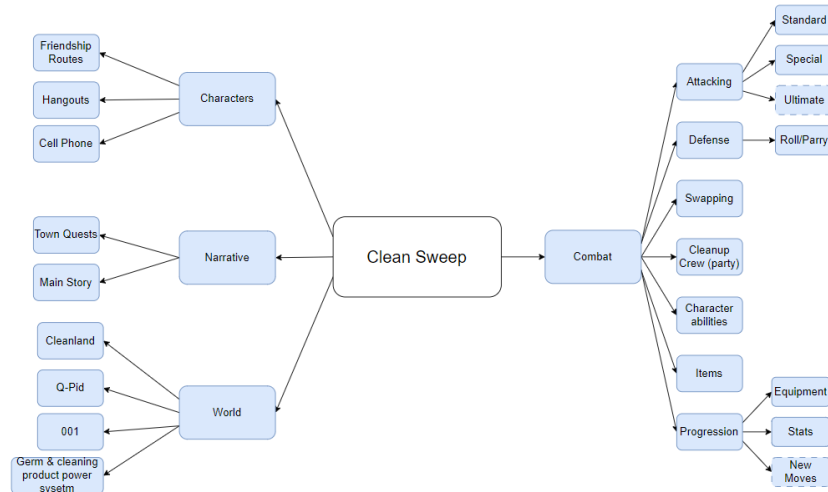


Figure 4.1: First iteration of game concept map

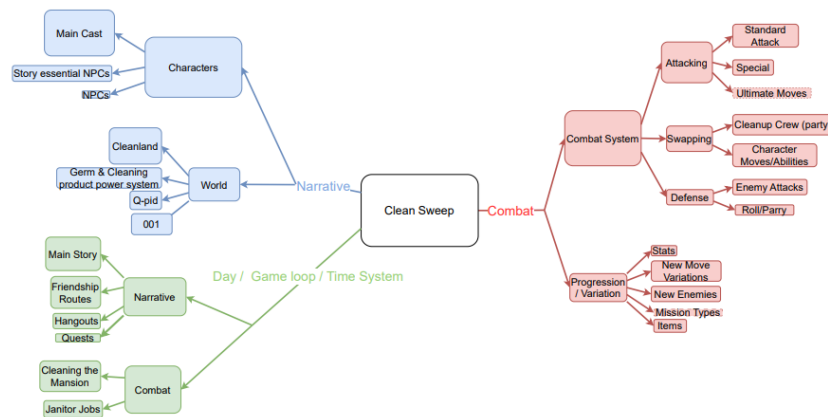


Figure 4.2: Second iteration of game concept map

4.5 Integrating Narrative and Combat

Once we decided our core elements, we needed to integrate them into a cohesive game loop. Both the story and combat sections required surrounding game mechanics to incentivize completion and provide a feeling of progress throughout the game. Organizing elements into repeating structures also helps to define a system for implementation and create the ability to

work modularly. Below, we detailed how each section of our game was organized or enhanced with these mechanics.

4.5.1 Character Routes

Character routes are where each character's individual story unfolds. Within the world of clean sweep, stronger relationships translate to greater cleaning combat abilities, which is justified to the player in that stronger relationships lead to better trust and constructive collaboration when working together. companionship and support are also the opposite of the negativity that germs generate. We produced the system of Friendship Experience Points (FXP) to symbolize this relationship and mechanically tie the strength of the player's friendship with a character to the character's combat strength. With this system, character routes set the pace of game progression by dictating the strength of combat missions you can take on. Route events are also more naturally spaced out, as each event is locked behind a certain FXP threshold. Additionally, certain route events include combat encounters to leverage the combat system, break up long sections of dialogue, and make the game's genres feel more interconnected.

4.5.2 Hangouts

More casual and lighthearted conversation takes the form of hangouts. Each day a random number of character hangouts are offered to the player where they can choose to spend time with specific characters. These events are designed to provide a casual glimpse into a character's personality. During these events, the player might join a character in one of their favorite activities or discuss a subject they are passionate about, strengthening their friendships through shared experiences. At the end of the hangout, the player and participating characters

earn FXP because of their friendships growing from time spent together. This was added as an option of leveling to encourage players to grow closer to characters and to break up the monotony sometimes present in traditional RPG leveling.

4.5.3 Quests

Quests encapsulate the activities and current happenings of Cleanland and are designed to foster player attachment to the town and its characters. Quests encompass the large variety of tasks you need to perform as the town's janitor. Each quest begins with a conversational exchange with a character where they detail their predicament and request help. It's then the player's job to complete the task, often requiring them to walk around town and gather information from other characters. Upon completion, the player may gain FXP or items to reward from the quest giver to incentivize further questing. Though quests may appear urgent, they are functionally asynchronous and serve simply to motivate the player to explore the town or interact with other characters.

4.5.4 Mansion Storyline

The mansion blends combat and story sections of the game, serving as the central location for both. Rooms are grouped to form levels to organize combat and space out main story segments. Access to these levels is controlled by difficulty, meaning players will need to level up their characters through route progression to continue the main story. Parts of dialogue are placed at the beginning of each combat section to progress the story and give context to fights. Additionally, pieces of environmental storytelling are discoverable after combat is

complete, incentivizing players to explore the newly cleaned room and adding a sense of mystery and discovery.

4.5.5 Janitor Jobs

The main way to level up through combat is janitor jobs. Janitor jobs are combat missions offered every day at the town hall, for which you assemble a party of characters to take on a germ infestation. If the encounter is successfully completed, all participating characters earn FXP because of growing party coordination. Jobs are offered in easy, medium, and hard levels of difficulty, with harder jobs resulting in greater rewards. This separation allows less experienced players to progress while still offering a challenge to those ready for it. We also plan to implement different types of jobs and subquests for each job to keep missions from becoming stale.

4.6 Phone UI

Many of the game's systems, particularly those focused on friendship like FXP and hangouts, require a centralized display to conveniently present information to the player. We discovered that a smartphone serves as an ideal real-world parallel for this purpose. Since phones are commonly used for maintaining contacts and staying updated with friends, their familiar user interface (UI) instantly makes its function clear to players. The fact that the UI is grounded in the world also offers the added benefit of enhancing player immersion.

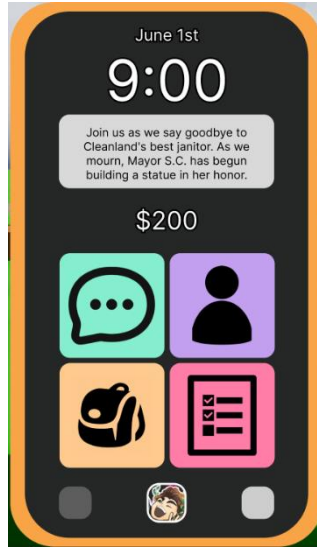


Figure 4.3: early mockup of phone UI with messages, contacts, items, and to-do list

4.7 Combat

4.7.1 *Combat Vision*

4.7.1.1 Pacing

The combat sections of our game are where the action happens. We felt that the best fit to our upbeat and goofy tone was a fast paced, action-packed combat system. This includes fast player movement, quick anticipation and recovery phases on attacks, and a dynamic interaction system that encourages seamless transitions between different characters and their abilities. Keeping the combat fast makes sure players are forced to think on their feet and utilize the options they are given in different ways depending on the scenario.

4.7.1.2 Combat Style

There are several approaches of combat style that melee combat games can take, but we found the most entertaining to be that of emergent space puzzles between the player and enemy positions. In this system, players must quickly determine which characters and abilities were the right fit for a given scenario. Each move has unique properties and covers a unique portion of space, so swapping between characters to access the right move or manipulating enemies into favorable positions is essential to get the most millage out of your moves.

4.7.1.3 Combat Tone

To homogenize the tone combat with that of the game's writing, we made sure to infuse character combat designs with a powerful sense of combat with that of the game's goofiness and personality. Characters utilize eccentric, over-the-top weapons, and their moves are brimming with visual humor. This approach incorporates the personalities of the characters developed in dialogue to add flavor to the combat experience.

4.7.2 Controls and Game Feel

4.7.2.1 Input Device

Our game was initially developed using mouse and keyboard controls because of the many inputs it offers and simply because that was the option readily available to us during development. However, we eventually found, that the binary axis input of WASD controls felt cumbersome for fine combat movement and noticed that many games with a similar style of combat preferred controller for this reason. Controllers also made basic input options available

to the player more readily apparent, and surveying our audience found that many of them preferred controller for this genre.

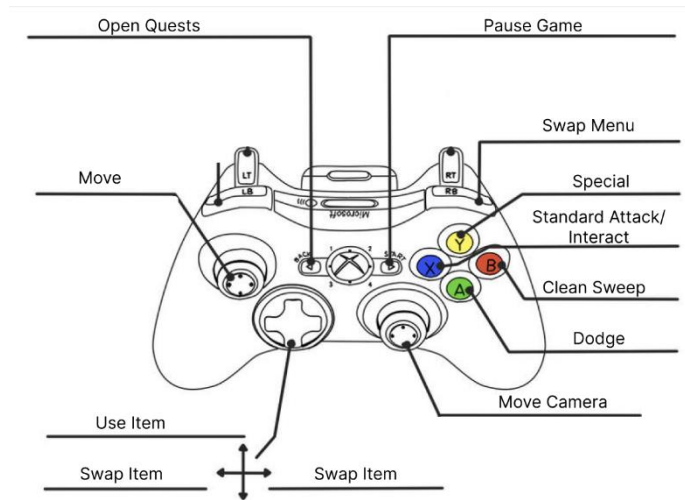


Figure 4.4: Inputs for the game using controller

4.7.2.2 Camera

For the game's camera, we opted for a medium zoom to give the player a wide enough view of the battlefield while still maintaining a more intimate perspective, keeping a close relationship between the player and the character. This balanced approach ensures that players can effectively manage both their immediate surroundings and anticipate incoming threats from a distance, without losing the detail needed to precisely land moves. The camera is also rotatable to allow players to get the best angle to judge distances and see threats from all angles.

4.7.2.3 Attack Snapping

One thing we noticed once we had gotten the basic combat system established was that it was sometimes difficult to turn the player and land moves exactly, either because of faults in

our judgment of space or because of the fast inputs sometimes required due to the pace of the game. These discrepancies between the player's expected outcome and the actual result would make gameplay feel clunky and result in far more punishing gameplay than we were intending. To remedy this, we instituted an attack snapping system where if the player's attack input direction was close enough to that of an enemy, the attack would snap to the intended target. This had to be carefully tuned so as not to be overdone, since too much snapping can remove a lot of challenge from the game and make it feel like it is playing itself. When tuned correctly, however, it does a lot to make fast paced gameplay more enjoyable and in line with the players' expectations without being overly intrusive.

4.7.3 Swapping

From the initial stages of our game's development, we established character swapping as a fundamental aspect of combat. With this mechanic, you can use the swap button to choose from one of four players to play as on the battlefield at any given time. This feature forces players to continuously evaluate which character to deploy, considering the balance of attributes and resources among their roster. This strategic layer not only enriches the combat experience but also significantly elevates the overall gameplay, adding depth and complexity.

4.7.3.1 Swapping Menu

The one obstacle of migrating to controller was the number of available buttons, since we had the unique requirement of swapping between 4 different characters in addition to the normal controls of a melee combat game. We eventually solved this by implementing a hybrid button radial menu. The action of swapping needed to be done quick, so using buttons rather

than an analog stick sped things up efficiently for it to not get in the way of the players immersion. Time is also briefly slowed down when pulling up the menu to give the player time to assess which character they would like to swap too, should they need it.



Figure 4.5: Prototype for swapping button radial menu

4.7.4 Attack Options and Moveset Design

4.7.4.1 Standard Attack

The standard attack is the most commonly used combo option for players since it is always at their disposal and typically features the quickest startup and recovery times. Standard attacks are primarily effective for closing in on and engaging a single germ. It includes a sequence of 3 or 4 attacks that proceed with each press of the player's button. The final blow in this sequence is usually more powerful or has greater reach, motivating players to execute the full combo. Since standard attacks target a single enemy, their weakness is that they leave the player vulnerable to attacks from other angles. Additionally, after finishing the attack sequence,

there is also a short window when the player is left vulnerable, so the standard cannot be spammed without thinking.

4.7.4.2 Hold Standard Attack

Hold standard attacks, which are activated by a longer press on the standard attack button, represent a more dedicated choice. Initiation of these attacks requires players to hold down the attack button, charging the move before it can be executed. We opted for a hold mechanism instead of an additional button press to streamline the control scheme and to emulate the sensation of charging the attack through sustained physical input. Due to their charging period, these attacks demand a greater commitment compared to quick standard attacks, necessitating that players carefully assess whether they have sufficient time and space to execute them successfully.

4.7.4.3 Special Attack & Hold Special

Special attacks and hold specials are the most unique attacks of each character and can fill a wide variety of roles within their moveset. These attacks are limited to a single use before requiring a cooldown period to recharge, prompting players to carefully consider when the right time is to use them. The need to wait for these attacks to become available again not only adds a layer of strategy but also encourages players to alternate between characters based on the availability of their special moves. Generally, these attacks are meant to excel in a specific situation and serve to define the archetype of a character. The difference between the special and hold special attacks is the same as that of the standard and hold standard attacks, with the hold special requiring you hold down the special attack button to charge and execute it.

4.7.4.4 Character Passives

In addition to attacks, each character has a passive effect that heavily shapes their gameplay. Passives also heavily vary depending on the character. Their mechanics generally push the player to play the character more in line with their archetype, add some extra layer to the character's playstyle. Passives are also designed to complement the game's swapping mechanic, frequently acting as key factors in determining the most strategic moments for swapping characters and prompting players to engage with the mechanic on a deeper level.

4.7.5 Character Movesets

4.7.5.1 The Janitor

Since the Janitor is the first character the player experiences, and is an automatic part of every mission, the goal of their design was a basic and well-rounded moveset. This allows them to appeal to a wide variety of players and function in any party composition. Their moveset features a variety of options: traversing distance with their standard attack, throwing a ranged projectile with their hold standard, elements of crowd control and combo assistance with their special, and space clearing crowd control with their hold special.

Since the player is forced to include the Janitor, they are given the unique ability to change their passive depending on which janitor suit they are wearing. This allows the player to adjust the Janitor's playstyle to better match what they find most enjoyable or effective.

4.7.5.2 Charming

Charming is meant to be a slower brawler-type character specializing in close-quarters combat. Most of her moves were designed for use in melee range to give her distinct strengths and weaknesses to encourage swapping. Her moveset features high damage options and mobility tools to help her stay in melee range. Her passive also helps her stay in the fight for longer, as she gains temporary health for each attack she lands, promoting more aggressive play in line with her design. Swapping with temporary health also heals the swapping character a small amount, giving players an option to cash out if they are left outside of melee range without movement options and encouraging character swapping.

4.7.5.3 Soap

Soap is the most technical character in the roster since their moves are linked to an extra layer of resource management to be used most effectively. Their passive has them manage a slipperiness value, with them becoming more slippery the more attacks they land. A certain amount of slipperiness is good, as it helps you to traverse the battlefield faster and keep up with enemies, but too much will become unmanageable, and the player will most likely have to swap off to reset it. Their moveset plays off this, as most of their moves require that you are in the correct position to be most effective.

4.7.5.4 Windy

Windy excels in the role of crowd control, meaning they are most effective when dealing damage to many weaker targets rather than a strong single target. Most of their attacks see them damaging crowds at mid-range, while their special attacks deal damage to groups at range

with a large projectile. Their passiveness allows them to mark multiple enemies with their attacks. All marked enemies are dealt damage when Windy swaps out, incentivizing swapping and solidifying their role as a crowd control character.

4.7.5.5 Lysa

Lysa is the archetypal ranged character of the roster, attacking at a distance using note beams from her keytar. Her passive sees these note attacks string together to form a song while she is uninterrupted, providing a proportional buff to a party member depending on how long the song chain was when she swaps out. This creates a risk-reward for the player, where longer songs are more powerful but also have a greater chance of being interrupted, encouraging swapping depending on the danger of the situation.

4.7.5.6 Roxi

While Windy primarily derives crowd control benefits indirectly through their passive abilities and from a distance, Roxi excels in direct, close-range crowd control. Her primary method of crowd control comes from her attacks, utilizing the broad, sweeping hitboxes of her pressure-washer weapon to impact multiple enemies simultaneously. Her passive plays into the strengths of her hitboxes by soaking enemies she hits enough, causing them to move slower and take more damage.

4.7.5.7 Cain

Cain is the slowest and heaviest hitting character, delivering substantial damage with each swing of his washing machine hammer. Due to the slow nature of his attacks, players must

carefully consider each move, ensuring they have a sufficiently large opening to strike without being punished. To enhance the complexity of his gameplay and diversify his style, Cain's passive ability causes his weapon to heat up with each successful hit, gradually increasing his attack speed and damage until the weapon overheats. His special abilities enable him to release this accumulated heat for additional damage, introducing a strategic element of resource management to his slower combat style.

4.7.5.8 Shawn

The primary focus of Shawn's design was on mobility. He utilizes his skateboard to swiftly traverse the battlefield, seamlessly integrating attacks from his mid-range spray-bottle weapons whenever an opportunity arises. His passive supports this dynamic playstyle by incorporating a combo system, where his damage increases the longer, he goes without a large gap between his attacks. This system encourages continuous, aggressive engagement during combat, requiring full utilization of his movement abilities.

4.7.5.9 Vic

Vic's moveset is still in the early stages of development, but his role has always been focused on delivering high damage. Armed with a double-sided mop Bo staff and mop shoes, Vic performs flashy, over-the-top movements reminiscent of his K-pop performances to swiftly unleash substantial damage. An early concept for his passive was to grant him a higher base damage, but decrease his damage output each time he sustained a hit until the player switched characters. This ability would enhance the risk-reward aspect of combat and establish Vic as a true glass cannon character.

4.7.5.10 Mr. Tidy

Tidy's moveset currently stands as one of the least developed among the roster, mostly due to its lower priority within the project. In the early stages of conceptualization, he was envisioned to assume a unique support role, specializing in bolstering other party members during combat. His unique ability involved concocting high-class dishes on the fly, which he would then serve to teammates upon swapping characters. As for his offensive capabilities, Tidy was designed to utilize old chemistry experiments as improvised weapons, hurling them at enemies to unleash deadly effects upon impact.

4.7.6 Items

Items function to spice up combat and create an extra layer of strategy by serving as the player's primary source of healing and temporary effects. Each item has a finite number of uses, encouraging players to think carefully about when to use them. Consuming an item activates a cooldown window before an item can be used again, encouraging players to plan their item usage in advance.

To diversify the functionality of items and add variety, we introduced three distinct categories: food, drink, and essential oils. Food and Drinks always carry some amount of healing, the difference being food affects a single target while drinks affect the whole party. Food is simple to manage and good for when a single character is low. Drinks are harder to take advantage of but offer greater rewards if the player has swapped consistently enough to spread their damage across the whole party. Essential oils do not heal like the other item types, but instead provide strong positive effects for a single character. The main goal of essential oils is to encourage the player to temporarily change their playstyle to best take advantage of their effects, providing a change of pace.

To prevent items from being overused or oversaturating combat, we limited the number of items players can bring into combat with item slots. Players can only bring as many items into combat as they have slots, encouraging them to think carefully about which items they will need and pushing them to create a diverse balance of effects.

4.7.7 Enemies

Enemies are the primary obstacle players must overcome in combat and are the main contributing factor to the game's difficulty. They serve to test the players' knowledge of game mechanics, challenging them to apply what they have learned in increasingly complex scenarios. Enemies serve the significant role of adding variety to the game. They do this in two significant ways: through diversity of enemy types and through their arrangements in the level.

There are two types of germ enemies currently in our game: melee and ranged germs. Melee germs aggressively engage the player by closing in with various short-range attacks. They either execute a leaping jump or run up to the player and deliver one of three close-range strikes. Ranged germs will stay back and fire spit projectiles from a distance, incentivizing the player to come to them rather than engaging themselves.

These two enemy types together create interesting emergent behavior where the melee germs shield the ranged germs. By introducing enemies in different environments and arrangements, enemies can force players to adapt and produce new strategies, keeping the gameplay fresh and interesting. Players often must decide whether to push through the melee germs to reach the ranged attackers, or deal with the initial melee germ threat first while dodging ranged attacks.

4.8 Recommendations

Before deciding to blend genres, it is crucial to carefully evaluate your intended scope since this decision can significantly expand your design workload. Use core pillars and a game design mind map to create a better understanding of your game early on. Collaborate closely with other teams to ensure that your design unifies and leverages the strengths of every aspect of the project. In terms of combat, ensure that each player attack option is distinct and balanced with clear strengths and weaknesses. Additionally, do not overlook the design of enemies, as they are essential for creating engaging and challenging combat scenarios.

5 Art

5.1 Art Style

With one of our pillars being "goofy" and having such a silly premise, the art style needed to be punchy, energetic, and most of all, appealing. The overall color palette for the UI was developed using bright colors with a blend of geometric shapes and bubbles to mix ideas of cleaning and a dynamic, semi-retro vibe. Character sprites were developed to provide a closer look at the character's facial expressions and exaggerate them for comedic effect. Akin to our comparable games, the character designs had a strong anime influence, while also blending in realism to help communicate characters' ethnic backgrounds.

In most media forms, Caucasian characters have most of the spotlight. Stylized media tends to either erase ethnic features or exaggerate them to the point of being offensive. The anime art style is one of the biggest culprits, with a history of racist depictions of non-white characters. With *Clean Sweep*, we wanted to provide positive, accurate representation and show that it is possible to use anime elements to create diverse characters.

The art style later evolved into more semi-realism with a sprinkle of anime influence during the hi-poly character sculpts, with facial planes and eyelids becoming more defined to assist in retopology and rigging.

5.2 2D Art

While primarily being a 3D game, we still wanted to blend 2D and 3D, using 2D to support the 3D. In the game, we wanted the 2D to further build the style and use comical character expressions to propel the pillar of being goofy.

5.2.1 Concept to Model

All characters were brought from 2D concepts to 3D models through an art-team wide iterative process to make sure characters were representative of both their backgrounds and their cleaning product. Heavy importance was placed on gathering referential material for each character to make sure we entered character development with a unified vision of which background the character was going to represent and how the cleaning product's elements were incorporated into the design. Included in this reference material were different images of real people from these backgrounds. We did this to ensure that we illustrated the character's ethnicity, especially through their hair and facial features, in an inclusive and empowering way that would make someone from that background excited to identify with them and feel seen. In addition, stylistic appeal was also an important element, with both body and facial proportions being key points to illustrate through the concept art. Constructive critique sessions were held during art meetings, and overall led to improvements in the designs and style. See Appendix H for an example of the first and final stages of the pipeline.

5.2.2 UI Design

For the UI, the vision was to achieve a polished, vibrant, fun interface that also rooted itself in the cleaning theme. For dialogue, we wanted to have a dark background contrasted with lighter text for

better visual ease. We included a bubble border around the textbox to add cleaning flair, and used our thematic colors of teal, hot pink, and warm yellow.

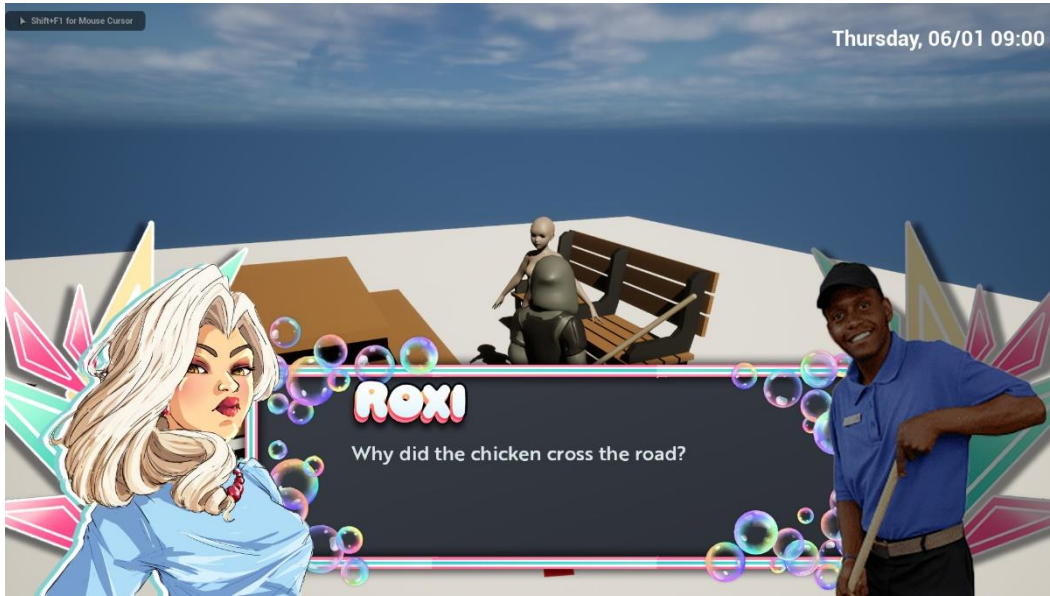


Figure 5.1: A dialogue UI first pass with a placeholder janitor sprite.

For shops and janitor job selection screens, we decided to use black and pink as our primary colors, using teal and yellow as accents where we could. The goal of these screens was to exude sleek coolness and make the player feel excited about combat, whether through buying items or prepping to enter the germ-infested fray.

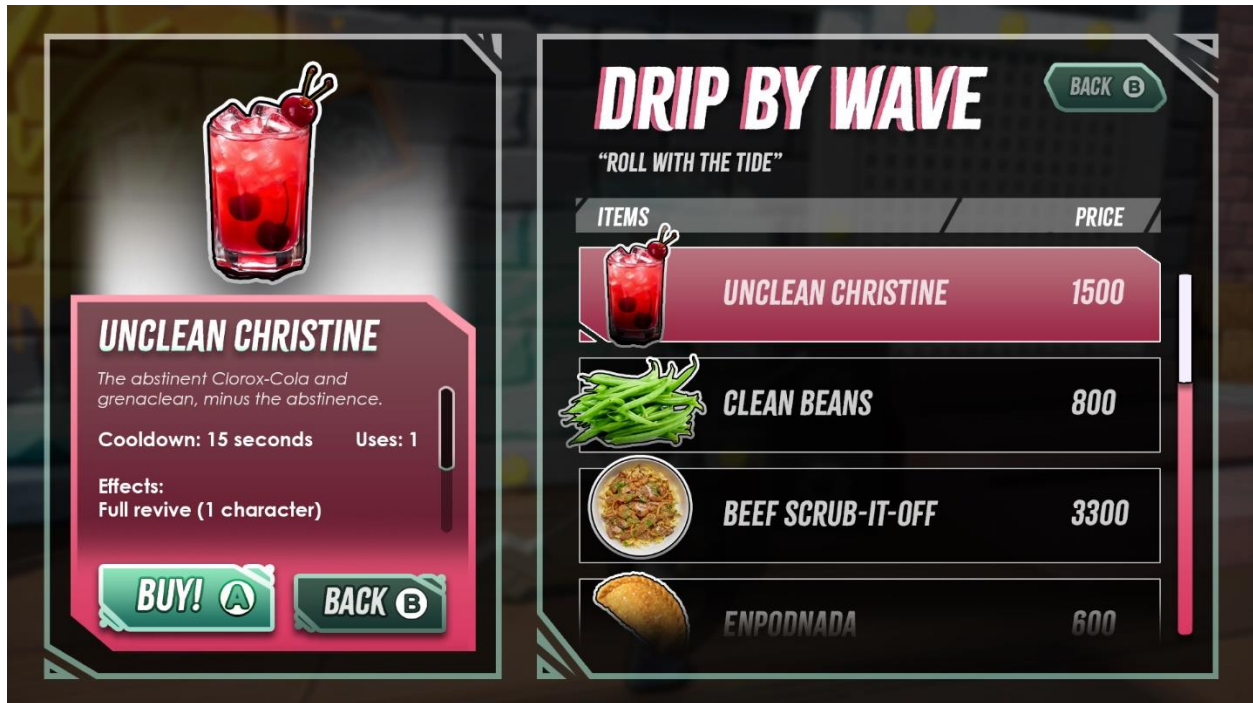


Figure 5.2: A rough mockup of an interactable item shop in the market.

Another interesting UI element was the character emotion backings. These triangular assets were used to support the character's emotive sprites through the color scheme and add a unique stylistic element to popular visual novel style dialogue. The geometric appearance adds energy and contrasts to the UI and background gameplay, grabbing the player's focus.

5.2.3 Sprites

Drafting for dialogue sprites began in late C Term to prep for the PAX build to show the full functionality of dialogue. The art style had drifted from heavy anime influence to a light anime influence with a base in realism. This way, character's features could be portrayed and articulated more accurately to their ethnic background. The new style was different from the 2D artist's usual

style, and it became clear that trying to draw all the sprites from scratch would be an inefficient use of time, especially with the deadline for the build rapidly approaching.

Instead, they chose to sculpt the desired expressions onto the character models, screenshot them in ZBrush from the same angle, and then use them as a base for the sprite sketch. That way, the sprite facial expressions could be easily modified while maintaining cohesion with the 3D art style. For shading, Persona 5 was the main comparison, using their color composition and limited usage of shading as a reference for a polished 2D portrait. Using the lasso tool to block out areas for rapid shading as well as clipping layers, the sprites were able to be completed quickly. The color scheme is where the art lead thought the sprites should differ from Persona; Persona uses a lot of grey for shadows, which did not compliment the colorful vibe of Clean Sweep. Instead, they opted for a more, but not overly, vibrant palette that brought out characters skin tones and hair details, including simple hard lights and shadows. As for character emotions, four for each character were developed: neutral, positive, stressed, and a comedic sprite unique to each character (see Appendix I). The final published build will feature a lot more emotions, but we decided to keep it scoped down to prioritize other UI and various game elements for our Showfest vertical slice.



Figure 5.3: A side by side of Charming's sprite reference screenshot and her first pass neutral sprite.

The item sprite pipeline followed similar steps, sans preparatory modeling. Instead, references of the popular foods we were trying to include were found online and were referenced side by side during the sprite development. Coloring still placed a large emphasis on hard, vibrant lights and shadows. To further stylize the sprites and make sure that they stand out on any backing surface, a black outline and secondary white outline were used.

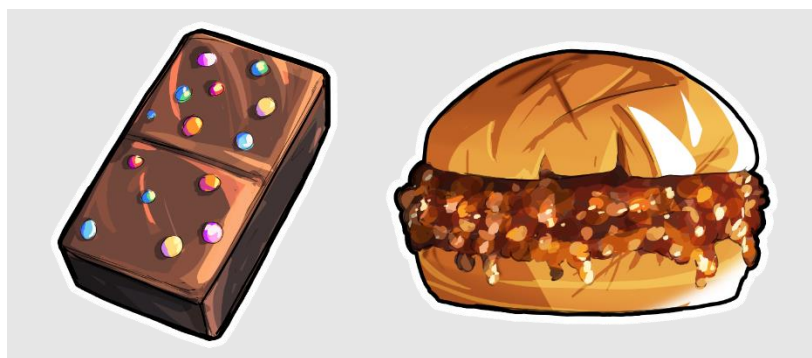


Figure 5.4: Sprites for the Cosmic Broomie (left) and the Moppy Joe (right).

5.3 Characters

5.3.1 Humans

Human characters were each designed after a cleaning product. With our initial idea being a dating simulator, the main playable cast was modeled to be attractive to make sure players would want to explore their routes. Though the idea was later scrapped to place a stronger emphasis on friendships and growth rather than romance. To avoid copywrite, the characters' designs do not blatantly reference their inspirations, and instead utilize the products color scheme and some small motifs from the branding. Playable and non-playable characters have different ethnic backgrounds, sexualities, and body types to provide positive representation to a variety of underrepresented groups. The following are some notable examples of intentional character design choices.

5.3.1.1 Olive

Olive is a plus sized black woman, a group consistently misrepresented in media and a common victim of the Euro-centric beauty standard of perceived "Whiteness". In the PMC article "Beauty and Body Image Concerns Among African American College Women," it is stated that "features more akin to the African esthetic are deemed ugly, undesirable, and less feminine," which is "transmitted daily from multiple external forces and social institutions (church, government, business industries, media, and family/peer groups)." Olive's original chill-inspired outfit with subtle glamor, while not intentionally doing so, did not actively work against these stereotypes as effectively as we would have liked. Instead, we opted to change Olive to become

one of, if not the most well-dressed, feminine, and attractive character in *Clean Sweep*. Instead of focusing on the comfort and wellness-centric portion of her character, the character designer pivoted the design to match her interest in plants and her floral, organic storefront aesthetic.

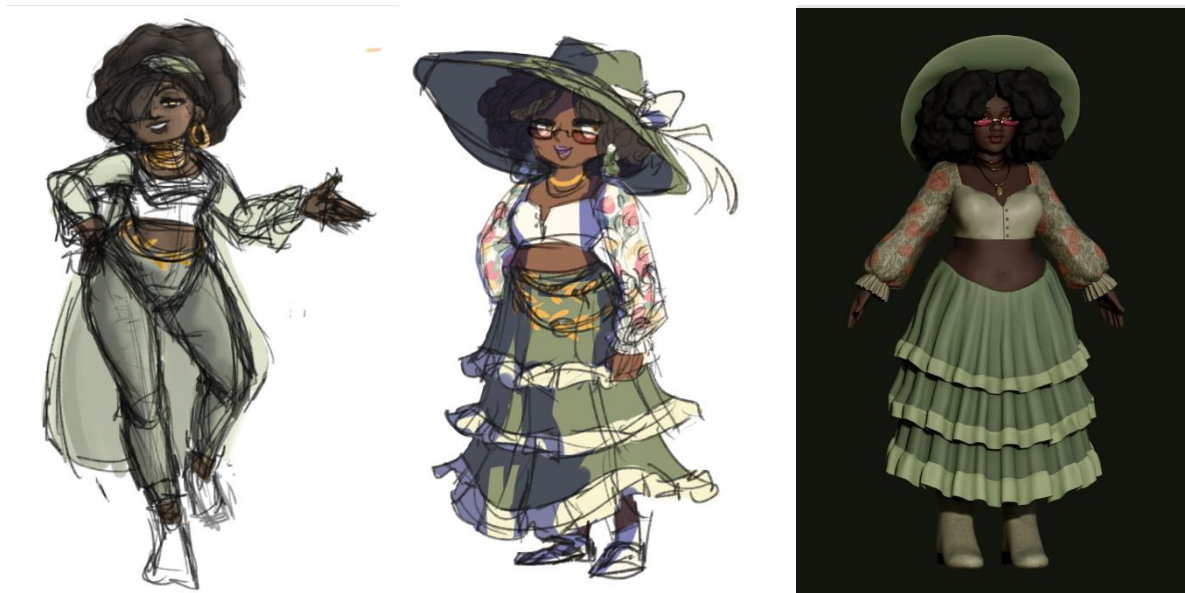


Figure 5.5: Olive's first design, her finalized design, and her model.

5.3.1.2 The Janitor

In an interview about Spider-Man's design, Marvel's Stan Lee stated, "The best thing about Spider-Man's design... [is that] he is completely covered," and "any kid could imagine he's Spider-Man because no color of the skin shows," though he admitted it was "accidental". This was the main inspiration for the Janitor's design and character. They have no gender, racial, or other implication towards identity, allowing for players of every background to fully immerse themselves as the hero. The only pitfall was, because the character is modeled and must take a defined physical form, we had to give them a body type.

5.3.2 Germs

For our germ characters, we wanted to lean into our goofy pillar with comical designs and movement. Germ enemies look very silly to make combat more slapstick and high-energy. In the current build, three germs are present: melee, ranged, and exploding. The melee germ represents the standard booger. They have one yellow eye, a mouth, and giant arms to swipe at the heroes. When VFX are added, they will drip goop constantly. The ranged germ was designed to resemble the pinnacle of grossness of adolescence, with acne, chapped lips, and random hairs. They, in professional terms, “hock a loogie” to attack. In future development, if the ranged germ is left alive for too long, it will enter a second state, called “Glow Up”. The Glow Up state of the ranged germ was envisioned to emphasize the beautiful metamorphosis of the once grotesque germ, giving it luscious lips and chiseled cheekbones... which apparently means higher defense and damage output. The exploding germ was modeled after the standard idea of a bacteriophage virus, consisting of a thin eye, an elongated octahedron body, and 4 legs. The exploding germ is the fastest of the germs, and will sprint, jump, and plant itself into the ground near the player and blink until explosion. The player can also hit the ranged germ before it explodes to send it flying at other enemies.

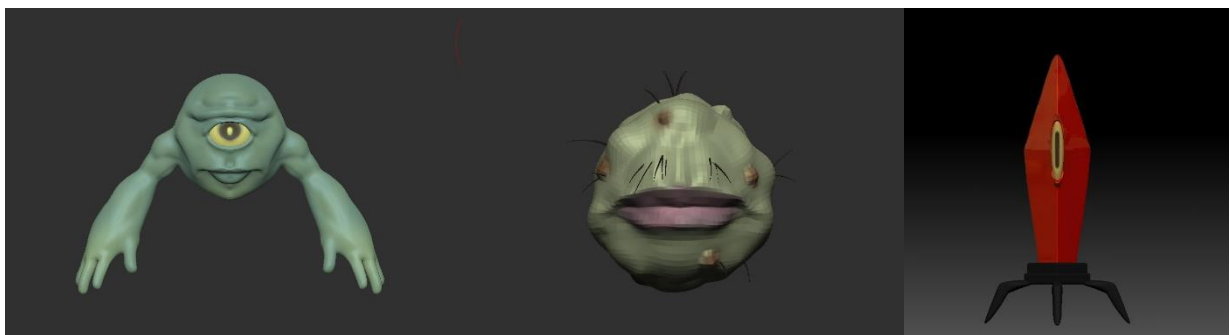


Figure 5.6: From left to right, the Melee, Ranged, and Exploding Germs.

Our boss characters 001, Typhoid Mary, The Poop Monster, and Cerberlyssus, are exceptions, as they take human (or three headed dog) forms to help convey their inspirations and what/who they are meant to represent. The boss germs become closer to their namesake and denser depending on how strong they are. The boss characters are not implemented or set in stone now, as detailed in the writing section, but will be further developed moving into summer production.

5.4.2 The Social Implications of Jiggle Physics

A popular animation technique in visual media, jiggle physics, was a point of discussion amongst the art team and its advisor. The team was unified and passionate about their shared opinion, and decided to omit jiggle physics on human characters, not because of the fear of an increased workload, but out of respect. We recognize and acknowledge the importance of providing better representation for different body types in media, as an argument can be made that jiggle physics created through a lens of respect can help highlight realistic physics on human bodies. However, we did not want to capitalize off the sexualization of underrepresented groups in any capacity, especially considering the perpetuated fetishization of marginalized groups. Thus, though we could have achieved it, we decided against jiggle physics for humans. As for the germs, in future development, we are playing around with different ideas, and plan to utilize them to make the germs' movement and combat sillier.

5.4 Environment

5.4.1 Overall Style

While making the environment art, cohesion with the game's overall aura and mood, as well as with the characters, was paramount. To achieve this, a similar bright, colorful, and cartoonish style was adopted. Most props were created using a semi-realistic approach, with some exaggerated proportions and colors. A common theme that can be seen across all the environmental textures is the Kuwahara filter. This is a nonlinear smoothing effect that preserves hard edges instead of blurring them, making textures visually appealing and giving a hand painted effect. In other words, the filter essentially pixelates a texture using irregular shapes instead of perfect squares as the pixels. To achieve this, a slope blur filter was applied to most color variation layers inside of Substance Painter. The color, opacity, and size of the Kuwahara pixels could be adjusted per layer, allowing for lots of customization and diverse textures using the same effect.



Figure 5.7: A pot asset before and after applying the slope blur filter

To further stylize the environment, several post processing shaders were made. These were produced early on to guide the development of 3D assets. For example, a “toon outline” shader was made, which applies a thick black outline to assets in the world, giving a similar effect to hand-drawn cartoons. An additional feature of this shader is that the thickness of the outlines adapts to how far

away the object is from the player, so that objects further in the distance aren't overwhelming black blobs. This shader was made with many parameters allowing quick and easy customization, such as the thickness of the outlines and the furthest distance from the camera it is applied to. Finally, a custom render pass was added, meaning that certain objects can be omitted from the shader's effects such as grass and leaves.



Figure 5.8: A pile of rocks without the outline shader.

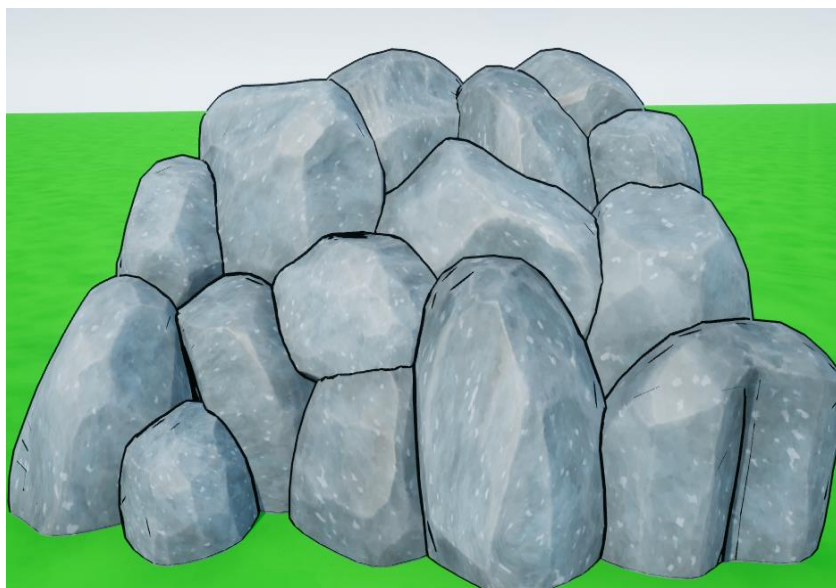


Figure 5.9: A pile of rocks with the outline shader.

Lastly, lighting was what brought the environments all together. An ordinary lighting style was used across all maps. This was partially due to game performance, but also because the art style of our game is eccentric in many other ways, so unconventional lighting was not needed to make the art stand out. However, most lights were given slight temperature changes at the very least to give subtle hints of warmth or coolness, and colored lights were used in some areas. Finally, all lighting maps were baked to improve performance, so they did not have to be calculated in realtime.

5.4.2 Main World

For each of the outdoor maps, a landscape mesh needed to be created. This was done using Unreal Engine's built-in Landscape Mode. For easy control over how the landscape looks, a master material was made and applied to all maps. This material included a Landscape Layer Blend Node, which allows for multiple base color maps to be plugged in to a single material, allowing them to be easily painted on to the landscape mesh. Most of these base color textures were created with opacity maps. These are grayscale images that take two colors and determine where each of them is applied, where black values represent one color, and white values represent the other, and in-between gray values represent a blend between the two.

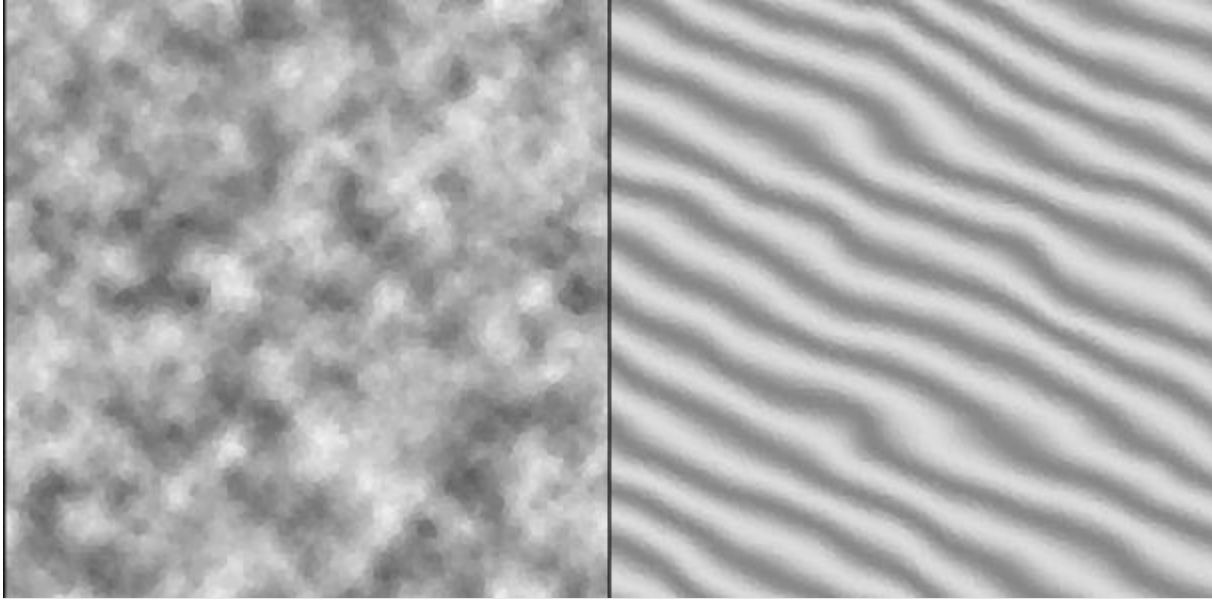


Figure 5.10: Opacity maps used to make the color variation for the grass (left) and sand (right)

Additionally, Runtime Virtual Textures (RVTs) were set up to allow for assets to be blended to the landscape, commonly used with nature-based assets such as rocks and leaves. Instead of using a traditional base color map, these assets take the color of the landscape under them. Additionally, vertex painting can be used to only apply this to part of the mesh. An example where this would be used is to have the landscape's color only applied to the top of a rock asset to make it look like it has a mossy top.



Figure 5.11: A demonstration on how RVTs allow for assets to take the color of the landscape underneath them

Finally, a custom skybox was made to bring the environment together. The default Unreal Engine skybox blueprint actor was used as a shell, and from there custom textures were made in Photoshop and applied. A simple white-to-blue gradient was used as the background and a transparent cloud layer was overlaid on top. Finally, a panner node was used to allow the cloud layer to slowly move across the sky, giving the world some subtle background movement.



Figure 5.12: The Mansion with the Custom Skybox in the Background

5.4.3 *Mansion*

The mansion is where most of the game's main story takes place, so lots of effort was given to make it look detailed and polished. The mansion is organized and neat to give a feeling of elegance and prestige. However, lots of small props and clutter were still added to make rooms feel full such as candles, bottles, and clocks. For each room, a color palette was chosen before any assets were made to give each room a distinct and cohesive look. For example, the kitchen features warm tones of light yellow for the walls, orange-brown for the cabinets, and rose gold for pots and pans. This is balanced out with some contrasting notes of cool gray appliances, dark purple countertops, and navy-blue cabinets in the bar and pantry.



Figure 5.13 The Mansion's Dining Room



Figure 5.14 The Mansion's Kitchen

5.4.4 Market

In contrast to the mansion, the market is meant to look less structured. Each interactable market stall is meant to seem sophisticated and inviting, while still being very intricate. Additionally, lots of clutter was added around the scene, such as piles of crates, barrels, and boxes, some of which are

overflowing with bars of soap. Furthermore, other decorations were added such as seating areas and a mop bucket fountain to fully flesh out the scene.



Figure 5.15: Cleanland Market



Figure 5.16: Wave's stall at the Market



Figure 5.17: Olive's stall at the Market

5.4.5 Beach

The beach was meant to be colorful and vibrant, yet relaxing. A tiki bar called the Squeaki Tiki acts as the heart of the beach and is filled with eye catching decorations like tiki masks and totem poles, as well as brightly colored lighting. The beach itself has some clutter such as chairs, umbrellas, and a volleyball court, and is meant to be a place where lots of NPCs can be casually found. Finally, animated water planes meet the edge of the shore. Waves and faster animation were experimented with, but we found that a slowly moving, level surface provided a more calming vibe.



Figure 5.18: The Beach

5.5 3D Character Art Pipeline

5.5.0 Pipeline Overview

A pipeline ensures the efficient flow of assets between programs and departments. A pipeline strategically addresses short-term, mid-term, and long-term goals and without a pipeline, large-scale projects can encounter significant challenges. For example, a lack of department coordination can lead to bottlenecks that hinder development. Assets may also be developed with methods that create more work later in development. A pipeline has a significant impact on the success of the project and must be robust. The most important thing we learned when developing a pipeline was to identify bottlenecks in art production and to prioritize handing off work to allow every department what they need to begin.

5.5.1 Concept

Character designs were first conceptualized in Photoshop using features of the background we were aiming to represent, elements of the cleaning product's branding that the character was based off, as well as their personality and role in the town. When the design was finalized, the character was sent to the team and Professor Farley for review and critique. After the character was either greenlit or reworked, the character was moved into ZBrush to be modeled.

5.5.2 Modeling

Characters were first blocked out in ZBrush, using multiple subtools to create the muscle structure for distinct parts of the body. Using multiple subtools allowed for quick, on-the-fly editing between characters, as we were able to use base blockouts and inflate/deflate different muscle groups to create new characters with diverse facial features and body types.

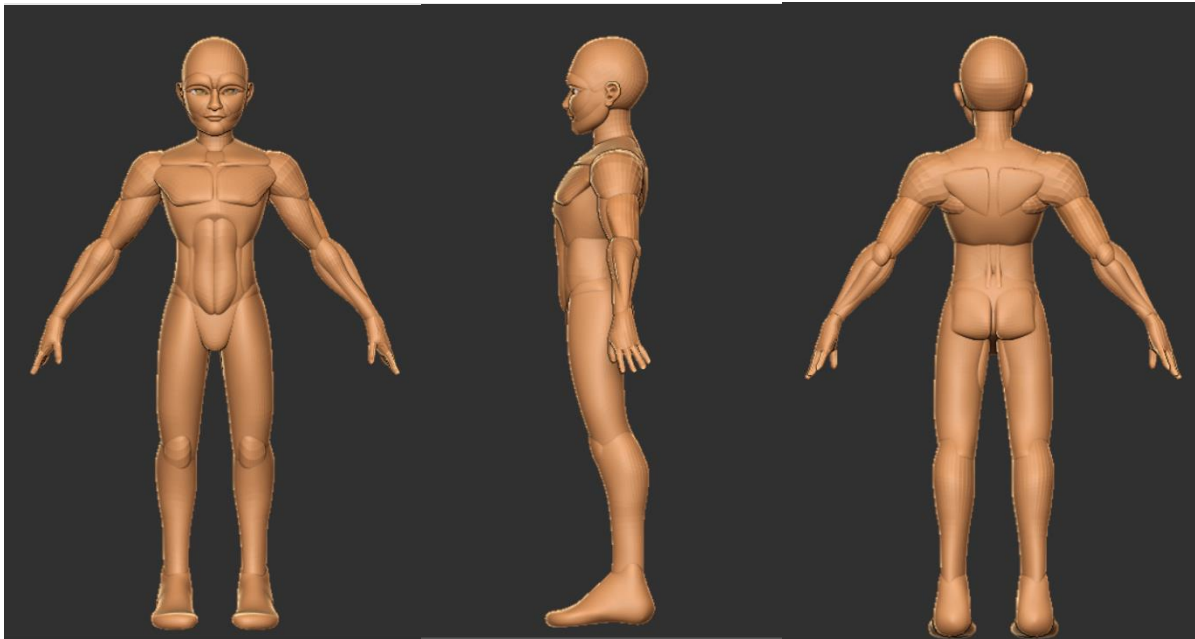


Figure 5.19: A blockout of Beau using different subtools to build the different forms of muscle groups.

After the body and face were approved, the body was merged, dynameshed, and smoothed to prepare for clothing modeling. For clothing, we used a blend of extruding from the base character mesh for shirts, jackets, pants, and other items drawn from the character's form, and modeling scratch for items such as skirts, dresses, hair, and accessories.



Figure 5.20: Beau's full model with smoothed features, clothing, and hair.

From these initial cloth forms, we sculpted in the natural folds and forms with Inflate, Damian Standard, Clay Buildup, Move, ZModeler, Trim Dynamic, and Pinch. Throughout each step of the process, we used masking, ZRemeshing, and polygroups to help achieve the desired forms and polyflow. After the general character was approved, we returned to add subdivisions for further detail on the clothing and facial sculpts to add an extra layer of polish. The .ztl files of the characters were then passed onto the technical artists for retopology.

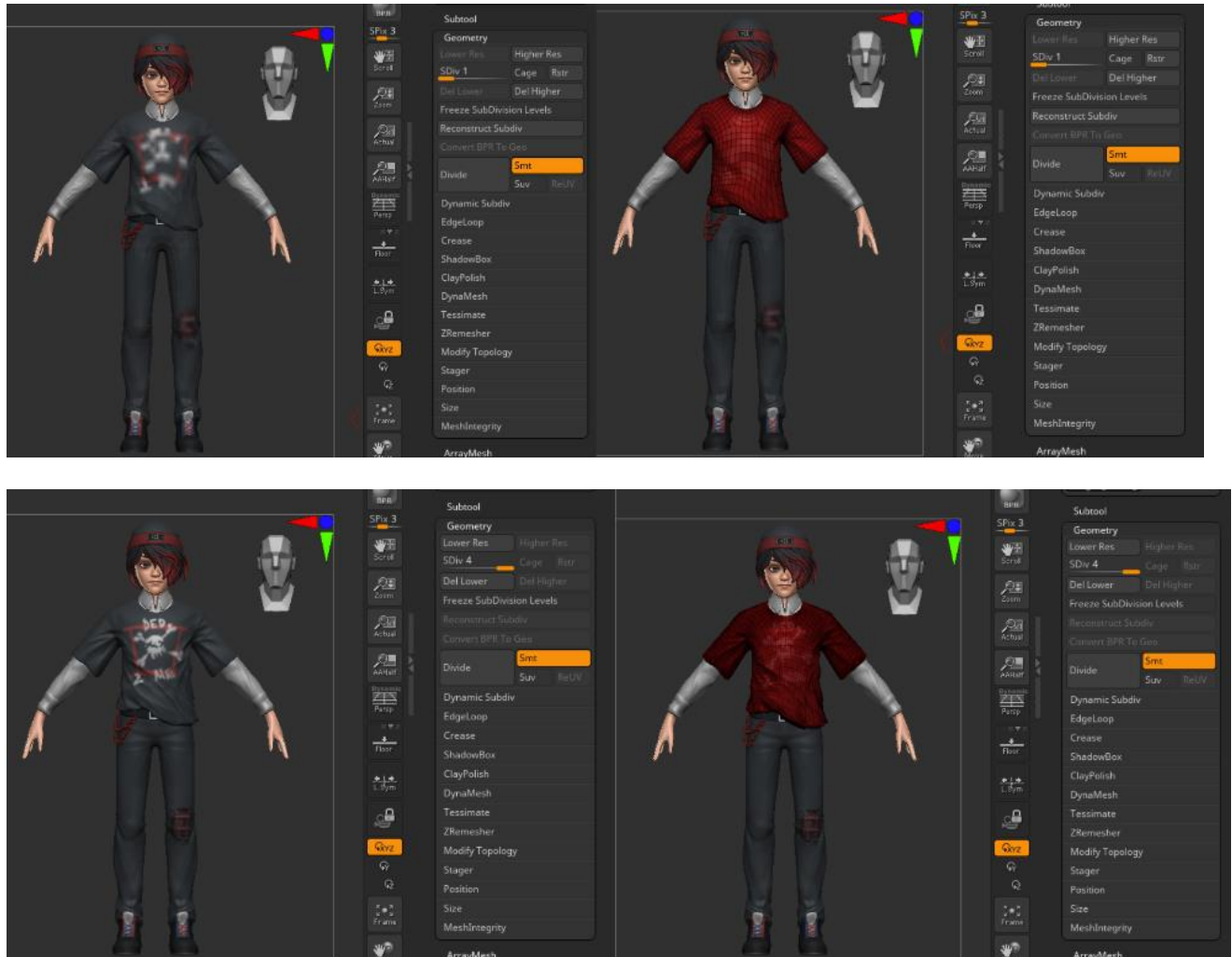


Figure 5.21: The differences in subtool subdivision levels; getting a low poly surface and subdividing it to sculpt will help technical art create retopology easier.

For the early builds, characters were decimated to get them in a low enough state where they would work in engine. However, this was neither ideal nor sustainable, as characters were still relatively high poly and could not be rigged in that state, so edits had to be made to the models, mostly involving ZRemeshing by half, to reduce them to a usable state.

5.5.3 Texturing

The technical art department handled character UV mapping and material setup. UVs were automatically inherited from the base mesh asset, which is explored more in depth in the technical art pipeline. However, this base mesh was unwrapped using methods to ensure hidden seams and higher texture density for the face, hands, and eyes. The UV layout was optimized per character as each character's body geometry could be deleted if obstructed by clothing. Uvs were further optimized by stacking similar shells. UV seams for clothing meshes were cut where natural seam lines would occur due to the sewing pattern of the garment. Once UVs were created, 3 to 5 materials were assigned to the character geometry for faster performance and setup in Unreal Engine. Finally, high, and low poly assets were exported for baking in Substance Painter.

After the low and high meshes were prepared, they were imported into Substance Painter for baking and texturing. All character surfaces had 4 main base components: base color, ambient occlusion, curvature, and position. Each of these was created by using a base fill layer and tweaking the color, roughness, and metallic values based on the material. To create the ambient occlusion, curvature, and position layers, a black mask was added to their labeled base fills and their generator was applied. The ambient occlusion and position layers were usually the same color, which were used to bring out the shadows on the material. Curvature's color was usually a slightly lighter tone than the base color to bring out the highlights and/or higher surfaces. Skin and eyes had custom painted details, as well as some other clothing items.

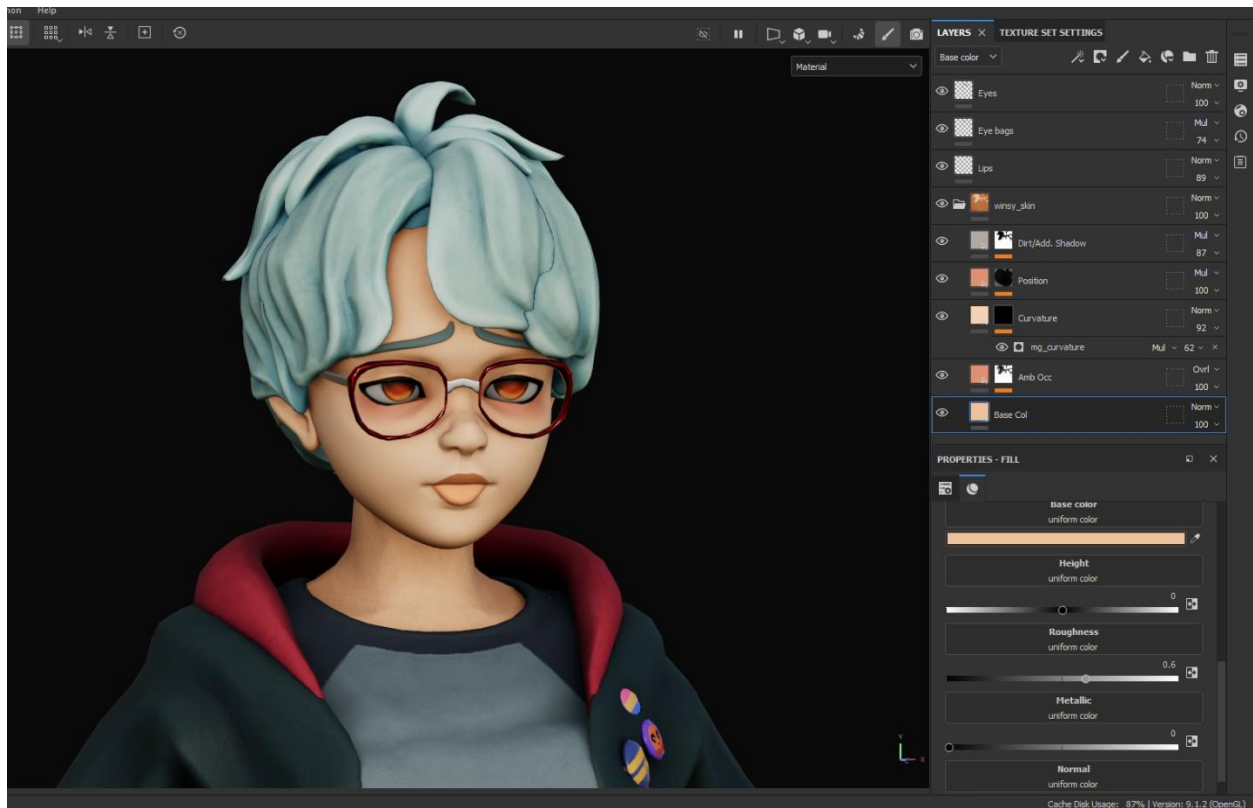


Figure 5.22: A texture breakdown for Windy’s face, focusing on the skin’s base color.

Clothing patterns that would be difficult to paint by hand in Substance, like Lysa’s half tartan pants and Shawn’s flannel, were instead created in Photoshop as alpha maps. These maps were then imported into Substance and procedurally applied across the mesh in Substance to uniformly get the details more accurately onto the model. To add depth to meshes, layers of differing height were used, including layers to paint in specific details.

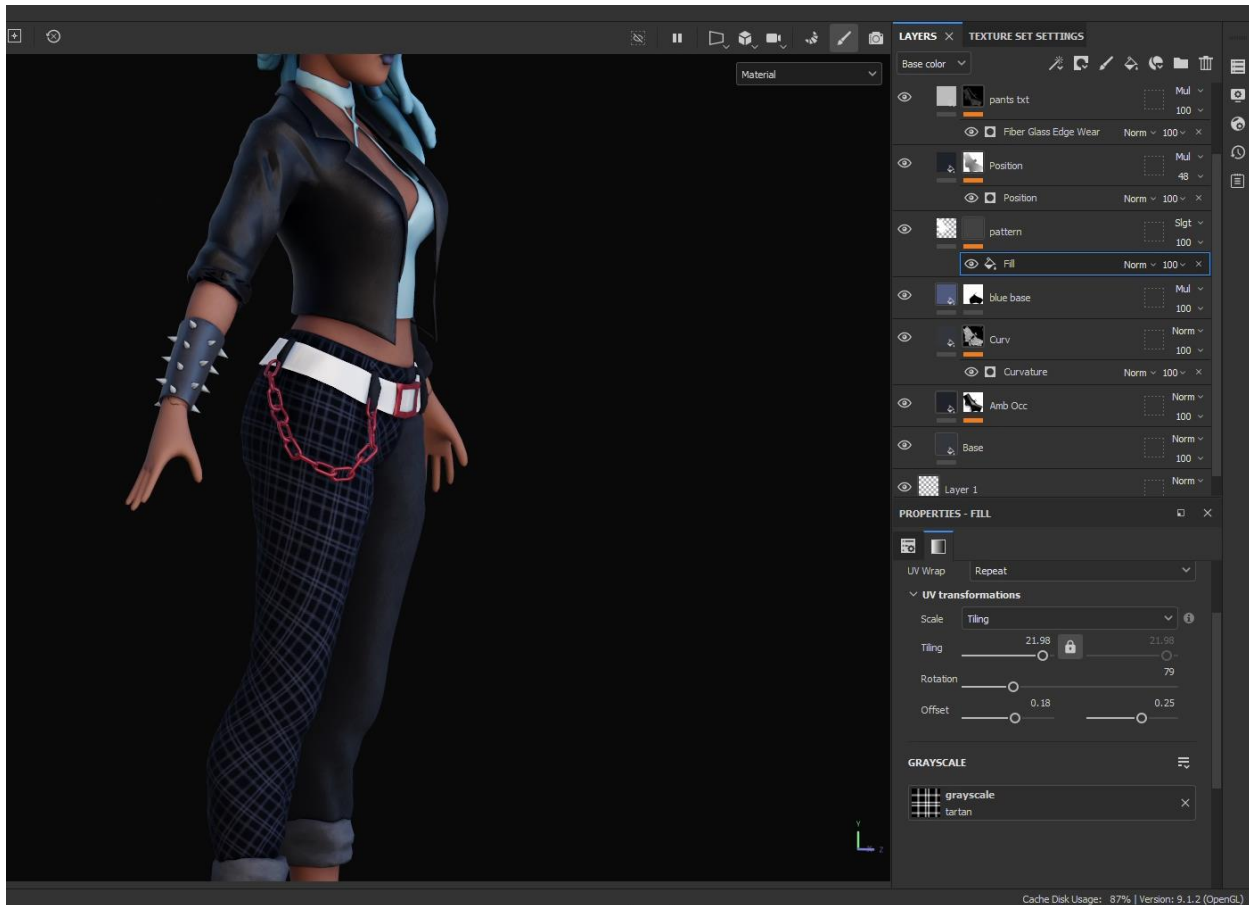


Figure 5.23: A texture breakdown of Lysa's pants, showcasing the procedural application of the tartan pattern.

5.6 3D Environmental Pipeline

5.6.1 Layouts and Concept Art

In collaboration with the writing team, 2D layouts and concept art were made for environments and levels. This was done to ensure that each environment matched the writing team's vision before work was put into modeling them. Also, photo references were provided to guide the overall feel and style of the space. This made the overall pipeline much smoother, as it was clear which assets needed to be included in the final product for story purposes. The actual

process for this was quite simple, reference images were gathered using Google and Pinterest, and the layouts and concept art were made using Photoshop.



Figure 5.24: Concept Art and Reference Board for the Squeaki Tiki

5.6.2 Grayboxing

Grayboxing is blocking out an environment in its simplest form with basic shapes and placeholder assets to establish proportions and level flow. This allows for the general idea and feel to be presented in 3D space, making for rapid and drastic changes to be easily made to a level's overall shape without backtracking or redoing finished models. Although grayboxed levels will look rough and clearly unfinished, they should be fully functional, allowing the design and programming teams to begin their work on a level without having to wait for the artist to finish completely.

The exact process of modeling placeholders varied between types of assets, but most were made using *Autodesk Maya*. Graybox assets were made with proper topological flow, consisting of as little of a polycount as possible. Additionally, objects were made with mostly quadrilateral polygons and minimal triangular polygons. This is because surfaces with polygons that have more than four sides can often cause performance and lighting issues in engine. The retopology process is started this early to allow for later steps of the process to go smoother and quicker. For example, it's a lot more complicated and time consuming to UV map an object with a high polycount. Also, lots of attention is put into setting up exact proportions that transfer over into *Unreal Engine*. During this process, there is lots of importing and exporting between programs to get the proportions exactly right.



Figure 5.25: Graybox for the Squeaki Tiki

5.6.3 Detail Modeling

Detail modeling is the process of further enhancing the placeholder graybox assets into finished products, except for texturing. For most assets, this process is continued within *Maya*. This includes adding more edge loops to exaggerate and modify the shape of an object while

retaining its proportions, and the smooth preview mode can be used if necessary. Even hard edges are given a slight bevel to avoid a boxy look. Lastly, if needed the smooth tool is used for more organic assets.

Occasionally, *ZBrush* is a much more effective program for detail modeling than Maya is. This is usually if the object is highly organic, such as trees and rocks. Subdivision levels are used to produce both a low poly and high poly version of the asset. The clip curve tool is used to wear down edges, acting as a non-uniform bevel with more user control. After that, commonly used brushes to add detail are the clay buildup, trim smooth border, dam standard, and pinch brushes. *ZBrush* also has a particularly useful tool called ZRemesher, which automates the retopology process for high-poly models. Once detailing is done, the lowest subdivision level with an accurate silhouette and the highest subdivision level are exported as the low-poly and high-poly models, respectively. Later in Substance Painter, the high-poly mesh can be projected on the low-poly one through a normal map, which substantially helps with game performance.



Figure 5.26: Low-Poly (left) and High-Poly (right) versions of a tiki mask asset

5.6.4 UV Mapping

UV Mapping is the technical process of projecting a 3D model's surface onto a 2D image, allowing for texturing. UV maps act as coordinates that guide how 2D texture maps look, so they can be properly re-projected back onto the 3D model. This varies on a case-by-case basis, depending on the model's complexity.

In *Maya*, assets are UV mapped manually. The bulk of the UV mapping process is determining where the object's seams should be. The goal is for the object's seams to be hidden, so they are often put on the bottom or backside of a mesh so they cannot be seen in-game. From there, one or more UV shells are formed, which often are stretched or distorted. This can be solved

by using the Unfold and Optimize tools. Finally, the shells need to be properly aligned or else the textures will be rotated, which is done with an easy click of the Orient button. Finally, the UV shells cannot overlap, or else there will be issues later in Substance Painter. Maya can do this by pressing the Layout button, which automatically orients and resizes the shells proportional to each other, so textures are not noticeably bigger or smaller on different sections of the model.

If the asset was modeled in *ZBrush*, the process is much simpler. *ZBrush* has a UV Master ZPlugin that will automatically create seams and UV Map a mesh. Some important things to note are that the object is UV mapped on the low-poly mesh, and that symmetry for the UV map is either enabled or disabled properly.

5.6.5 Texturing

All textures for the game were made in Substance Painter, Substance Designer, Photoshop, or a combination of all three. To accelerate the texturing process, several texture templates were made inside of Substance Painter. Instead of starting each asset from scratch, these templates could be simply dragged and dropped onto the 3D asset and adjusted accordingly. This was done to keep style consistency between artists as well as drastically speed up the pipeline.

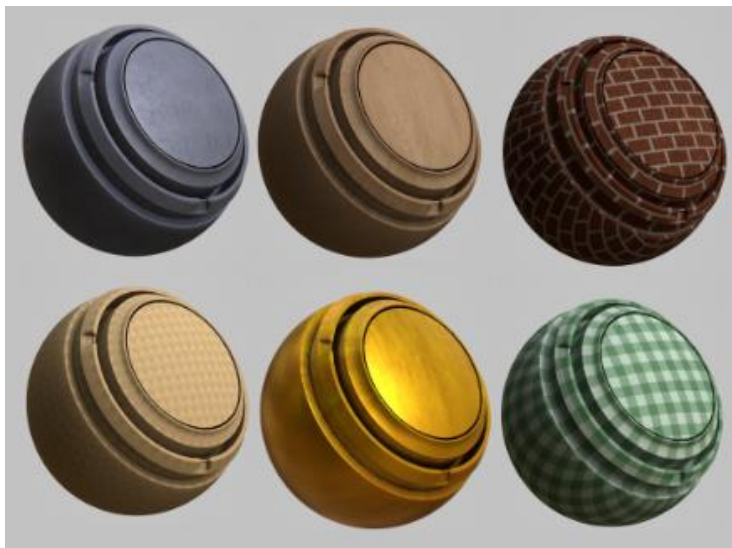


Figure 5.27: Examples of the Substance Painter Templates

Most textures were some variations of the stylized grunge template, which consisted of a base color layer, two color variation layers, a curvature layer, and a dirt layer. Most of these layers are simple and self-explanatory, but the curvature layer is slightly more complicated. This layer detects where the edges of the model are and can be used to add things like rust around the edges of a metal object. On top of these layers, a roughness/metallic layer was added. This affects how the object interacts with light and determines if the model looks shiny or matte. Other objects require more layers and attention, such as a pattern or a normal map. Normal maps create the optical illusion that an object's geometry has a height change. For example, this is used to create divots or scratches in a mesh's surface. Once the texturing process was complete, a base color, normal, and AORM (Ambient Occlusion, Roughness, and Metallic) map are generated, and can be exported and imported into Unreal Engine.

5.6.6 Polish and Juice

After each environment was textured and deemed presentable, they were later revisited, and polish was added. This includes adding extra decoration pieces, trying to minimize seams in textures (especially walls, floors, and ceilings), and adding VFX and lighting.

At first, a basic cel shading lighting technique was used. This converted gradient shadows to solid color blocks. Later, this was replaced with a more dynamic lighting method. For most cases, a general point light was put in the center of the room, which was the main ambient light source with shadows enabled. Then, point lights were added to other light sources such as lamps, ceiling lights, and wall lights. These point lights were given a slight warm temperature change to give the scenes a warm glow, and had shadows disabled to prevent several shadows per object overlapping. Lastly, spotlights were added outside of windows pointing inwards. These lights were given a slight cool temperature change to balance out the level. In some cases, such as the tiki bar, special colored lighting was added. Finally, the lighting was baked. This means that the lighting was calculated before building instead of during gameplay, allowing for the game to run smoother.



Figure 5.28: Polished version of the Squeaki Tiki, with baked lighting and simple VFX

5.7 Recommendations

5.7.1 2D

Put a strong emphasis on polished UI, and do not push it to the last second. UI is one of the major keys in game development to achieving polish on a game. Unfortunately for us, this task was consistently pushed off during development, and led to a scramble at the end to attempt to juice up the gameplay experience. Having a clean and established UI early in production can help shape and support the overall immersion of the gameplay experience, as well as improve stylistic cohesion.

Cheat. Cut corners where possible (*ethically*). As a wise Professor Farley Chery once said, “if you’re not cheating, you’re doing it wrong.” One of the main reasons that 2D was successful at pumping out dialogue sprites as quickly as it did was because the 3D sculpts were able to be used as a rough base and helped establish facial proportions for the sprites. While not a direct trace, they served as easy references to be “stolen” from. The key point of this recommendation is for teams to realize what elements of their art can be used in a way that can improve the look, speed, and appeal of other assets, and to use what you already have.

5.7.2 Characters

Be ready to pivot and willing to compromise. The biggest pitfall the art team ran into during character modeling was style cohesion between different artists. Characters between artists looked noticeably different, and the team decided to offload the main playable cast to the lead artist to ensure that the playable characters all looked cohesive early on. This also ensured they

could be moved through the pipeline quickly and effectively. The other character artists on the team then worked on their NPCs over a longer time to get more practice with the style, while backups were made by the main character artist as placeholders until the style was able to be matched. This also allowed for the other modelers to reprioritize their workload and improve their content output from their specialty disciplines.

Be loose about character designs and models until the character ideas are finalized. One of the elements that otherwise would have contributed to a positive pre-production that backfired was entering the development process with character ideas and designs. As the year progressed, it became clear that as the characters were changing and developing into their finalized versions for writing, their designs no longer suited their personalities. Thus, these characters had to have their clothes entirely redesigned and modeled, in some cases twice. Thankfully, due to the lead character artist's speed, this was nonissue, and new iterations were able to be created quickly and efficiently. However, in a different pipeline, this could have severely damaged production speed. It is imperative for future writing and art teams to work hand in hand to make sure that character creation moves smoothly without excessive need for reworks.



Figure 5.29: Vic's first, second, and third iterations through development, sans shoes.

5.7.3 Environments

The key to having an effective pipeline is modularity. Especially with the amount of content that our game contains, there is no shame in reusing where you can. For example, instead of making walls and floors for each environment, one of each asset can be made and used for each scene. This also helps with understanding space and proportions, as having one default wall asset can act as an excellent scale reference. The more assets you have, the more you can compare them to one another. This also helps the design team set up level shells with ease. We did not implement modular assets like these until partway through development, and once we did the pipeline for multiple teams accelerated greatly. Similarly, implementing modularity in texturing was something we did from the start through the templates made within Substance Painter. It cannot be stressed enough how helpful these were and allowed for more time and energy to be put into various parts of the pipeline, and made it much easier for ISPs to learn the program.

6 Character Technical Art Pipeline

6.1 Developing a Pipeline

A technical art pipeline was created to turn character art into game assets. This process included Optimization, Rigging, Animation, Model Preparation for Texturing, and Integration in engine. This section will cover the design of the *Clean Sweep's* technical art pipeline, outlining its rationale, discussing the challenges it encountered and concluding with recommendations.

This section is designed as a guide for several critical reasons which we would like to address. Although WPI's technical art curriculum is robust, offering fundamental courses like Pipeline Management, Rigging and 3D animation, WPI does not yet offer a course focused on Pipeline Development specifically, leaving a gap in Digital Content Creation (DCC) project management and collaborative efficiency training that needs to be addressed.

We are well aware of where students face the most challenges and can help students succeed faster. Through the privilege of being teaching assistants in technical art courses and having one-on-one mentorship with Professor Chery, we have unpacked some of their game design philosophies and feel compelled to share them. This opportunity has given us hands-on experience revealing students' hidden frustrations. Many students struggle with concepts like Rigging and Retopology, often abandoning them in favor of auto-rigging tools like *Mixamo* and auto-retopology tools like *Zremesher* due to a lack of proficiency. Professor Chery is beginning to tackle this issue. We also hope to address these recurring difficulties ourselves throughout the guide by offering practical, step-by-steps on the most challenging components of 3D technical art. Rigging stands out as a particularly challenging discipline in game

development. Having personally experienced its complexities and the frustrations of students, we feel compelled to offer what we know to help them succeed faster.

Lastly, we aim to initiate the circulation of workflows and tools. Students have asked us for guidance specific on workflows and tools to help them progress with their projects. We provided the necessary information to help them overcome their challenges and they have reciprocated sharing what they have learned. Ultimately, we hope this section can serve as a guide that will promote the sharing of information and resources for the WPI IMGD community. All workflows and tools are available at [Clean_Sweep_Project](#) found in Appendix K.

6.1.1 Process

Developing a game pipeline begins first by shaping a workflow through iteration and exploration. Second, focus shifts to streamlining and identifying bottlenecks and implementing strategies to address them. Ultimately, a standardized pipeline workflow emerges that scales to accommodate a team. This remained an ongoing process of iteration. Every week we would encourage analyzing time sheets. We used *Toggle Track*, a time tracking software for teams, to compare development time of different workflows.

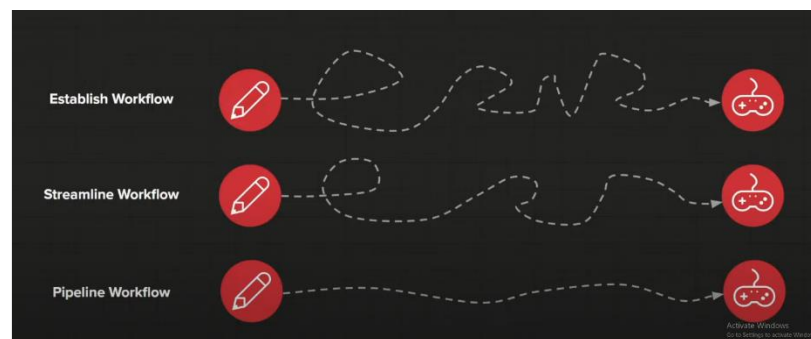


Figure 6.1: The development phases of a pipeline

6.2 Project Management

We achieved a level of iteration, collaboration, and efficiency by developing a project management system to support the team. Without this file system, the project would not have scaled effectively. The system saved considerable time at various stages and was essential for the success of the technical art department.

6.2.1 Version Control

We chose OneDrive over other version control such as Git or Perforce for several reasons. Version control software allows large projects to manage changes to files over time. It encourages collaboration, file protection, managing backups, history and has recovery features if issues occur. OneDrive was the keystone of the technical art pipeline enabling us to manage the entire project's storage, access assets remotely, and manage software licenses. Git struggles with large file sizes, so it was not an option as many tutorials needed for training exceeded the file size limit. WPI offers 1TB storage for an outlook group and through this method, we did not even approach the storage limit.

Git is also less intuitive to artists and requires time and training. With OneDrive artists can save files like they are accustomed to. OneDrive also allows the department directors to check the work of their assistants, accessing their files at any time. This ensures assets are on track and meet guidelines early in their development.

Although OneDrive is not perfect, its video streaming is frustratingly cumbersome and outdated and it has a one sign-on per user limitation, we felt it exceeded expectations as a free source control. We would not have succeeded without it.

6.2.2 Remote Access Software

Remote Access Software is an essential tool for managing work. This is because 3D art is graphically intensive and often requires a dedicated PC which cannot conveniently be transported. It is important for artists to have remote access software to access work during an emergency or on the go. *Parsec* was introduced to the art team by the programmers. It is a free and impressively fast remote software option. When using *Parsec* on a good connection the latency is so negligent you may forget you are working remotely. *Parsec* was also used across departments. For example, the programmers were able to attempt a mac build on the technical artist's laptop. This also allowed them to collaborate directly as if they were on the same computer in the same room. Without Remote Access software production can slow down unnecessarily and you may find it to be the Swiss army knife that gets you out of that troubling situation.

6.2.3 Setting Projects

When working on large-scale collaborative projects with Digital Content Creation tools (DCCs) like *Maya* and *Houdini*, proper path management is critical for collaboration. Setting the main project directory should not be overlooked as it facilitates the proper handling of dependencies and enables multiple artists to work on the project concurrently.

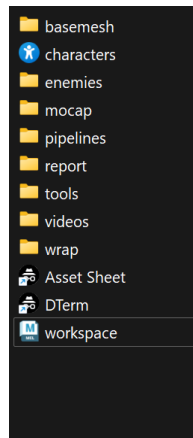


Figure 6.2: location of workspace file in the file system

Setting the project in *Maya*, like any other DCCs is critical for the program to interpret the relationship of files. When a project is set in at the root in *Maya*, creates a workspace.mel file. Refer to Figure 6.2 for its location at the root of the directory. This allows *Maya* to navigate the file structure and correctly associate the needed dependencies, models, textures. Set projects allow other artists on different machines to access dependencies in their scenes automatically. The project should be set for every new scene.

6.2.4 File References and Namespaces

File references and namespaces are an essential part of managing a project within *Maya* and facilitating an efficient and collaborative pipeline. Their absence can lead to challenges during production.

6.3.4.1 References

File References are like importing assets directly, but they offer a slight advantage. Behind the scenes, referenced files refer to data rather than copying it. Imagine a library book system where each person borrows the same book rather than owning an individual copy. In the context of *Maya*, this

allows an artist to update the assets from outside the main scene which automatically reflects in everyone's copy of the asset improving pipeline efficiency.

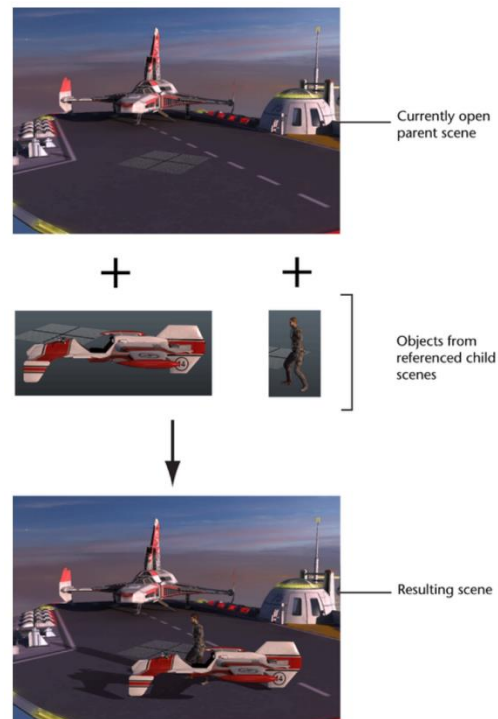


Figure 6.3: File references in Maya

References are useful when rigging unfinished props and characters, creating environments with unfinalized assets, and animating unpolished characters.

References allow every department to get started faster.

6.2.4.2 Namespaces

Namespaces are another essential tool for managing assets. Namespaces act like containers or file folders and group related content together. Namespaces create unique identification to help avoid naming conflicts. In Figure 6.4. a namespace called 'weapon' is created

in the Namespace editor for the `katana_weapon` object. As a result, a prefix `'weapon:'` is added to the object's name. This differentiates its content from similarly named objects.

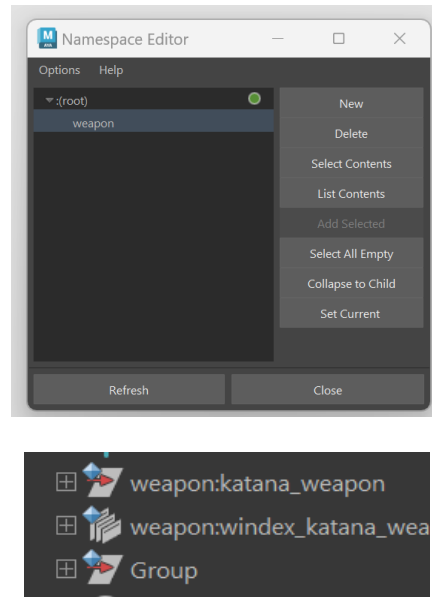


Figure 6.4: Namespace editor and resulting outliner

Namespaces are necessary when importing motion capture data from motion capture suits and software like *Rokoko* and *Axis Studio*. They are also needed when importing objects with the same names into one scene. Without namespaces, similarly named objects will lead to unpredictable behavior in the hierarchy.

6.3 Pipeline Development

6.3.1 Exploring Programs and Tools

Move AI, *Axis Studio*, *Advanced Skeleton*, *Wrap* and *ngSkinTools* were premade tools, equipment and software used throughout the technical art pipeline. It is important to research capabilities and limitations so they can be used effectively. We decided to use *Move AI*, a motion

capture tool, early in development without thoroughly exploring its capabilities. It was efficient but not appropriate for every animation due to foot contact issues. Unfortunately, this forced us to recapture every animation again in C term with *Axis studio*, a more cumbersome yet effective option.

The *Advanced Skeleton* plugin is a biped auto rigging tool for quick iterative rigging. Rigging is an arduous process prone to user error. A bespoke character rig can take weeks to develop by hand. In a production environment, this is not practical. *Advanced Skeleton* generates a consistent rig hierarchy for every character which allows making rigging easier.



Figure 6.5: Advanced skeleton rig

WRAP is a topology transfer tool. It can optimize character art by projecting a game ready basemesh onto a high-poly character. Additionally, UVs and vertex order are transferred allowing blend shapes and weight data to be reused. This process, introduced to us by Professor Chery, saved significant time in production.

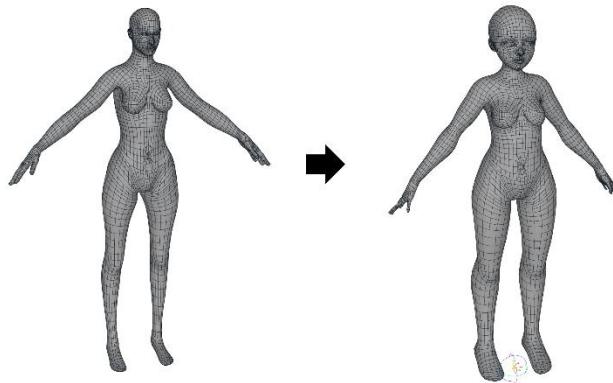


Figure 6.6: Topology transfer using Wrap

6.3.2 Tool Development

If premade plugins, tools, or assets cannot be found, create your own. Tools were developed to eliminate redundant tasks prone to user error and standardize procedures (see Appendix K). An example of a redundant task being posing the Janitor hand to grip the broom in every animation. Tools were created in Python using *Charcoal Editor*. Within most DCCs, shelves are a way to organize related tools and share them with the entire department.

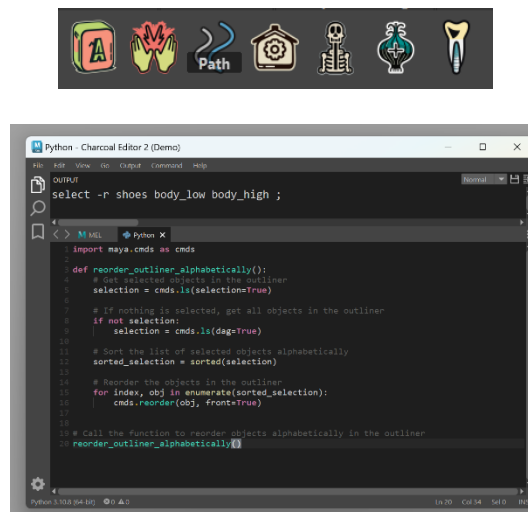


Figure 6.7: Developing tools with Charcoal Editor

When developing tools do not overlook documentation, instructions, and bug tracking. If code is poorly documented and explained, it is difficult to share. Creating an ecosystem of tools for other game developers is important.

6.3.3 Asset Development

6.3.3.1 Base Meshes and Retopology Research

As technical artists our biggest priority was efficiency. This game had 35 concepted characters, so it was necessary to think of the big picture. Base meshes were critical to our Wrap optimization and rigging workflows to improve efficiency. With guidance and revisions offered by Professor Chery, we developed base meshes with game-ready topology. This process was essential to our strategy for developing *Clean Sweep* and will be covered in depth.

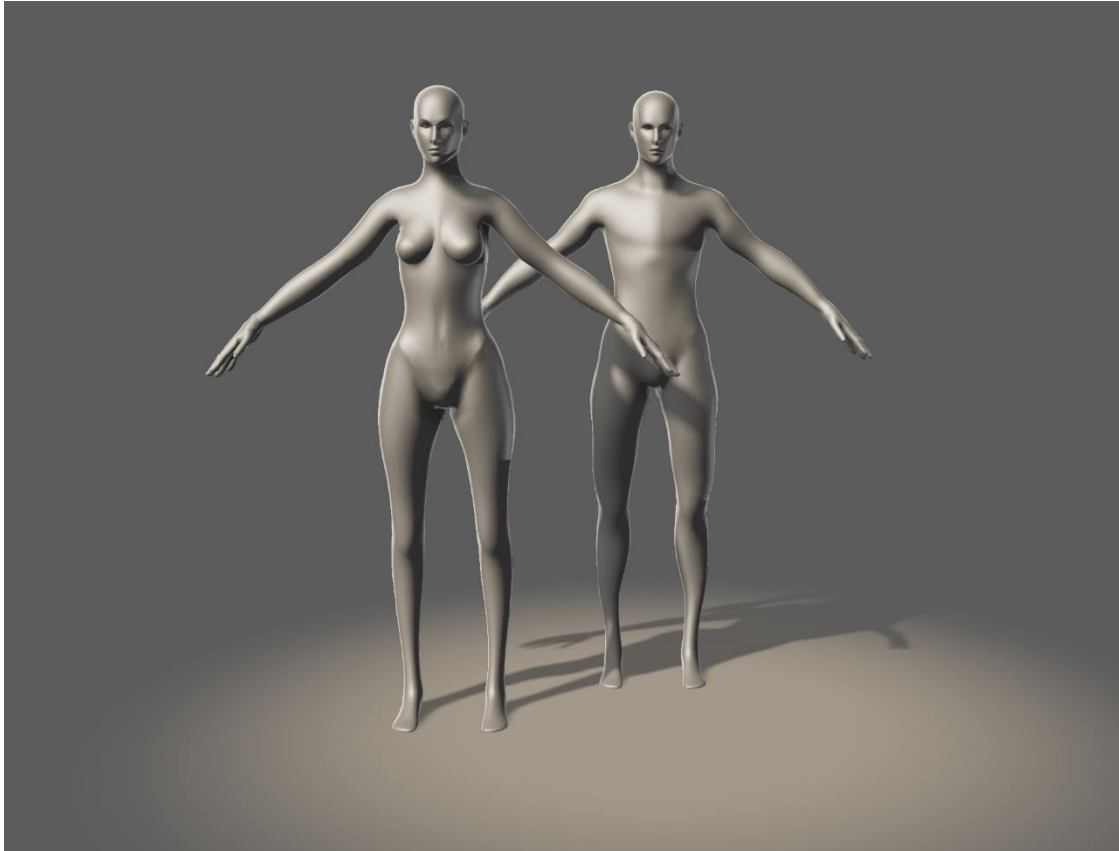


Figure 6.8: Base meshes with female and male anatomy

6.3.3.2 Developing the Base Mesh

When using *Wrap* and creating a base mesh to retopologize characters, understanding basic topology concepts is essential. Topology is the way geometry, edges, and faces fall across the surface of the model. Topology is especially important in real-time character animation for games as it affects performance, the silhouette of the mesh and its ability to be rigged for animation. Because this is our initial attempt at creating a base mesh, we recommend exploring professional-level resources to cultivate a more comprehensive understanding of topology but feel free to refer to the images to grasp the basic concepts.

6.3.3.3 Polygon Terminology

When developing meshes that need to deform, strive for a quad-based mesh. Sometimes triangles are unavoidable so try to hide them. Completely avoid Ngons and Non-manifold geometry which violate standard geometry rules and can lead to problems when modeling and rigging.

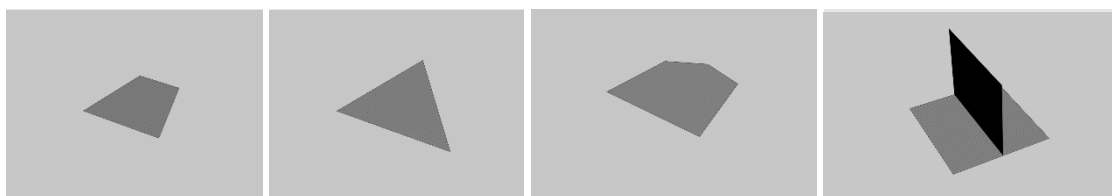


Figure 6.9: Quad, Triangle, N-gon, Non-manifold

6.3.3.4 Poles and Controlling Topology

Poles refer to vertices with three or five incoming edges instead of four. Poles can cause shading and deformation artifacts. However, they are an essential part of modeling and appear everywhere from bevels and insets to primitives themselves. Begin to identify three and 5-poles.

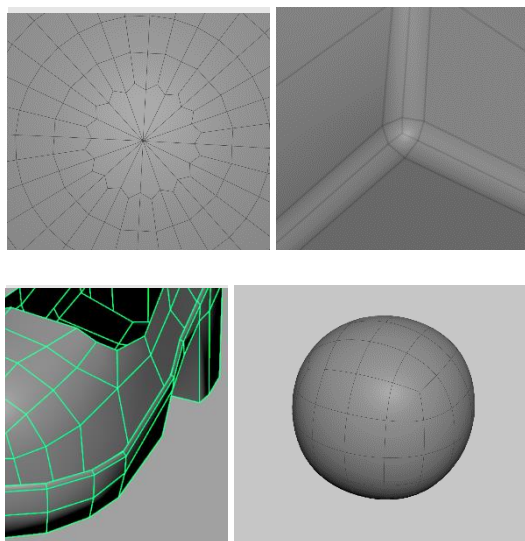


Figure 6.10: Pole Examples

3-poles and 5-poles can be used strategically to redirect edge flow. Placement of poles, especially on areas that do not deform, is necessary when retopologizing a character to reduce edge counts and control edge flow. Typical patterns and uses will be covered.

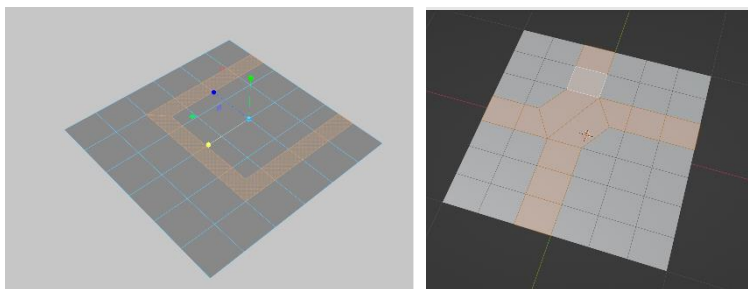


Figure 6.11: Pole Redirection Patterns

Poles are also used to connect round forms like cylinders or spheres to a grid. If the round form uses 8 edges it can easily connect to a grid of faces. This important concept is used throughout modeling and retopologizing areas of the body where round forms connect to the body.

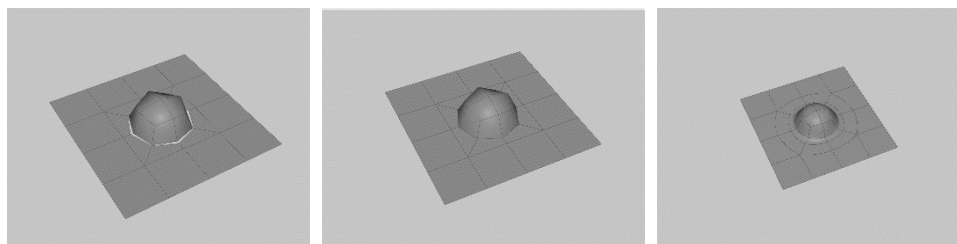


Figure 6.12: Using 8 edges to connect a round form to a grid

6.3.3.5 Common Reduction Patterns

These edge reduction patterns were useful when one location on the mesh was higher than another. This will occur throughout the retopology process. It is important not only to become familiar with them but to also recall and execute them from memory.

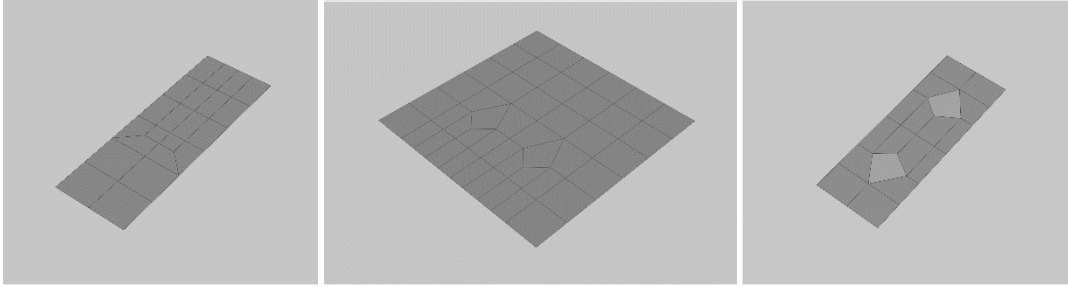


Figure 6.13: Inset reduction and diamond reduction

6.3.3.6 Localization

Localization is the guideline that high-density geometry should only affect the part of the model where it is needed. The hands and face are locations where density is high, however sending extra edges to the rest of the model can cause deformation and shading issues. Redirection and reduction patterns are used to localize topology to areas where extra density is necessary while avoiding unnecessary distribution throughout the model.

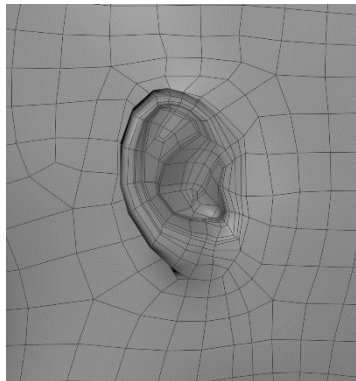


Figure 6.14: Localization of the ear geometry preventing extra edges throughout the face

6.4.3.7 Deformations and Local Density

The density, flow, and orientation of edges contribute to the model's ability to deform when rigged. Areas of compression like the joints, mouth and eyes require more density to support deformations.

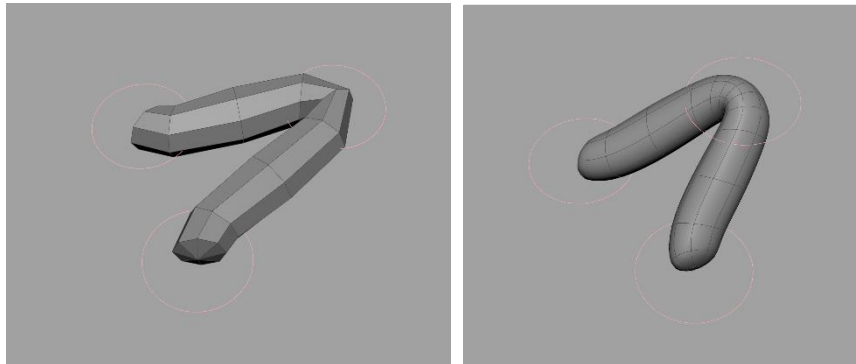


Figure 6.15: Mesh lacking density in deformation area (left). Additional edges in deformation areas (right)

6.4.3.8 Body Retopology and General Guidelines

The following list is a checklist to be used when developing the retopologized model. These easily overlooked steps should be considered before rigging can begin. If some of these processes are not complete, the mesh may cause rigging issues.

- Perpendicular edge loops to the major axis of the form
- Grid of evenly distributed uniform faces
- Anatomy support loops for secondary forms like the scapula and elbow allowing for an easier selection of faces when painting weights and resulting in more believable deformations
- Extra edge loops in deformation areas like knees, elbows, eyelids, knuckles, etc.
- No Spirals

- Manifold geometry and no N-Gons

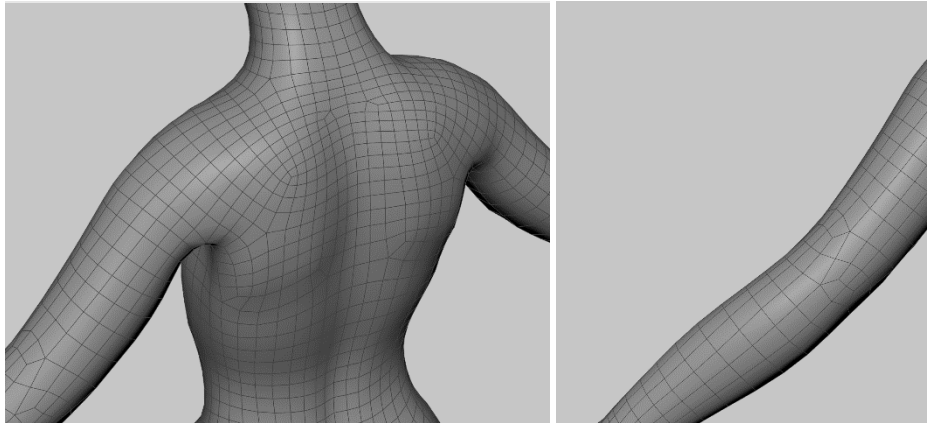
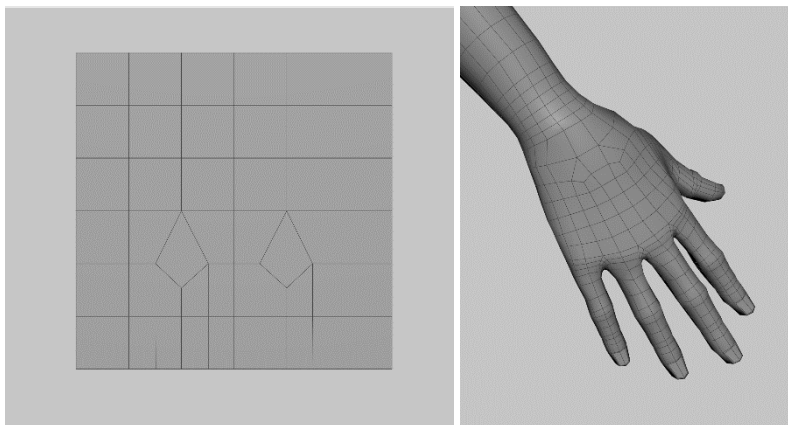


Figure 6.16: Using poles to create anatomy support loops for secondary anatomical forms on Scapula and Elbow

6.4.3.9 Hand Retopology

The topology of the hand must support its natural flexibility and articulation. Edges are added to areas of deformation like the knuckles and wrist. This ensures three edges for the knuckles and five for the wrist, plenty to allow for bending. In-between the fingers may require extra edges as well. Edges are redirected to support the anatomy of the hand specifically to allow the cupping action of the meta-carpal joints and thumb. Reduction methods were used to match the edge count of the hand to the arm.



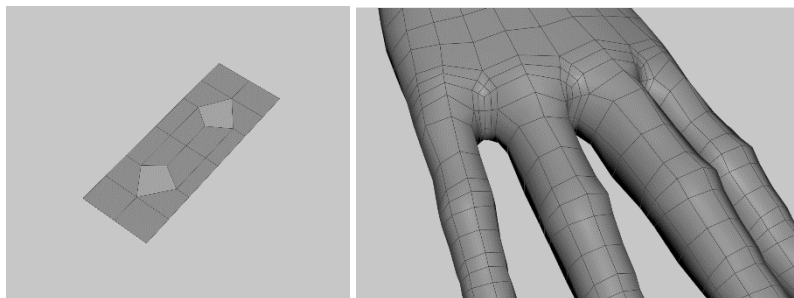


Figure 6.17: Diamond Reduction Pattern used to increase density between fingers

6.4.3.10 Head Modeling

The head was sculpted in Zbrush and retopologized to connect it to the body. A face bag and quad sphere eyeballs were added to support facial animation.

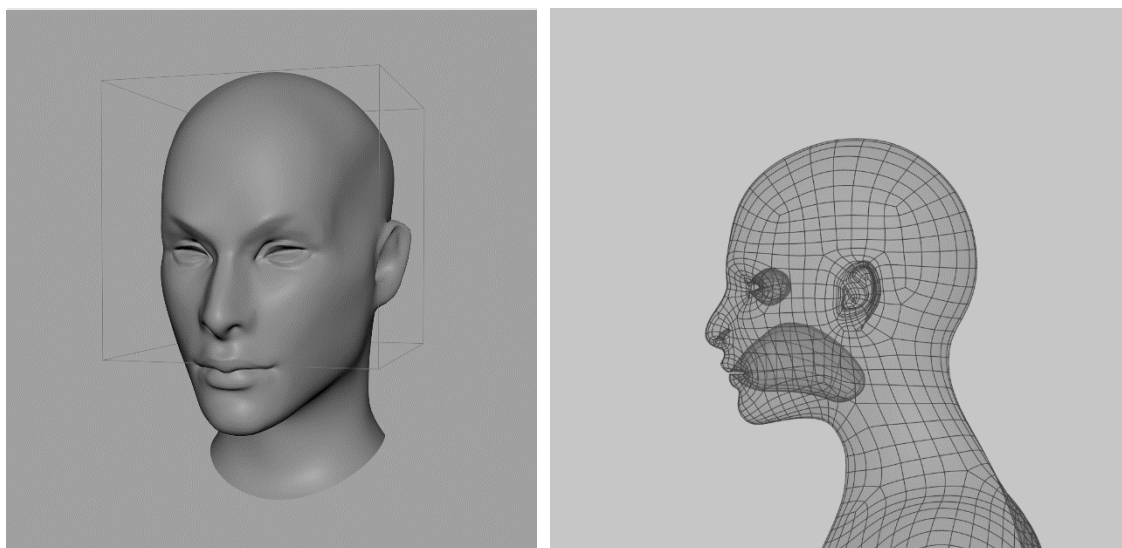


Figure 6.18: 3D sculpted head (left). Retopologized head with face bag (right)

6.4.3.11 Head Retopology and Guidelines

The head must consider more complex deformations and the underlying musculature of the face. Good facial topology will support rigging and animation for facial expressions.

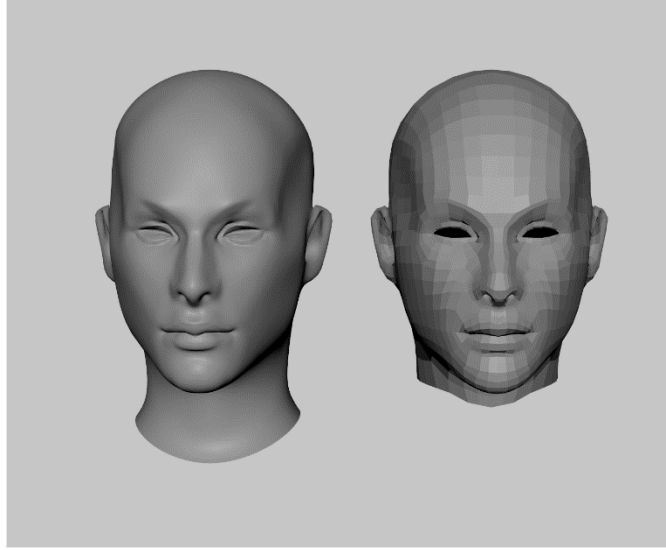


Figure 6.19: High-Poly head (left) and retopologized head (right)

The topology of the head should echo the facial anatomy. The pulling of the facial features to create expressions is driven by this underlying flow of faces. Higher density geometry will allow for stretching and compression integral to facial expressions.

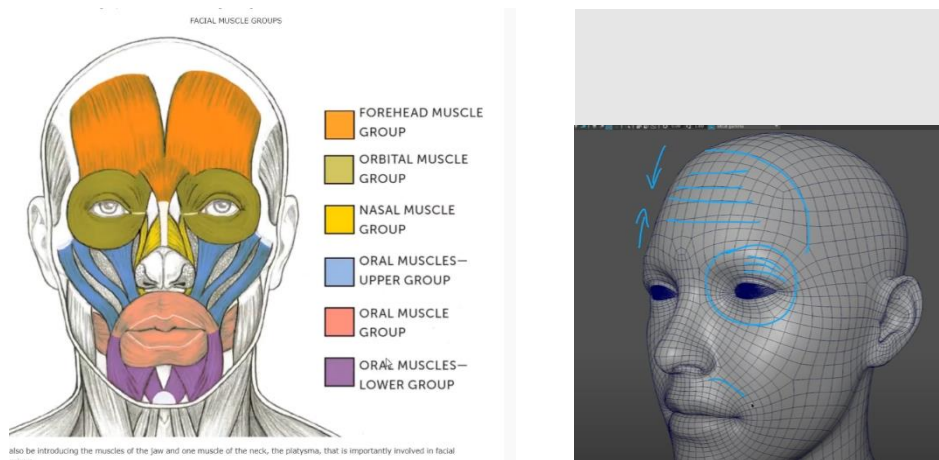


Figure 6.20: Facial Topology following facial anatomy

Proper pole placement will drive the face loops of the retopologized head. Well placed poles will cause loops to fall into place. We recommended studying professional level work once the basic concepts are understood.

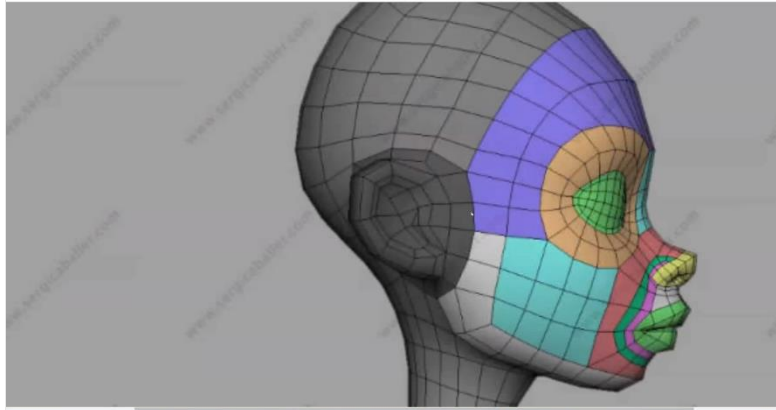


Figure 6.21: Sergi Caballer guide on pole placement

6.3.4 Miscellaneous Asset Development

Configuration files were exported from DCCs to speed up processes. For example, [Human IK](#) definition presets and skin weights were exported from [Maya](#) and spline data was exported from [Wrap](#). These assets were imported by the technical artists when an update was created.

6.3.5 Tutorials and Training

Tutorials were created to train the technical art team on workflows and tools. The pipeline's reliance on asset dependencies and complicated steps required workflows and tutorials to make training manageable. ISP training was a large investment due to the learning curve of new software and specific workflows. Seven weeks is not much time to train someone on technical art tasks and expect a large quantity of quality work to be produced. We found

training ISPs on one specific process and providing them the opportunity for greater repetitions was the most efficient learning strategy.

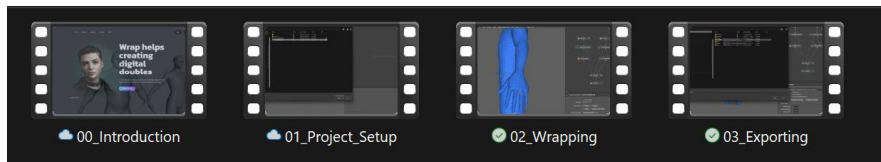


Figure 6.22: Training videos provided to technical art team

6.4 Clean Sweep Technical Art Pipeline

This was the final pipeline that we arrived at which supported transferring character models into game assets. The full list of pipelines and complete step-by-step instructions is available in the Clean_Sweep_Project found in Appendix K. The following sections serve as an overview of these instructions.

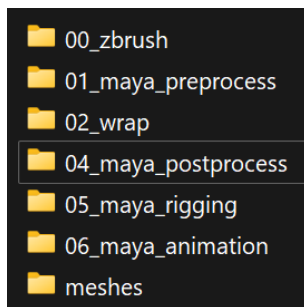


Figure 6.23: Pipeline workflow steps

6.4.1 Optimization and Retopology

When optimizing characters, we established a budget of 30,000 triangles. This aligns with a count similar to *Clean Sweeps* primary comparables: *Genshin Impact* and *Persona 5*. This count ensured that we maintained the initial silhouette and design of the artwork while keeping it manageable.

Zbrush preprocessing focused on exporting the low and high poly assets from *Zbrush*. The technical art department worked with the artists to establish best practices and guidelines. For us, this included using a naming convention for sub-tools and files, subdivision levels, primitives of certain densities, poly groups, and having a target polycount. These technical requirements allowed meshes to be processed more efficiently.

Preprocessing the files in *Maya* involved importing all the geometry from *Zbrush* and configuring transformation groups for the low and high poly assets. This allowed for easy reimporting and proper 1:1 scaling, saving time and avoiding mesh placement errors. Tools were later developed to expedite this process.

Wrap was used to transfer topology and UV data onto the character artwork, streamlining the rigging and retopologizing process.

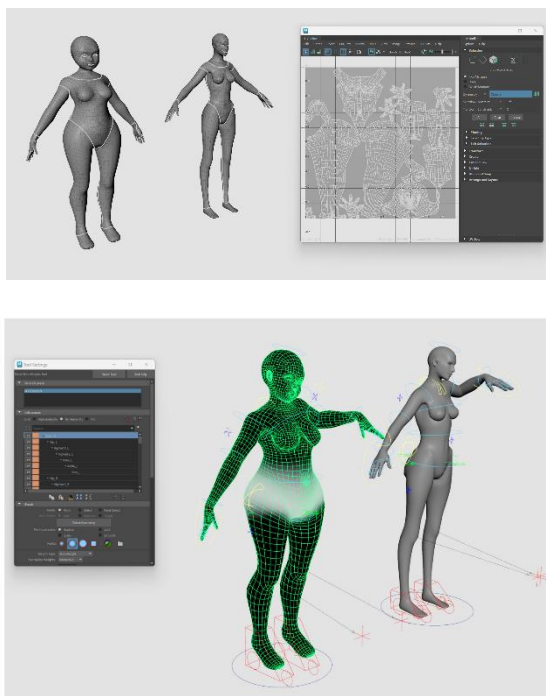


Figure 6.24: UV and Vertex Order Transfer allowing for skin weights transferred using NgSkinTools.

This node network allows Wrap to load geometry and match landmarks using splines. It ensures the geometry is symmetrical to avoid numerous errors and improve handling within 3D modeling packages. The model is then exported. With Wrap we now have the infrastructure to optimize and skin weight characters in hours rather than weeks.

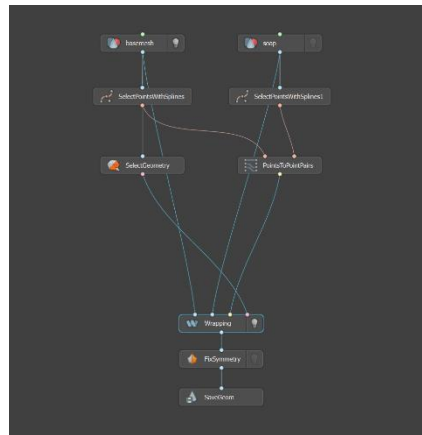


Figure 6.25: Wrap node network

6.4.2 Postprocessing

Postprocessing prepares the final model for texturing, rigging and animation. It involves optimizing the meshes, correcting topology from [Wrap](#), preparing 3-4 materials using vertex colors and creating UVs. Vertex colors are just another way to reduce texture memory and improve engine performance. Ngons and nonmanifold faces are addressed as well. (Refer to Fig: 6.4.3.3 for more clarification on polygon types.) The resulting models are ready for baking, texturing, and rigging.

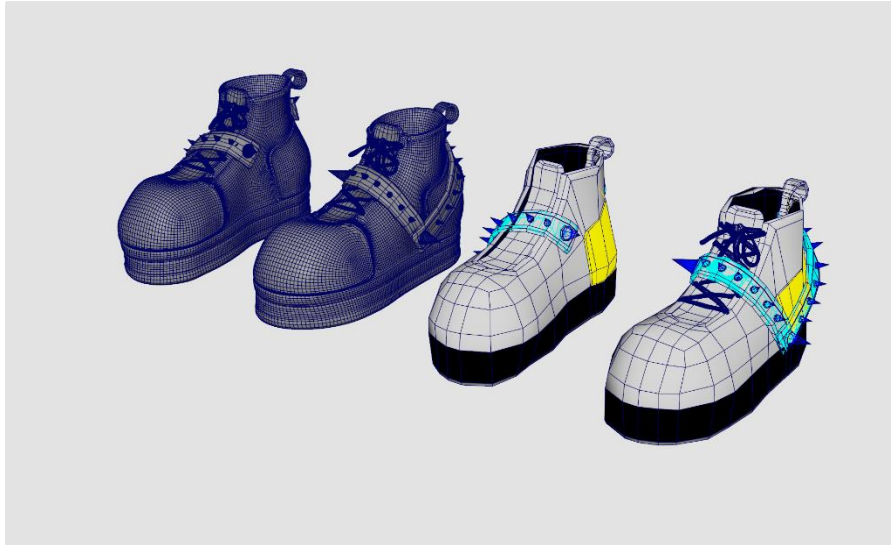


Figure 6.26: Optimized assets after assigning materials and vertex colors

6.4.3 Rigging

Rigging allows a mesh to be animated. It involves creating a system that drives the vertices of the mesh. The rig is a level of abstraction on top of the joint system that serves as an interface for animators. It has controls and other attributes for ease of use.

6.4.3.1 Iterative Rigging

We develop a strategy where each milestone in the rigging process is a discrete step that uses file references to reference the previous step. This allowed opportunities to revisit a step, make changes, and minimize the need for extensive rework. For example, if a joint needed to be added to the rig it could be done in step 02_skeleton which would not impact the mocap ready rig in step 05_anim_rig.

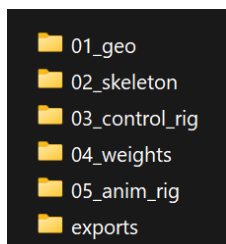


Figure 6.27: Iterative rigging workflow

The basics of rigging involve joints, constraints, parent child relationships, skinning, controls, connections, set driven keys, and more. Some understanding in these areas is necessary to add additional functionality to any auto rig in *Maya*. It is important to discuss the requirements of a rig with the design department before setting up custom systems. Creating a rig is like designing a product with stakeholders being the animators and game designers.

The *Advanced Skeleton* plugin was used to create the base character rigs. It enabled us to iterate quickly. Extra systems were added to the generated rigs like palm joints which allowed characters to interact with items using sockets in Unreal Engine.

Prop rigging required custom creation. In the Katana Weapon example seen in Figure 6.28, set driven keys, custom attributes, constraints, and a node network were used to drive behavior. This allowed the Windex Katana blade to extend from the bottle. Custom attributes were also created to allow the weapon to follow either the player's hand or hip.

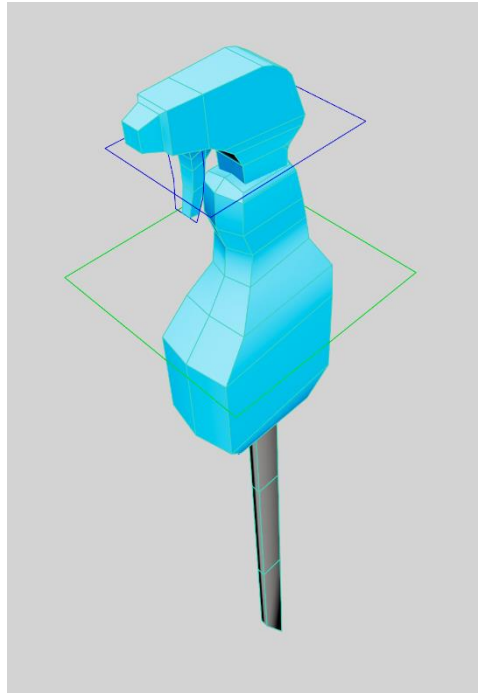


Figure 6.28: Windex Katana weapon rig

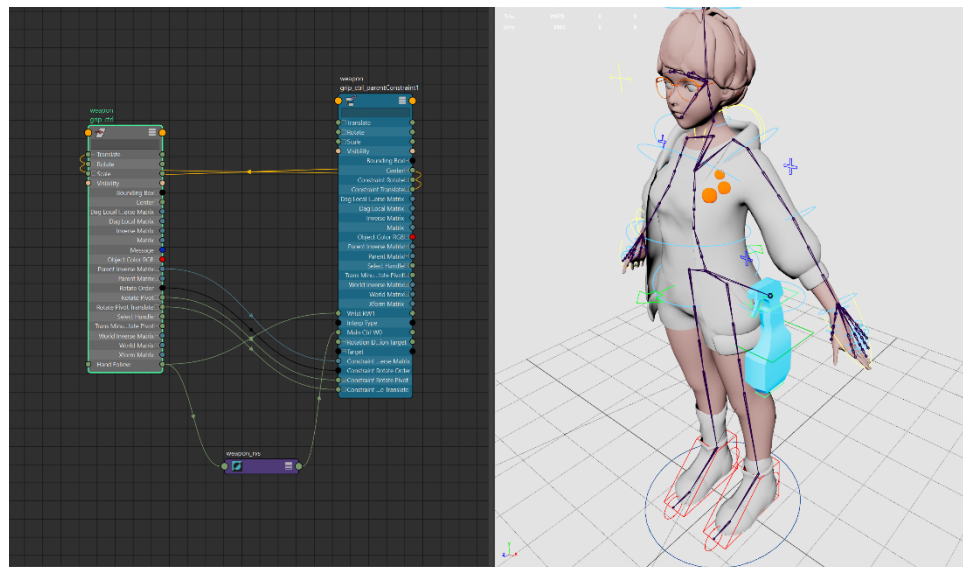


Figure 6.29: Rigging props with the Node Editor

6.4.4 Animation

Our animation process is tailored to transform motion capture data into stylized game animation. While motion capture, or mocap, accelerates the process, it can strip the appeal out of a performance. Animation fundamentals play a key role in enhancing mocap data. Given our scope, team size, and production schedule, implementing all of these practices was challenging. However, the full [Animation Workflow](#) was thoroughly researched and documented to support the future development of animations for *Clean Sweep*. Here is a brief overview of the critical steps we discerned:

Record Reference videos from the design department.

Record Mocap performance.

Retarget Data to create placeholder animation in engine as soon as possible.

Refine Animation in passes: Rough. Clean. Polished.

6.4.4.1 Animation Tooling

Animation tools encourage consistency and speed. Tools empower the animator to spend their time breathing life into static art rather than struggling with cumbersome steps. The following list were the most critical animation tools.



Sends the Character rig to the center of the scene using an animation layer.



Matches FK to IK or IK to FK for the selected limbs giving the animator more control



Allows the root of the Advanced Skeleton rig to follow for root motion for Unreal Engine



T-poses the Advanced Skeleton rig for faster HIK setup



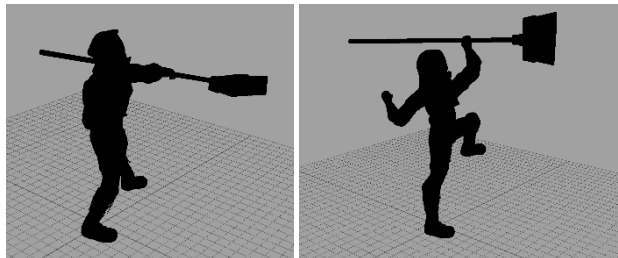
Generates animation layers for all the Advanced Skeleton limbs for faster cleanup

6.4.4.2 Mocap Cleanup

When cleaning up mocap data for stylized games it is important to balance appeal and exaggeration with responsiveness and input expectations for an engaging player experience. Because 'Goofy' and 'Thrilling' were our guiding design objectives, it was important to exaggerate the mocap performance. To do this, storytelling poses are keyed and pushed further to improve silhouette, clarity and impact using one animation layer. Clipping and other issues are addressed in four steps all executed in one animation layer:

1. Key the pose **before** it breaks.
2. Key the pose **when** it breaks.
3. Key the pose after it breaks.
4. Address the problematic key pose.

Without consulting the tools and workflows developed by the technical artists Victoria Rindeiko and Hanwen Xu from Django Unplugged, a robust animation pipeline for *Clean Sweep* would not exist.



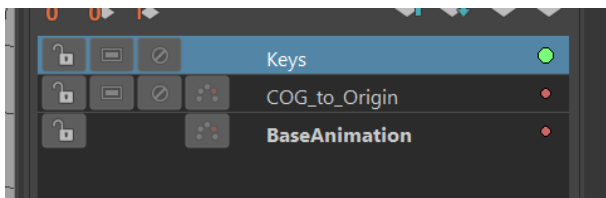


Figure 6.30: Using animation layers to improve silhouette

Timing is an essential part of animation for games and should be finalized before the animation is refined. When addressing the timing, the overall animation is sped up because realtime mocap data is not appealing on stylized characters. The anticipation, action, follow-through, and settle are specifically retimed to better accommodate player expectations. The anticipation is shortened for responsiveness of player input. Static holds are added to actions to clarify player actions. The actions themselves are made faster for more impact. Keys are manually cleaned up using the graph editor. Arcs are improved, limbs are broken, and in-betweens are addressed in the cleanup and polishing passes for more appeal.



Figure 6.31: Fine tuning action for stylized animation within *Maya's* time editor

6.4.4.3 Baking and Exporting

The *Game Exporter* was used to export character models and animation from *Maya* into Unreal Engine. The technical art department struggled with this process for several weeks and found it easier to first export the model with skin data and no animation. Then export the baked skeleton with the *Game Exporter's* export animation option.

6.4.5 Implementation

Within *Unreal Engine 5* there are two primary character importing steps. First, the character mesh and joints can be imported by checking the Skeletal Mesh and Import Mesh option boxes. Second, the animation data can be imported by selecting the appropriate Skeleton in the Mesh menu.

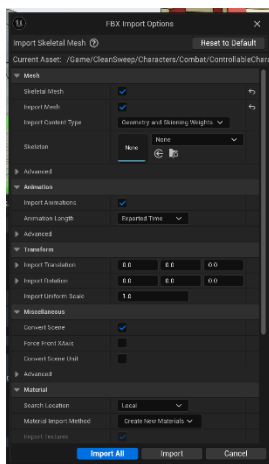


Figure 6.32: Model import settings in Unreal Engine

6.4.6 Recommendations

Develop a larger team of technical artists who can focus on specific aspects of the technical art pipeline. *Clean Sweep* only had one technical artist who, like many other members,

was split between different tasks including Animation, Rigging, Tooling, Scripting, and Optimization which caused bottlenecks.

To improve the optimization process, Professor Chery suggested supplying the artists with several game specific base meshes for specific body types that the artist can start with. This would have eliminated the need to use Wrap after each character was modeled. Instead, 4-5 distinct body types could be created. For future projects, we recommend this method.

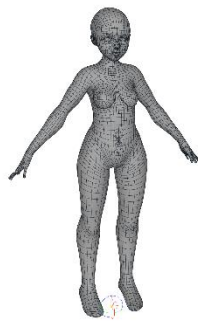


Figure 6.33: *Clean Sweep* female base mesh 02

Abandon perfection. The technical artists and pipelines should primarily focus on eliminating bottlenecks and encourage the flow of assets through departments as soon as possible even when they are not finalized.

7 Technical Implementation

7.1 Game Engines

The two main game engines that most 3D games are currently created with are *Unity*, and *Unreal Engine*. We deliberated months in advance to starting the project as to which game engine we would use. In the end, we chose *Unreal Engine* for the development of our game. *Unreal Engine* is known for its realistic graphics, major 3D support, versatility for teams small and large, visual scripting, and its wide arsenal of tools. *Unreal Engine* is primarily a game engine, supporting numerous built-in tools and plugins that are specific to game development. This includes classes like game modes, player controllers, game states, levels, and game instances with various plugins like *Niagara* which helps develop stunning visual effects and particle emitters, *Common UI* which adds support for development on various consoles and controllers, and Enhanced Input which allows for easy creation of actions that are controlled by various inputs from various devices. Unreal Engine also features a robust marketplace filled with various plugins and asset packs for every aspect of game development one can think of.

Unity, while being a go-to for many projects, but didn't quite meet our needs for developing *Clean Sweep*. While *Unity* supports both 2D and 3D game development, it primarily supports the development of 2D games for PC and mobile. Meanwhile, *Unreal Engine* has focused all its effort on 3D, offering high quality features that *Unity* lacks. For example, many classes like game modes and player controllers are built directly into *Unreal* but need to be developed by hand

in *Unity*. Additionally, *Unity* is written entirely in *C#*, and, while it does have visual scripting, it is not as user friendly and easy to use as *Blueprints*, *Unreal's* visual scripting language.

Overall, *Unreal Engine* was the right choice. It allowed for a quicker, easier development experience that *Unity* would not have been able to support. For any future developer thinking of making a 3D game, *Unreal Engine* is great for rapid development, scalable projects, and immense 3D support. The only downside is the lack of documentation. *Unreal Engine* has a lot of powerful tools, all without much documentation. To combat this, a lot of core systems that we created along with engine provided tools we used had documentation written by the programming team since there wasn't much support online. However, with an increasing number of developers using *Unreal* over the past year, this problem has improved and is expected to continue getting better.

7.2 Source Control

When deciding on source control, we had three main criteria. The first was ease of use. The second was familiarity and comfort level with the chosen source control. The third was how steep the learning curve for other departments is. Our deliberation led us to two primary options: *Perforce* and *Git*. Ultimately, we chose *Git*.

Git is a source control option that both the tech and art team were familiar with due to IMGD 4000/4500. We hosted our project on *Github*, a website to host applications made with *Git*. Using *Github Desktop*, a simplified and intuitive desktop application of *Git* that allows for using *Git* without knowing commands, *Git* became significantly easier to use. With great ease of use, familiarity and comfort, and there being no learning curve since the entire team knew how to use it, *Git* became the clear winner; it did come with its fair share of cons, however. First, *Git* limits the maximum file size that can be pushed to a repository. Pushing is when someone merges their own

code with the code project online. With our game being built in Unreal Engine, the potential for large binary files left us concerned with pushing files to the repository. Second, Git Large File Storage (LFS) would be costly. LFS allowed for large files to be pushed to the repository, but it has a limit on both size and number of files that can be pushed for free. Third, unlike *Perforce*, *Git* does not have file locking capabilities. This would result in, especially during early development with a small project, programmers, designers, and artists not being able to edit the same scene or blueprints in tandem. This would bring about issues of occasional merge conflicts we would need to address.

While we did not choose *Perforce*, it was heavily considered. *Perforce* is regarded as the industry standard for *Unreal Engine* and has many features including handling large binary files, great security, file locking and checking out, easy scalability, and easy collaboration with incredible speed. WPI does have a tutorial on the IMGD hub for setting up a *Perforce* server, and it seemed like the perfect option to use; however, we were not familiar with the software, and knew that the source control would have a steep learning curve for not just our team but everyone who would join the project.

While *Perforce* seems like the ideal option and a fantastic opportunity to familiarize ourselves with the industry standard, our lack of familiarity presented a significant hurdle. Additionally, *Perforce* is only free for up to 5 users. With more people joining the team through ISPs and the art team needing access to the repository, this made *Perforce* less desirable. Moreover, while WPI granted us free access to *Perforce*, if we wanted to make further changes to the repository after graduation, the project would be stored on a WPI server, becoming

inaccessible for future updates. *Git* provided the easy accessibility that our entire team would need for future development of our game.

7.3 Plugins

When developing *Clean Sweep*, we did not hesitate to use plugins. A plugin is a tool or piece of software developed to aid with using the engine. Rather than creating many core systems and assets from scratch, we found it most efficient to look online for existing solutions. We wanted to utilize every resource available when developing. Below, we outline some of the plugins we used over the course of development.

7.3.1 *Blueprint Assist*

Blueprint Assist was by far one of the most useful plugins we used. *Blueprint Assist* offers many features including, but not limited to, auto formatting blueprint nodes for increased readability, various useful key-binds for numerous functions like inserting nodes between others, and full keyboard programming support in blueprints for faster development. At the click of a button, entire graphs were formatted and organized. This streamlined the development process considerably. We would recommend this plugin to anyone seeking to program in Unreal Engine.

7.3.2 *Auto Size Comments*

Auto Size Comments was a simple plugin that allowed for comments to be appropriately sized to the nodes it encapsulated. It gave easier access to comment colors, tracked, and moved with nodes (which worked very well with *Blueprint Assist*), and allowed for better readability of

comments allowing for better code documentation. This plugin is another must have for any *Unreal* project.

7.3.3 Global Event System

Global Event System (GES) was a decoupled event system that allowed actors to send events to each other without needing a direct connection to others. Essentially, we could have one actor send information to another without needing to know if the other actor existed. This was extremely useful for our project as some systems, like the quest system, needed to receive information from various parts of the game, including the time system, enemies, dialogue, and items. With the GES, any actor could broadcast a message with a gameplay tag and data, and any actor could bind a function to whenever that gameplay tag was broadcast. This was useful for a number of reasons including our quest system which needed to listen to events from a variety of actors and binding every single one to it would be a hassle. More information on the gameplay tag system can be found below in the [Quest System](#) section.

As useful as this system was, it proved to be tedious and cumbersome for expansion. Binding functions for each individual tag made the binding function long and confusing. It made expanding the system annoying as we'd need to both add a new function for each event and a new node for binding. If any new tag event were added, it easily could be forgotten and not added to the quest system, causing more of a headache and hassle to update.

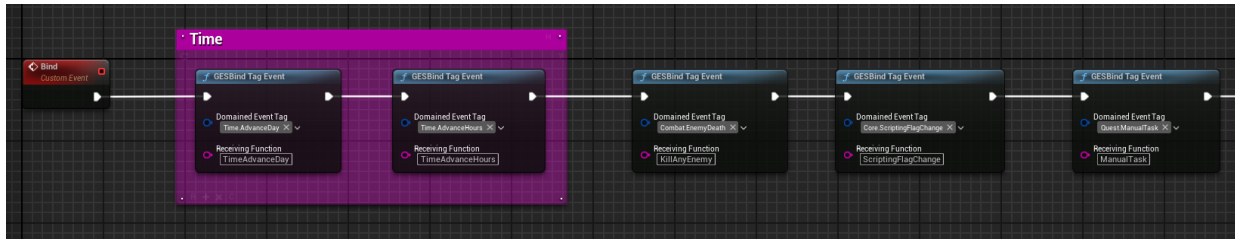


Figure 7.1: The bind function that bound new gameplay tag events to their respective functions

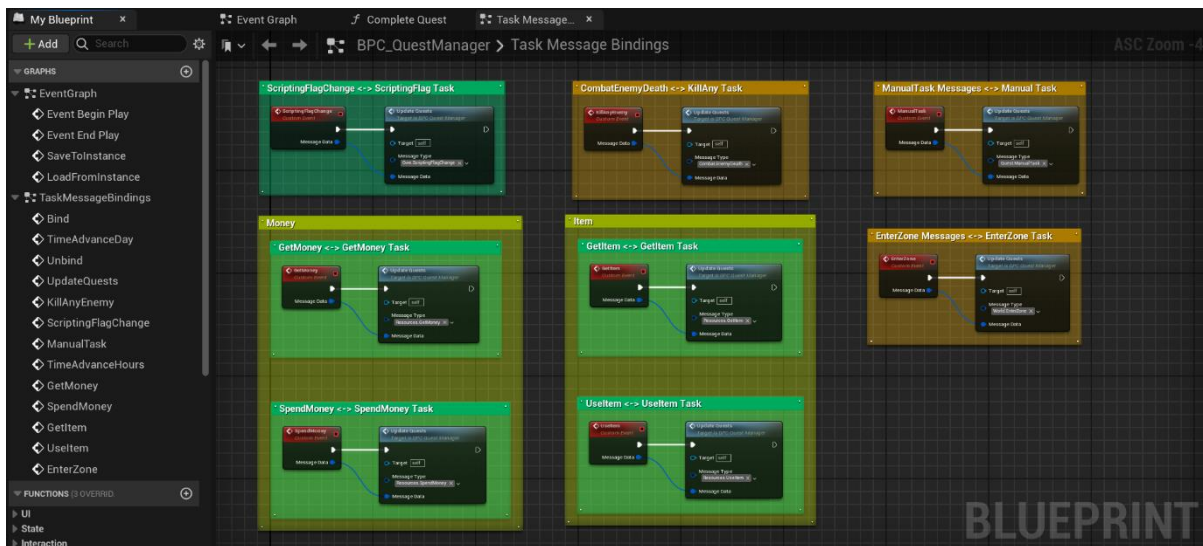


Figure 7.2: The functions each gameplay tag event is bound to.

As poor as using this plugin was, it worked for the most part; that was until we started to build and package the project. GES introduced a new crash bug that occurred when events were sent. The bug's source was initially undetected, but after numerous days of debugging, we discovered this plugin was the issue and were unfortunately forced to remove it from our project. This bug was the same bug that prevented us from getting selected for PAX. With numerous systems relying on this plugin, we were forced to find or create an alternative that provided similar functionality. With the amount of time remaining on the project for the year and the complexity

of developing our own, we decided to find one online. We ended up finding and selecting the plugin *TMW Event Aggregator*.

7.3.4 *TMW Event Aggregator*

TMW Event Aggregator was another decoupled event system that provided the same functionality as GES with some added benefits. It ran on its own subsystem which was much more promising than the GES. It allowed for events to be bound not only with gameplay tags, but also with classes and names, a type of string.

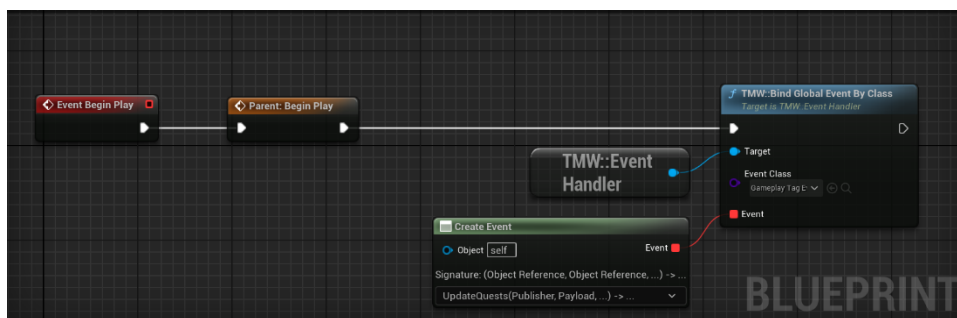


Figure 7.3: *TMW Event Aggregator's* event binding function.

We created a custom class that included the gameplay tag and data we wanted to pass. One of the limitations of the old event system was that we could only send one struct worth of data. With the new plugin, we can listen for both when the class is sent and the data within that class. In direct contrast to GES, *TMW Event Aggregator* allowed for an easily expandable system. It also allowed for the creation of metadata to pass along with the events so we could have better debug messages of who sent a message and what the data sent was. While the plugin

does cost money, it is worthwhile for any large game that needs many actors to communicate without the need for them to bind to each other directly.

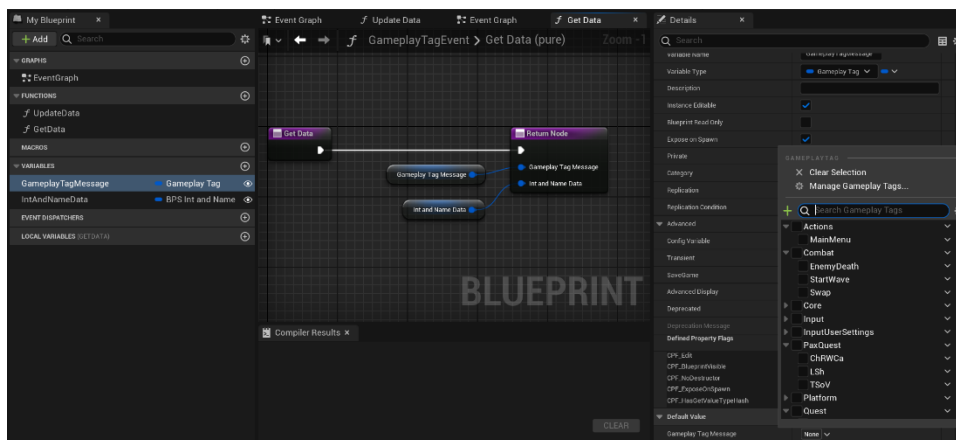


Figure 7.4: The *TWM Event Aggregator* class used to send data between actors.

7.3.5 Dialogue System X and Inkpot

Inkpot is the foundation for our dialogue system. It was also the second dialogue system we worked with. Our initial dialogue system, *Dialogue System X*, was extremely easy to work with and set up, but lacked expandability. It didn't allow for numerous features that would be necessary in a dialogue system like on screen animations, interacting with other in-game systems, or remembering what had previously been done or said. Trying to import any of these features was an intense amount of work and effort that would be cumbersome overall. Creating dialogue files turned into an implementation nightmare with both creating the files and associating them with NPCs. We wanted to turn to a new dialogue system, but there wasn't much we could do. Most dialogue systems in the marketplace had an expensive cost and didn't seem to offer the features

we desired. It was initially decided that we would create our own until we ported the project to *Unreal Engine* version 5.3.

Upon porting the project to 5.3, we found a new option at our feet. We had initially discovered the plugin *Inkpot* through one of our advisors. Unfortunately, the plugin was for 5.2, and our project was in *Unreal Engine* 5.2 with no plan of porting up. With numerous errors and crashes resulting from the sequencer, a built-in engine tool for in-game creating animation sequences, we began looking into porting our project to 5.3 as a solution. While porting to 5.3 solved our problem on the sequencer, it also allowed us to use the *Inkpot* plugin which was only available in version 5.3.

Inkpot has been fantastic to use and work with. *Inkpot* is based on the narrative scripting language, *Ink*. *Ink* allows for easy to create and use dialogue. The formatting of *Ink* made it easy for writers to directly write in the .ink file format without many errors and for other writers to read and understand the dialogue. *Ink* also allowed for functions, variables, and conditional statements directly within the file for logic to be implemented without needing to process it in *Unreal Engine*. *Dialogue System X* had some of these features, but it was very primitive compared to *Ink's* capabilities. *Inkpot* is a definite must have for any project using a dialogue system.

7.3.6 LE Extended Standard Library

LE Extended Standard Library is an extensive blueprint library with numerous useful functions. The functions include string modification, custom delays, creating local variables in event graphs, latent actions, queues, flow control macros, and getting info on platform and

build. The *LE Extended Standard Library* is a great free plugin with a lot of useful functions and macros that should be used in any *Unreal Engine* project.

7.3.7 Runtime Audio Importer

Runtime Audio Importer allows for audio to be imported, recorded, exported, and streamed at runtime. We had initially used the plugin for playing dialogue voice lines and sound effects at runtime. One big issue we found out was that the plugin tried to find the file to play the audio by its absolute location. When building the project, the game files get compressed into .ucas, .pak, and .utoc file formats which do not have the correct path. The engine allows for virtualizing a file path within blueprint functions, but this fails to properly work as runtime audio importer does not have the capability to do this. The way to fix this issue is to pre-import audio so it can be found in the engine's virtual file system; unfortunately, after pre-importing the wave files, Runtime Audio Importer could not read the files. While this plugin has its uses, it fails to work for a built application. While it can work for one where all files are not compressed or packaged, that allows for others who obtain the *Clean Sweep* download to have direct access to all files in the game.

7.4 Levels

Within the project we utilized multiple levels. A level is a collection of actors, landscapes, and lights that make up the world. They are useful for organizing and categorizing various parts of the game. We created a level for each section of the town. Two questions we had to find out the answer to were how to switch between levels, and how to maximize performance within each level. For maximizing performance, we debated on whether the interiors of buildings would be

modeled directly or if the interiors would be their own level. To minimize the number of polygons on the screen for the player to render, we decided to make each closed interior have the interiors as their own level. With that solved, we also needed to worry about the former problem, how do we switch between levels.

To switch between levels, we created a custom actor called BP_LevelLoader. This actor was given a level to open and the display name of the level. When players overlapped with the actor, a UI widget appeared on the screen that said, "Press A to travel to Location," with "A" being the interact key for the active input and "Location" being the display name of the level. Upon interacting, the player opens the level and spawns in the player start location within that level. "Player Start" is an actor in Unreal Engine that tells the engine where to spawn a player.

Another issue that we considered was how do we limit the number of actors loaded into the level. This became a severe problem when trying to push the levels to *GitHub*. *GitHub* allowed for a maximum file size of 100 MiB to be pushed. With levels becoming larger and larger with more actors placed on them, their file size eventually broke the 100 MiB limit. One way to fix this was utilizing *Unreal Engine's* level streaming.

Level streaming opened endless possibilities. It allowed us to segment levels into their bare essentials like lighting, foliage, building, obstacles, and other actors. It allowed a minimum file size while also allowing users to work on the same map at the same time. Since Unreal Engine uses binary files, it is impossible to merge them on *GitHub*. With level streaming, one developer could work on a map and modify the buildings while another developer could work on the lighting without interfering with the first. This was perfect for lighting as we could bake multiple lightings into distinct levels and achieve a day, evening, and night lighting.

The mansion severely required optimization. Wanting multiple rooms and the ability to seamlessly walk between rooms and floors of the mansion, we needed the ability to selectively load certain parts of the map. Level streaming gave us that ability. We created a custom actor called BP_Streamer. This actor was responsible for loading and unloading certain levels based on where the player was walking. This allowed for not only the mansion to have seamless loading between rooms but also any interior building to seamlessly load and unload the other portions of the world. In the example below, we loaded the room the player came from and the room they were going to so they would always have visibility and quick loading between the rooms they just entered and came from.



Figure 7.5: An early prototype map of the mansion that utilized level streaming with each room being its own sublevel.

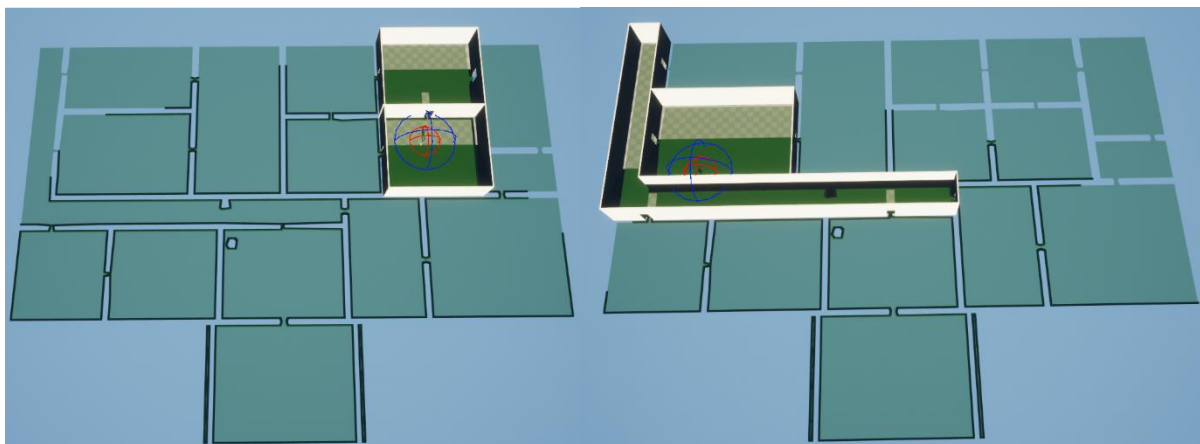


Figure 7.6: The level streaming of the prototype mansion with levels being loaded based on what rooms the Janitor had previously entered from.

7.5 Combat

7.5.1 Hitbox System

The hitbox system was one of the first systems we developed. As such, it needed to be adaptable to animations as they were integrated, rather than being individually programmed for each animation. To accomplish this, we developed capsule-shaped hitbox actors that attach to the joints of an attacker's skeleton. This design allows them to move in sync with the animation, resulting in hit detection that matches the animation's movements.

Hitboxes were given a plethora of controls and settings to ensure that they were flexible enough for our game's combat needs. The first of these are settings for the hitbox capsule's half-height, radius, rotation, and offset from the socket. These controls functionally allow the positioning of a hitbox at any size and location, making them versatile enough to cover the space needed for any animation. Next, each hitbox is assigned an ID to ensure that entities recognize it

and avoid being affected by it repeatedly. If multiple hitboxes are intended to combine into a larger, more complex shape, a Coexist ID can be used, which enables the hitboxes to group together and function as a single unit. Lastly, each hitbox contains all the necessary data to determine its effects upon landing a hit. This data includes the damage inflicted, the duration of hitstop and hitstun, and the strength of the knockback. knockback can be further customized with controls for adjusting the angle and selecting between forward or radial knockback directions.

To make implementation of hitboxes easier and more accurate, we implemented a feature in the engine that visually represents capsule hitboxes as they would be positioned in the game. This visualization is achieved by calculating the position of a debug capsule in each frame and drawing it within the *Unreal Engine* animation previewer. This allows developers to see and adjust the hitboxes in real-time, ensuring they align correctly with the character animations.

When a hitbox overlaps with the collider of an entity that can take damage, it activates the receive damage response, which includes applying damage, knockback, hitstun, and various effects. The amount of damage dealt is calculated by considering the hitbox's base damage, the attacker's strength, the enemy's defense, plus the chance for a critical hit. Both knockback and hitstun effects are determined by the values set on the hitbox. Also, any effects that the hitbox carries, like slowness or poison, are also applied to the target.

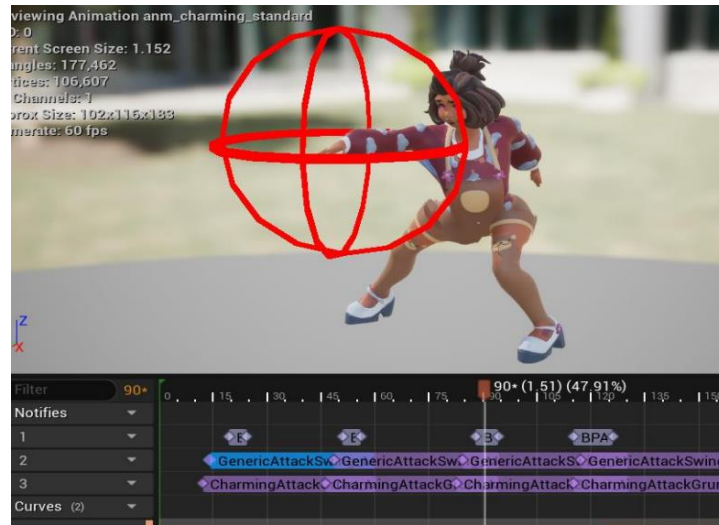


Figure 7.7: Capsule hitbox attached to Charming’s palm socket and drawn with a debug capsule

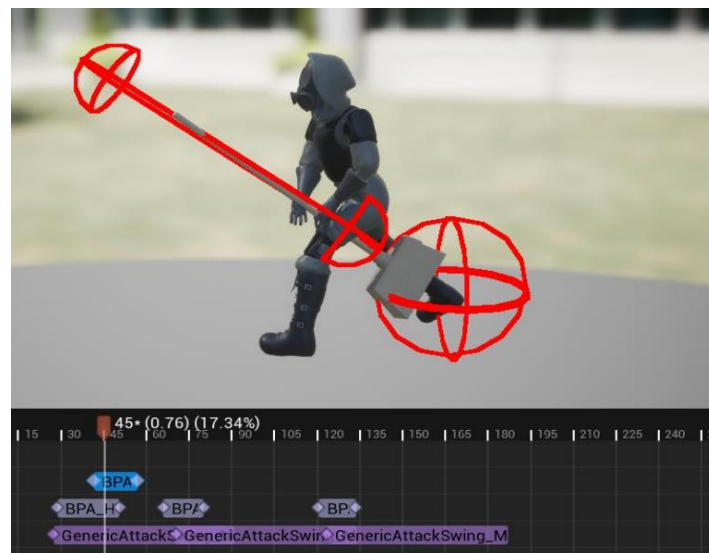


Figure 7.8: Group of multiple hitboxes using Coexist IDs to make the shape of a broom

7.5.2 Basic Character Function

Characters inherited logic from a base character class, simplifying the development process by establishing a consistent framework for all character behaviors. This inheritance

allowed us to implement shared functionalities, such as health management, hit behavior, and movement controls across all characters without redundant coding. Character movement was based on *Unreal's* character movement component, which provides a basic control system for moving humanoid characters. Features such as setting the walk speed, acceleration rate, and rotation rate of the character helped us to quickly dial in the feel of the game's movement and helped speed up development early on. Characters utilize a shared base animation state machine, which manages their animation transitions between idle, walking, and running states. Animations not handled by the state machine are handled programmatically using animation montages.

7.5.2.1 Combo Attacks and Dodges

Two of the significant character actions programmed using animation montages were standard combo attacks and dodging. This approach provided the necessary flexibility to allow these actions to end earlier or later based on player inputs. Using montages for the combo system, we could insert markers at the start of each hit within the combo. This setup enabled us to monitor for inputs to determine if the animation should continue, rather than using multiple separate animations.

For player dodges, we implemented an 8-directional system based on the player's rotation. Using animation montages, we streamlined the process by supplying the appropriate dodge animation based on player input logic. This method replaced what would have been a complex 8-node dodge state machine, making the implementation less cumbersome. Additionally, montages grant the flexibility to cancel the dodge animation midway should an attack be input by the player.

7.5.3 Playable Characters

The game currently features three playable characters, each equipped with unique moves and passive abilities. While the code for each character's standard attack is identical, they each have a customized velocity timeline that syncs their movement with the animation of the standard attack. Beyond this commonality, the functionalities of each character are entirely distinct. These unique functions are typically created using a combination of animation montages and special sequences of logic. Custom functions can be linked to a hitbox's overlap event, allowing for more tailored effects upon hitting an enemy. Some more complex cases include attaching germs to the root socket of Charming's skeleton to simulate her picking up and slamming down enemies and implementing a spline path for the Janitor's boomerang to create a customizable trajectory for their hold standard attack.

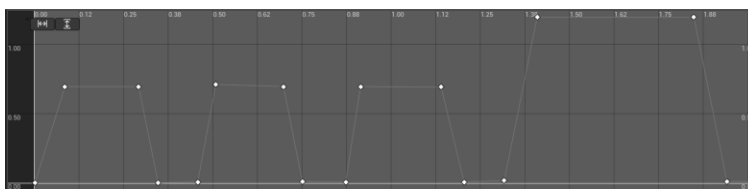


Figure 7.9: Custom velocity timeline for janitor standard attack

7.5.4 Inputs

We managed game inputs using *Unreal Engine's Enhanced Input System*, which simplifies the creation of actions and their association with diverse types of inputs. This feature was particularly beneficial as we developed the game for both keyboard and controller use. Early in the development process, it also enabled us to experiment with several types of input triggers, such as tapping versus holding buttons. The *Enhanced Input System* also simplified the creation of

an input settings menu, allowing players to customize their control bindings for greater comfort or to meet accessibility needs. Along with these inputs, we also implemented a custom buffering system which stores when an attack was buffered and executes them in order of priority once the current action is complete.

7.6 Enemies

7.6.1 Enemy AI and Token System

When developing the AI of the enemies, or germs, in our game, we wanted to make sure that they would be dynamic and fun to fight. The germs went through numerous iterations before we were able to completely settle on an AI design.

When developing an AI, we had to contemplate what would be the best approach to designing the AI. The two main options that surfaced were utilizing a behavior tree or a state machine. We ended up choosing behavior trees due to the several benefits of them and because *Unreal Engine* made creating them easier. With a behavior tree, actions are selected from a tree going from left to right. It reads a node, or task, on the left side of the tree and checks if it will succeed. If it fails or the tree cannot reach that node, it returns and tries to execute the next branch. This would allow easier task management and reuse of tasks for various germs. It also allowed for easier viewing of what actions a germ could take due to any given conditions. The behavior tree allowed for more complex logic and patterns that could easily be updated and reused among other germs.

On the other hand, state machines would be able to have germs enter different “states” that would dictate various behaviors. We could have a germ choose to go from one state to another based on different rules. For example, we could change the germ from a wondering state to a pursuit state when seeing the player. Moving from one state to another would be based on a set of rules that we implement. While state machines have more modular states, the task system that behavior trees provide are far more universal. With *Unreal Engine* having behavior trees built into the engine, working with them was much easier and allowed for easier visual representation of what our germs were doing.

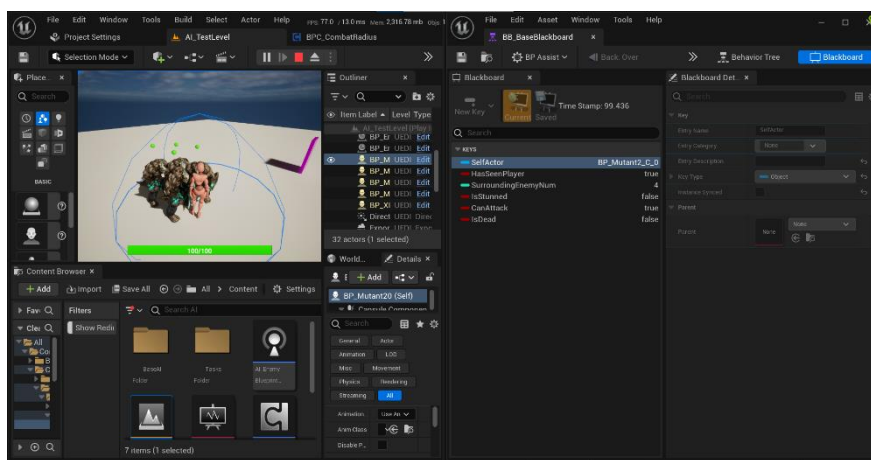


Figure 7.10: An early prototype of our AI Blackboard.

To keep track of variables in a behavior tree, a blackboard is used. A blackboard is a container for holding numerous variables that each germ can individually use as well as having the option for shared variables among every enemy using the blackboard. In the figure above, we created a simple blackboard and behavior tree. This blackboard holds onto variables for different enemy conditions like if they have seen the player, if they are stunned, or if they are dead, and it

also holds on to how many enemies are currently surrounding the player which is shared among all germs. We used that value to communicate how many germs should attack; we wanted to limit the number of germs that could attack the player at a time. Organizing every germ in the game to coordinate their attacks would prove to be a monumental task. We had to consider how all germs would know when to attack, if they could attack, and if they should give up on attacking. We also had to consider what types of germs should be allowed to attack and when. This is when we created the AI Token System.

The token system we developed was quite simple. We researched and modeled it after *Marvel's Spider-Man's* token system ("The AI of Marvel's Spider-Man" 00:11:35 – 00:15:07). There were managers for each class of germ: melee, ranged, and explosion. Each manager has the same functions with each manager implementing different versions of that function. There are functions for requesting tokens, returning tokens, and generating additional tokens; there are also variables for the tokens themselves, the intensity of the enemies, and if an additional token has been generated. To explain the system, let us analyze what a token is.

A token is a simple structure, or struct, that contains 4 key pieces of information: an ID, if it is checked out, if it was stolen, and the owner of the token. Those key bits of information tell us everything we need to know. For an enemy to attack, they need a token. An enemy will request a token, attack, return it, and be on a cooldown from requesting tokens. Enemies can only request tokens when they have seen the player, are actively trying to attack them, and are close enough for the germ to reasonably get to the player and attack. Also, they can only request one token once per second. When an enemy requests a token and successfully receives one, the token is checked out. The manager knows who owns the token and can interact with the germ.

Tokens, as said, also keep track of if the token was stolen. How does an enemy steal a token from another enemy? There are two main ways an enemy can steal a token. The first is based on the player. If a player walks close to an enemy, instead of standing and staring at the player, provided there are no available tokens, the enemy will steal a token from another enemy who has one. That said, there are certain rules. A token cannot be stolen from a token that was already stolen. This limits the number of stolen tokens as the player can run back and forth between enemies and have them constantly steal tokens without fully attacking. The second rule is that a germ can't steal a token if they already have a token. That way no germ ends up with multiple tokens or stealing tokens from themselves. Finally, if a germ is stealing a token, it will always generate an extra token. This way, all germs will not suddenly stop attacking when another germ tries to steal a token.

The second way to steal a token is based on distance. If a germ attempts to request a token from a germ that is a large distance away, the closest germ will get priority and steal a token, leaving further germs without the ability to request tokens. Any time a token is stolen, it always prioritizes the furthest enemy from the player. Each manager keeps track of these rules and conditions. The way extra tokens are generated when a germ is stealing are also generated naturally. The intensity of the manager is dictated by the intensity variable. The longer the player stays on a single character, the more the intensity increases. This makes the game harder if you swap less frequently. The higher the intensity, the bigger the probability of generating a new token. Each second, a new token is calculated by rounding down the intensity to the nearest whole number as the number of tokens available and taking the float remainder and making that the percent chance to calculate a new token for that second. For example, If the intensity is 2.5, two

7.7 NPCs

With *Clean Sweep* being a narrative driven game, every NPC had a couple of key requirements. They needed to be interactable, they needed to move during various times of the day, they needed to be able to open shops if applicable, they needed to have animations, and they needed to have varying dialogue depending on current player quests, scripting flags, and time. While most of the varying dialogue was solved due to the dialogue system, more information can be found below in the Dialogue System section, the others had to be done through our NPC system. We created a custom data asset to store all our NPC data. Each NPC had a name, animation blueprint, skeletal mesh, dialogue pool, daily schedule, and shop if needed. The data asset currently contains three functions as well. This includes getting dialogue, getting the pooled dialogue, and getting the daily schedule.

7.7.1 Schedules and Spawning

NPCs had schedules to control what level they would load into, what location they would spawn at, and what time they would appear. Schedules were a simple data table that could be imported via a spreadsheet to allow for easier additions of schedules for writers to create. Each character had a schedule for different days of the week; we had plans for expanding the system to allow for various locations depending on certain scripting flags. The NPC Manager we created stored the NPC data assets and would iterate through them to determine if the NPC would spawn. In conjunction with our Time Manager, the NPC Manager would check the day of the week and the current time to see if each NPC had a location set during the scheduled time that matched the current level and the location. Target points were used to determine the exact location the NPC

would spawn, with the names of the target points being the same as the names of the locations NPCs would store in their schedule.

7.7.2 Shops

Certain NPCs were needed to open shops. Shops were stored in the NPC data asset as their own type of data asset. The data asset contained the name of the shop, tagline, and a struct called `BPS_ItemsAndConditions`. This struct stored an item and an event condition. An event condition was another struct created for quests to set conditions for quests to be given out. This allowed for custom items to be sold when certain conditions are fulfilled or for certain items to be sold during specific quests being completed or in progress.

The Shop Manager was responsible for taking in the currently speaking NPC's shop data, opening the shop via dialogue, closing the shop, and unbinding it to dialogue. Any time the player approached an NPC, their shop data updated the shop manager. When the NPC's dialogue invoked the function `OpenShop()`, it created the shop widget and loaded the shop data into the widget for players to buy items. The shop widget checked for which items could be sold, displayed the ones that could, displayed the name and tagline of the shop, and displayed the total amount of money the player has. The shops had logic for determining if the player could afford to buy an item and limiting the max number of items a player could buy at once.

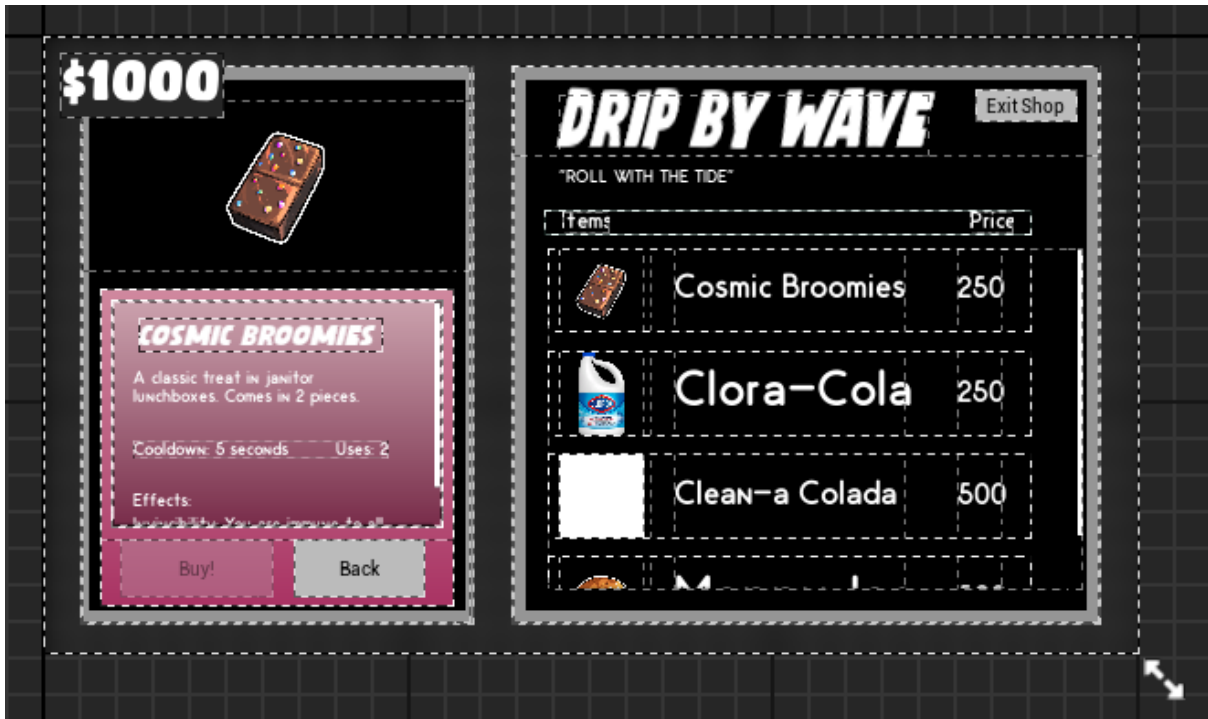


Figure 7.12: The widget designer of shops with Wave's shop data loaded into it.

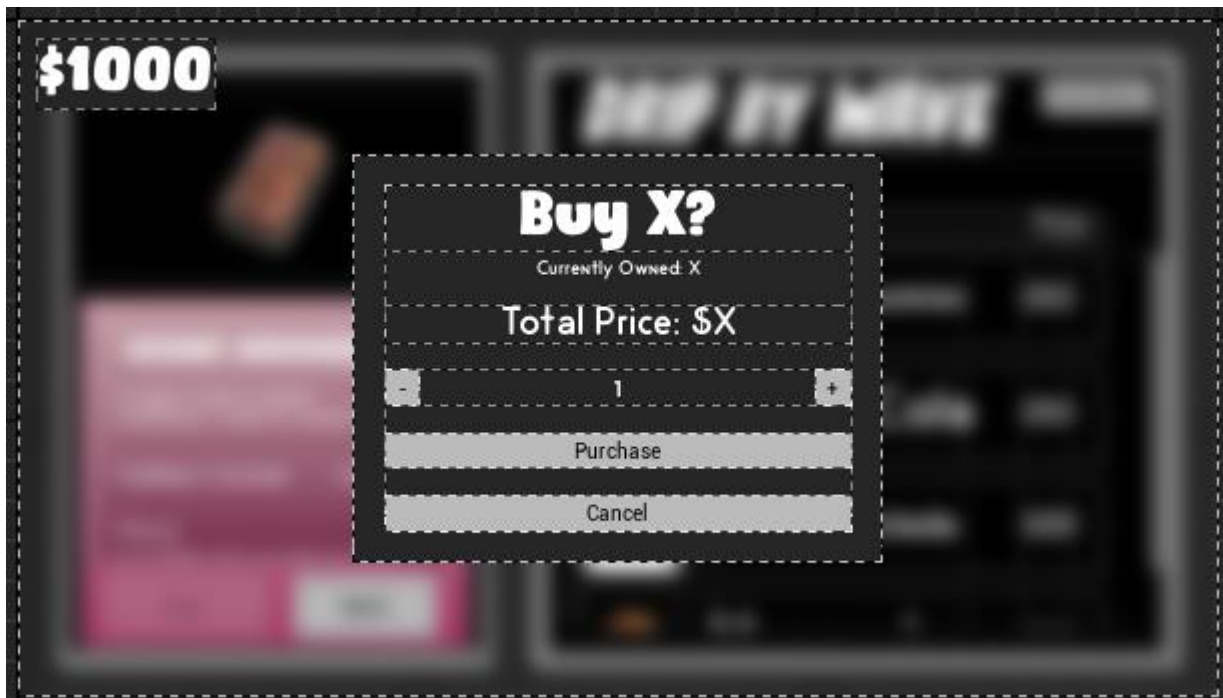


Figure 7.13: The shop purchase item pop-up in the widget designer.

7.8 UI

Developing the UI within *Clean Sweep* turned out to be a monumental task. Mocking up, developing, iterating, and programming UI was a lengthy process that required hours of work from multiple departments. Development time in *Unreal's* widget designer took a long time with poor design patterns on the tech team. Unreal UI creation was a confusing process that took a while to understand, so we turned to many *YouTube* videos that covered *Unreal Engine* UI creation. One of the main ones we utilized was called "Advanced UI Templating Techniques using Widget Blueprints and Materials | Unreal Fest 2022." This video was created by the *Unreal Engine* team and gave a lot of insight into widget design patterns like parenting widgets, named slots, and parametrized widget properties that accelerated our process. However, even with this new knowledge, we found ourselves at a roadblock. Controller support was challenging to do in *Unreal Engine*, and we needed to find a solution. Luckily, *Unreal Engine* created a plugin to alleviate the concerns of developing for various controllers, *Common UI*.

7.8.1 Common UI

Common UI is a built-in plugin from *Unreal Engine* that allows for easy development of navigable UI, UI support for multiple consoles, and better built in UI widgets. *Common UI* gave support for controller buttons to be mapped to images, images to update based on what controller was in use, and common styles for various widgets to use. We heavily utilized these features to make *Clean Sweep* as easy to develop and use as possible. The most useful addition from *Common UI* was styles. Buttons, text, and borders can create custom styles.

A style is a collection of data that can easily be added to a widget that gives info on various aspects of their design. For example, a button style has information on what sounds should be played based on different actions performed, assorted colors for varying states the button can be in, and size information to say how big the button will or can be. A style is incredibly useful as this information can be easily applied to multiple assets via a variable. All common text widgets, buttons, and borders have this variable to allow for easy modification. It also allows for buttons with similar styles to all be updated simultaneously when one style is updated. Using this, we gave placeholder styles to a variety of buttons so when we eventually update the button styles, all buttons will be updated. There will not be a need to individually go into every single place a button is used and change every aspect of that button.

Common UI also has another useful feature, and that is the `CommonInputAction` widget. This widget allows for inputs to be updated based on the currently used input by the player. This means that when a keyboard is in use, whatever input the player would need to use will show up on screen. The `CommonInputAction` also allows us to add custom images to define each input the player can perform. We created a custom data table for *Common UI* that links the inputs to the images. As pictured below, we use an A button for controllers while using an image of an E key on keyboard. This gave us even more control over allowing mass replacement of buttons. If we want to change the keyboard and mouse and controller images, we could very easily swap it out in the table *Common UI* uses to map the inputs to images and update all uses of that input simultaneously.



Figure 7.14: CommonInputActions being used to change the interact key for controllers and keyboard.

On the left is the controller being the active input and on the right is keyboard and mouse being the active input.

These features in *Common UI* that we utilized allow us to confidently recommend this plugin to everyone in the future developing a game in *Unreal Engine* with interactive menus. There are a lot more features of *Common UI* that we utilized, but the ones covered were the most important. For more information on *Common UI* including our self-made documentation on it, please check out Appendix J.

7.9 Dialogue System

Our dialogue system went through the most iterations and revisions compared to every other system in our game. With such a big focus on dialogue and trying many different plugins and solutions to creating our dialogue system in the engine, we spent months trying to perfect it. As previously stated in section [7.3.5](#), dialogue went through multiple plugins and states before we decided to use Inkpot. Inkpot allows for in-text logic, variables, and functions with the ability to call functions within blueprints. Utilizing these features along with its easy to import and update assets, we created a robust dialogue system.

7.9.1 Dialogue Widget and Logic

Most of the logic within the dialogue system exists within the dialogue widget named CAW_Dialogue. This widget is responsible for linking to the ink subsystem, linking all functions related to ink dialogue flow, and performing numerous functions for both inside and outside of the widget itself. Each NPC is equipped with a dialogue player that creates this widget when interacted with and starts the story with their *Ink* story asset pulled from their data. Once CAW_Dialogue binds to the *Ink* subsystem and current story, it binds multiple functions within *Ink* to external functions in *Unreal Engine*. *Inkpot* allows for binding these functions as well as listening for variable changes; we decided to utilize both. We listened for variable changes for smaller changes like the talking character's name, voice line, and sprite while saving functions for anything that interacts with another system or requires multiple parameters. This includes dialogue animations, starting and updating quests, tag events, and more.

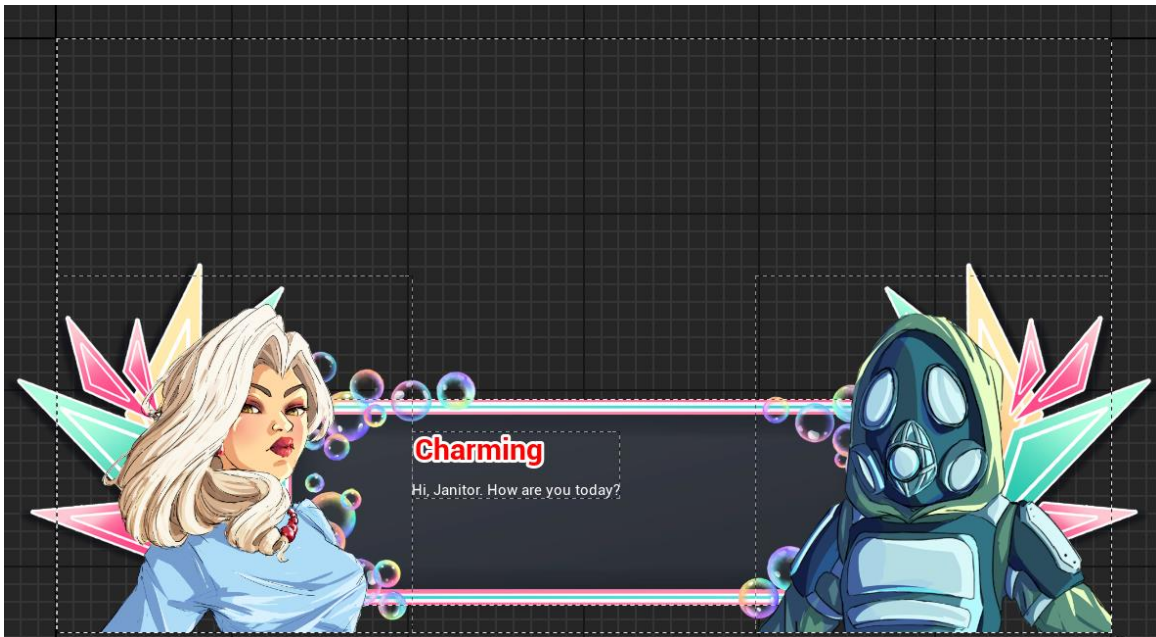


Figure 7.15: The widget designer of our dialogue widget, CAW_Dialogue.

One quick issue we realized was this made the dialogue widget exceptionally large. With binding to every function and variable for both direct dialogue control and outside system control, the widget needed to access multiple systems and ended up containing multiple functions for every single system. Realizing this growing issue, we decided to opt for switching binding to external systems onto the systems themselves. Since the *Inkpot* subsystem can be accessed anywhere in the project, we moved binding external functions to their respective systems. For example, the shop manager, instead of binding directly to the dialogue widget now binds on its own without needing to go through the dialogue widget whatsoever. This solved our growing issue and gave an overall better design pattern. Moving functionality for each system out of the dialogue allowed for better organization and allowed for the categorization of various functions.

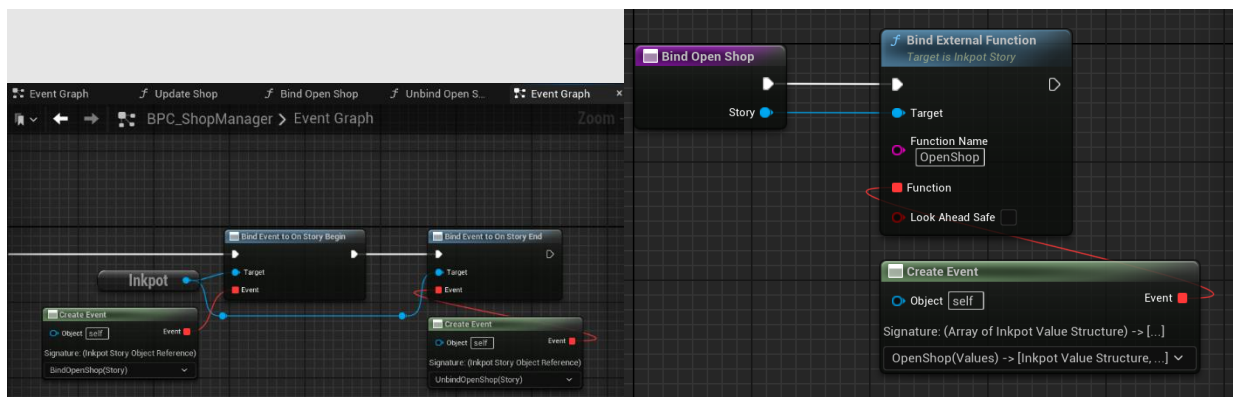


Figure 7.16: The Shop Manager binding to the *Inkpot* subsystem's On Story Begin event to a function that binds the function "OpenShop" to *Ink* calling OpenShop

7.9.2 Perfecting the Dialogue Pipeline

With programmers needing to be the middleman between writing and the engine to reimport all dialogue, we wanted to find a way to accelerate this process or cut it out entirely.

Reimporting assets, fixing errors, and needing to communicate back and forth with the writing team when syntax was not properly formatted or fixed caused large headaches on the team. With that in mind, we discovered a way to auto import dialogue assets so each member of the writing team could edit their ink files on their own machine and have it automatically updated in the project.

We first set up a shared *OneDrive* folder through our *SharePoint*. Each member of the writing team synchronized this folder onto their own personal computer. Since *OneDrive* automatically synchronizes each folder across machines, we were easily able to share ink files across machines. To allow for auto importing to occur, we modified *Unreal Engine's* editor settings to search for the *OneDrive* folder and import them directly into the project's content directory. This addition made *Unreal Engine* automatically look for changes in the folder, create new assets, and update already existing assets with the new *Ink* file. With these recent changes, writers could directly upload their *Ink* files without needing to go through tech. Any issues with the dialogue could be directly resolved through the writers with minimal communication needed. It also gave way to create multiple example files of dialogue formatting so communication of proper syntax could be centralized and maintained without needing to log in directly to *SharePoint*. With *MS Word* automatically formatting and changing quotation marks, it made understanding, copying, and writing *Ink* files easier.

7.10 Quest System

The quest system in *Clean Sweep* was by far the most involved part of the game. Since the quest system needed to interact with just about every aspect of the game, we needed to make

sure that we developed an easily expandable, easily modifiable, and easily maintainable system. When developing the quest system, we had the foresight to realize that there was not an effortless way to bind events directly to the quest system. *Unreal Engine* has an event dispatcher to broadcast events to actors that bind to it. This does have some limitations, namely the actor binding needs a reference to the actor sending the event. With multiple systems being in contact with the quest system along with many actors needing to broadcast events at runtime to the quest system, like every enemy needing to broadcast to the quest system when they're dead, we decided to switch over to a decoupled event system. More information on our process of selecting an event system can be found in section [7.3.3](#) and [7.3.4](#), but we ended up using the plugin *TMW Event Aggregator* for our decoupled event system.

7.10.1 Sending, Receiving, and Interpreting Events

As previously mentioned in section [7.3.4](#) and in figure 7.3, binding events to the event class in *TMW Event Aggregator* was extremely simple. We switched from using multiple bind statements for each class to using a single bind statement. Our gameplay tag event class contained two pieces of data: the gameplay tag and a struct called `BPS_IntAndName`. When planning the quest system, we considered all the quest types and what format data would need to be to update each quest type. When breaking it down, killing an enemy just needed an int for the number killed and a name for the enemy, getting money just needed the gameplay tag with an int value associated, using an item just needed the name of the item and the amount used, and every other quest task type we could conceive only at most needed to use the gameplay tag, a name, and an int. With that in mind, we minimized the data sent through the event system, only sending those

three pieces of data. This made our event system more universal for every type of quest and event and helped with developing a standard way to increase quest phases.

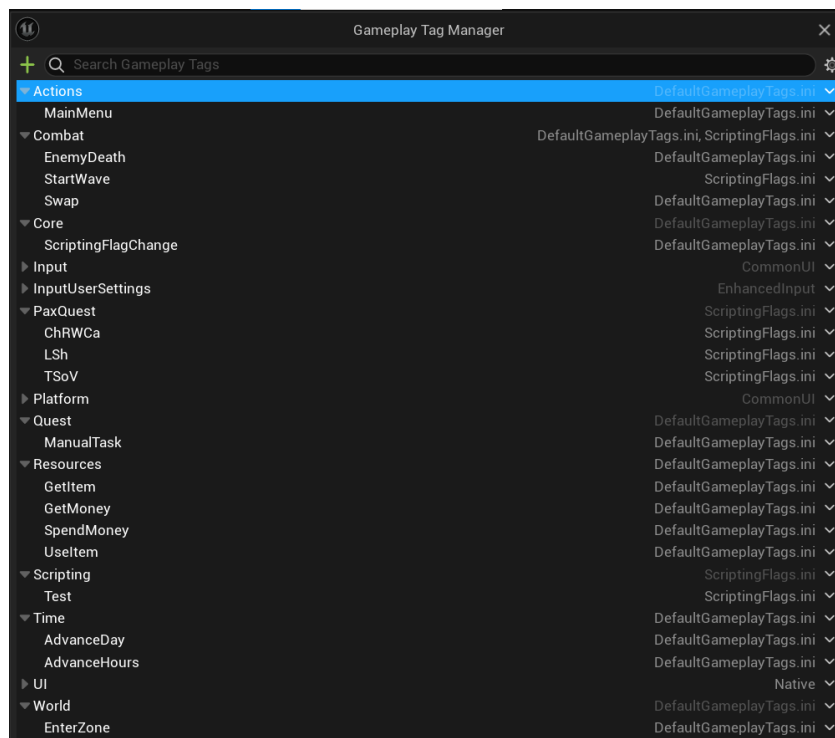


Figure 7.17: Our Gameplay Tag Manager with every gameplay tag we've used so far.

We created an event class through *TMW Event Aggregator* called `GameplayTagEvent`. This solely contained the data mentioned above along with two functions, one for updating the data and one for getting the data. Every gameplay tag event followed a similar structure for creation and sending data. We started by creating and saving a `GameplayTagEvent`. If this class already has data that we know it would give without the option for multiple, we set it in the exposed variables. Regardless, we just saved the tag event. Part of the reason for doing this was to limit creating events at runtime and limiting overall actor creation to being at the beginning of play. Next, we would load the data into the class, create any meta data we would need to, and broadcast the event to any actor subscribed to the class.

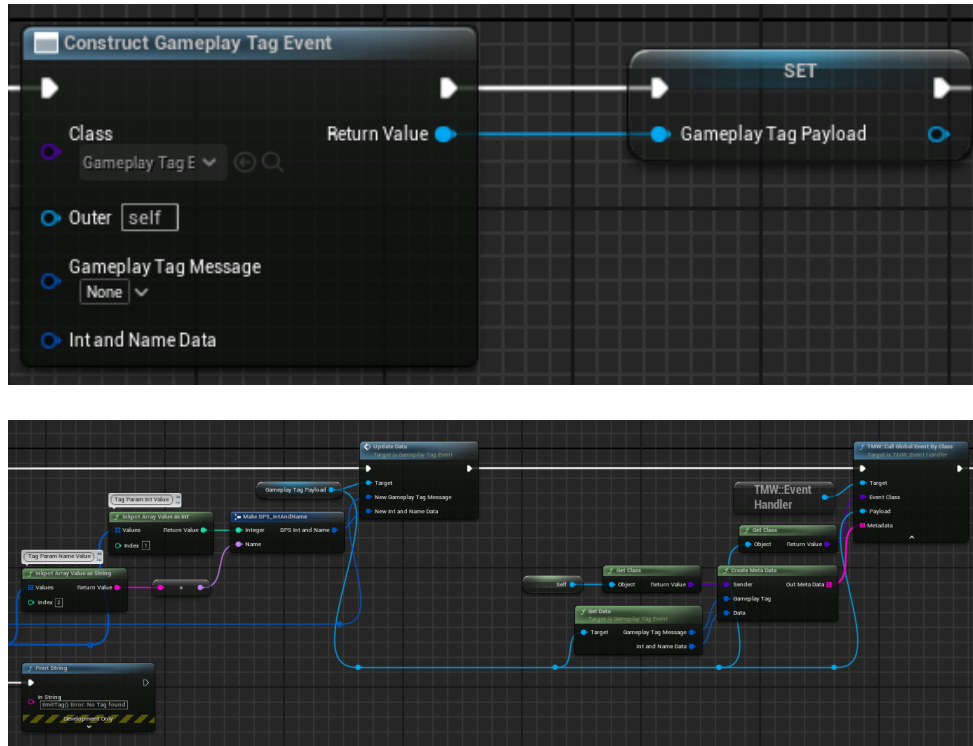


Figure 7.18: An example of creating a tag event, loading data into it, and broadcasting the message. This example is in CAW_Dialogue.

Once the quest system received this event, we were able to get the data from the tag event and interpret it. One of the ways we made the quest system expandable and efficient was how we interpreted the data received. Every quest may utilize the data a little bit differently. For example, one quest may only need a name to be completed while another quest might need both the name and value to indicate that it should update. We created two important pieces of data, a custom interface and a parent quest task type class. Each child class would already have the interface implemented with each using their own version of the class.

The quest manager would pass the data to each active quest which would start processing the message data. It utilized the quest task type class as an interpreter. We were able to use the quest task types as a variable despite being an object by having the quest use a reference to the class type. This also told the editor to only allow children of the parent quest task type to be used. We got that class from the quest, constructed an object to process the data, and moved on to the next quest. The quests were also responsible for updating the UI to the player, utilizing a custom-made queue that limited the total number of messages to the player to avoid having thirty notifications pop up and to have one notification pop up at a time.

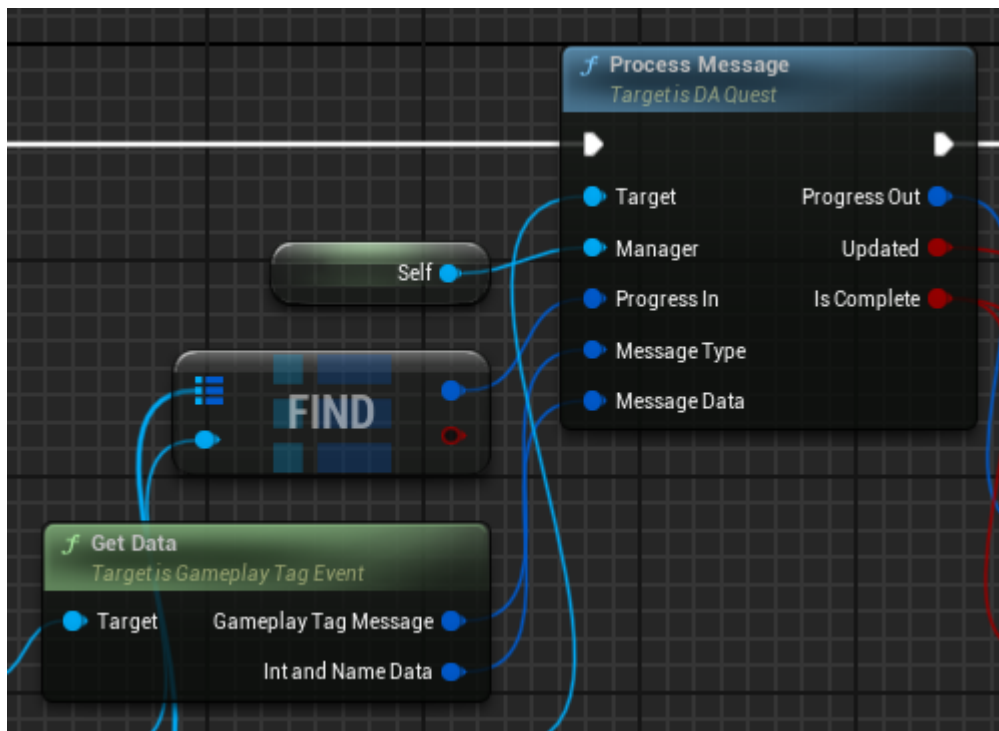


Figure 7.19: The Quest Manager processing the Gameplay Tag Event Class' data it received.

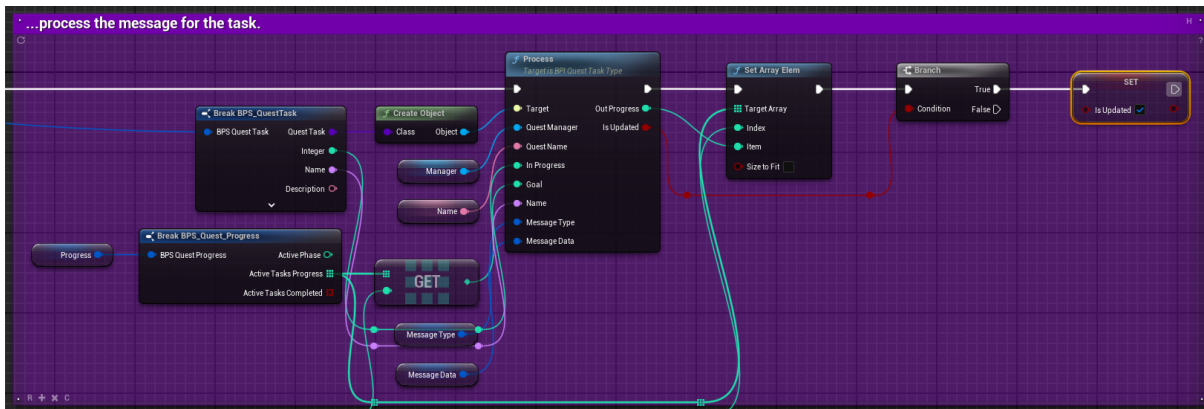


Figure 7.20: The Quest Data Asset utilizing the quest task type to process the Gameplay Tag Event's data and update the current progress.

7.10.2 Quest Creation and Expansion

Our quests were stored as a custom Data Asset called DA_Quest. They contained functions for checking progress on quests and giving rewards to the player. They also contained numerous variables that make up our easy to use and create quest system. These include meta data like name, description, and image, conditions to activate it and who gives the quest, each phase of the quest along with the quest task type needed to complete it and description, and finally the rewards for completing the quest. We wanted to make each step of creating a quest as simple as possible for designers and writers to use. To further that, we made sure each step of the quest creation process has tooltips on them that explain what each part of creating a quest is used for.

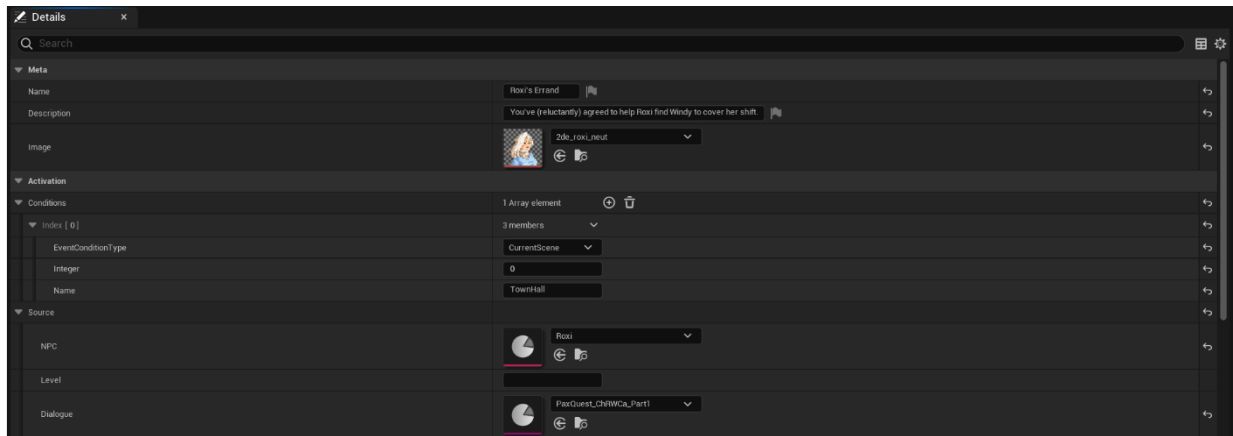


Figure 7.21: The quest data asset's meta tags for quest UI and its activations requirements.

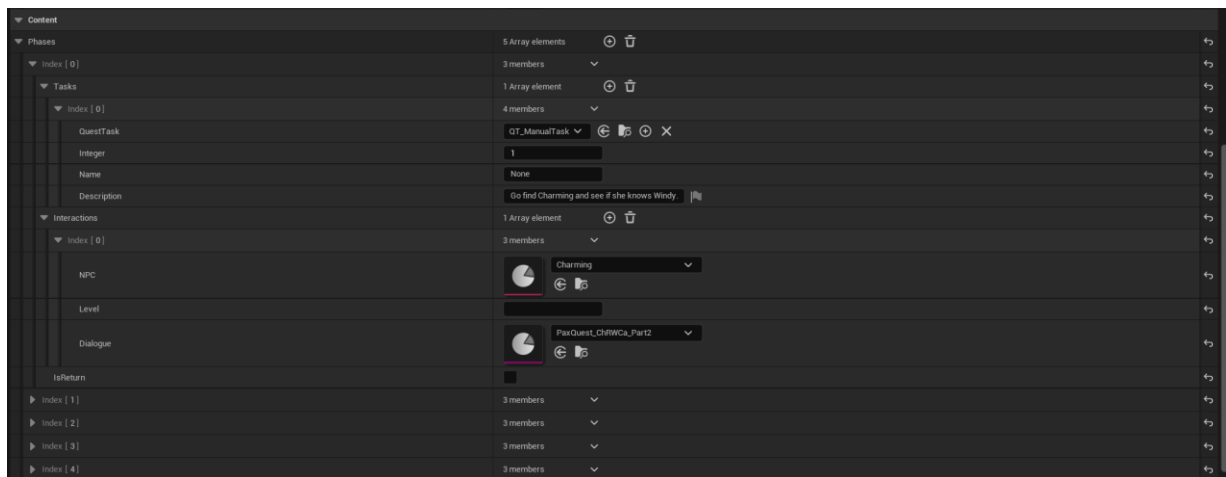


Figure 7.22: The quest data asset's phases utilizing quest task types, meta information, and dialogue associated with that phase.

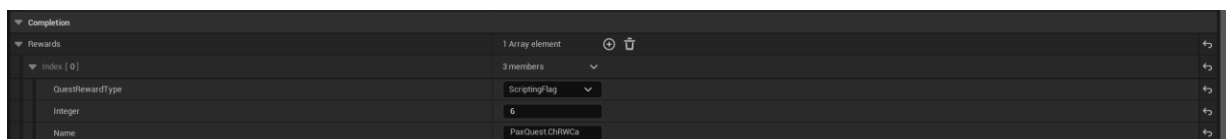


Figure 7.23: The quest data asset's rewards.

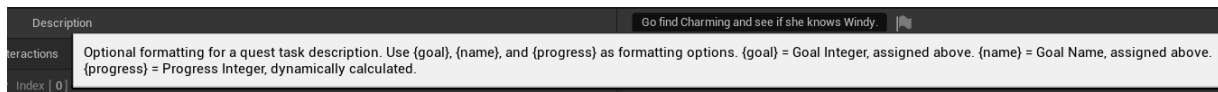


Figure 7.24: An example of a tooltip in the quest data asset.

As previously stated, quest task types were created to interpret gameplay tag data. To expand the system, we needed to create a new quest task type off the parent class. The functions needed to be updated fell into two categories, state and UI. Each quest task type needed to check if it was complete, process any message, and initialize progress. It also returned progress text and a description of the quest task for UI. This gave each quest unique instructions on how to complete them and how to showcase how much progress is left for the phase.

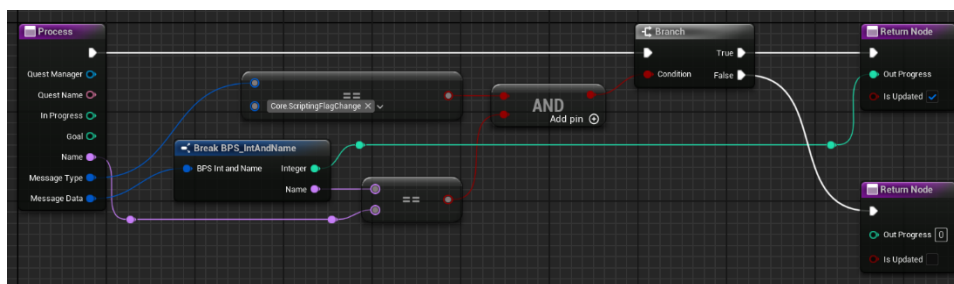


Figure 7.25: An example of a quest processing a message. This example is from QT_ScriptingFlag

7.10.3 Scripting Flags

Another big part of the quest system that is also used outside of the quest system is the scripting flag system. A scripting flag is a map of gameplay tags associated with a value that gives information about a certain aspect of the game. We decided to use them as ways to remember what quests were complete and give values to various stages of the game. For example, a scripting flag can be used to see what part of the main story the player is on and may have a scripting flag value of

<MainStory, 3>. We use these frequently in quests to check what progress of the quest the player is on. One such location is in the "Medicine for the Soul" quest. We have the player grab a rock from townhall to bring back to Shawn. The rock reads the scripting flag, and both allows the player to pick it up when they're on the right stage as well as destroy the rock after the player has picked up the rock; the rock checks for values greater than its pickup value and destroys if it is.

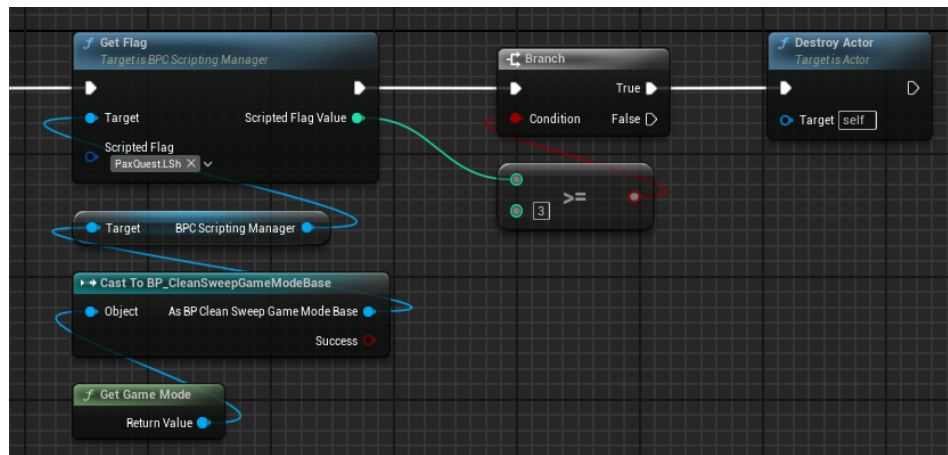


Figure 7.26: The quest rock's code for deleting the rock when spawning if the scripting flag is higher than the required value.

7.11 Recommendations

Our biggest recommendation is to build early and often. We had many sleepless nights because we built extremely late with many things coming down to the wire. If we had built earlier, we could have slept more and tested more thoroughly for bugs and errors that appeared within the project after building.

Another big recommendation is to think like a programmer. It's an unusual piece of advice that doesn't feel necessary, but it really is important. By thinking like a programmer, I mean plan out big

systems, write logic on pen and paper, create diagrams, code once, and adjust when needed. There were so many times where we'd try to brute force a problem until we solved it. B-Term was when we started thinking like programmers. We spent more time writing code on white boards and notebooks than coding. Overall, this helped us write better, more efficient code. The final, biggest piece of advice, communicate with your team members! When things went wrong for the programming team and members didn't communicate, all of production was slowed down. Waiting on a single task that is a blocker to numerous other parts of the project causes a big slowdown, and when no one is communicating, those blockers can't be alleviated.

Don't try to tackle a single problem all by yourself. Learn when to consult others, whether it be members of the team or online resources. Even just discussing with friends who could help come up with solutions would be a better way of solving things. If more communication was done, the project would have gone a lot smoother.

8 Audio

8.1 Soundtrack Design

Our goal for music was to create a full soundtrack that can be published alongside the game on Spotify. We wanted to create something that felt cohesive and polished. Soundtrack work fully began in D Term, running parallel with the Audio Lead's Producer as Composer music practicum so they could double dip with the work. The goal was to produce the highest priority music tracks first so they can be placed into the playtesting and Showfest builds.

8.1.1 Overworld

For the overworld, we decided to create dynamic tracks. For each of the three times of day in Cleanland (day, evening, and night), a theme was made. From these base themes, the timbre would be changed based on the location to best suit the environment. We did this to keep the soundtrack modular and cohesive, taking inspiration from Toby Fox's compositional approach with developing the music of *Undertale*. A side effect of this approach was an increase in the speed and efficiency of the music creation process since extra time was not needed to make each track completely unique. Tracks would fade in and out based on the change in the time of day.

For interiors, we created a unique location theme that will be played all day. We wanted the interior music to fully match and encompass the environment, and we felt like it was unnecessary to create multiple tracks that would change with time considering that the player would not be able to see time pass within these locations. Also, the time spent inside is noticeably

less than outside, so one track for each location would suffice. This approach also allowed us to focus on realizing the creative vision for each location to create the best experience for the player.

8.1.2 Combat and Character Themes

For combat, we wanted to create high energy tracks to keep the player pumped up, engaged, and eager to fight germs! We utilized genre blending between hard rock and EDM to diversify the timbre and interest for the player, including adding vocals on some tracks. An audio design decision that was made was to use electronic synth-y instruments when germs are present and limit their usage elsewhere, making them symbolic of The Bacterial Plane. Another audio design decision for the combat music was to have the lyrics reflect cleaning themes to root the music in the world of the game and add to the comedic effect (see Appendix L).

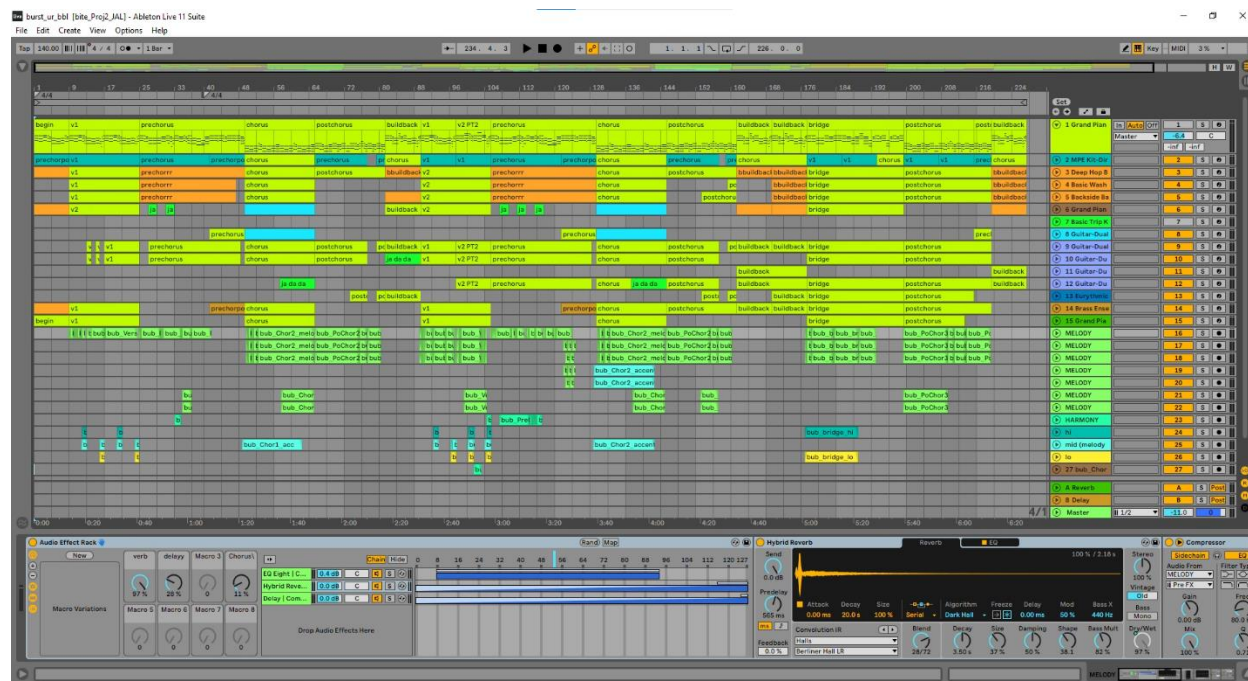


Figure 8.1: The Ableton workspace for *BURST UR BUBBLE*, a combat song.

Every character route has a pivotal event near the middle of their arc and a final event showcasing their growth. To build the route's pivotal moment as the focal point of personal development in character routes, the plan was to work alongside writers and create a unique song/theme for each character that represented their personality and arc. This theme would be played during the character's house cleaning combat level to help build the intensity around overcoming personal struggle and acknowledging flaws. The character themes from that route event would then be reworked into an uplifting reprise featured in the character's final route event to showcase their growth and sonically remind the player of the character's identity. The reprise would be played at the end of the route during a heartfelt moment between the Janitor and the character. These songs were not worked on during the academic year due to low priority but have been planned for future development.

8.1.3 Worldbuilding and Miscellaneous

Outside of our highest-priority music development, songs for worldbuilding were also planned but have been delayed until future development. Various parody songs were ideated about in playful fun, and one was recorded for a marketing post that was never published, titled *'The Cleana Colada Song,'* a parody of *'Escape(The Pina Colada Song)'* by Rupert Holmes, involving developing the drink recipe for the song's namesake. We also tabled the parodies due to us not knowing how to navigate parody law for a parody in a video game. As for world building songs, the Scent5ations' fictitious top-charting hit, "Fresh Start Fantasies", was in development, but was tabled due to its low priority status. These worldbuilding songs may be placed in the overworld,

on a jukebox in the mansion, or selectable during janitor jobs like selecting a song for a stage in *Super Smash Bros*.

8.2 Sound Effects

Clean Sweep includes over 700 sound effects. Our five main categories of sound effects were UI, combat, barks, movement, and world. To keep everything streamlined, the audio lead, who was also the creative director, decided to take over all UI sounds to make sure the vision stayed cohesive with the rest of the game. When UI sounds were created, keeping the sound profile consistent with the vibe of the game was integral. The general vision was something bright, synth-y, and clean, taking inspiration from *Genshin Impact*. The UI sound effects were made in Ableton by adding reverb, delay, pitch shifters, EQ, compression, and other effects to layers of synths and electronic drums. For more melodic stingers, such as the janitor job jingles, a music pipeline was followed, adding various instruments, and balancing them until the desired timbre and emotion was achieved.

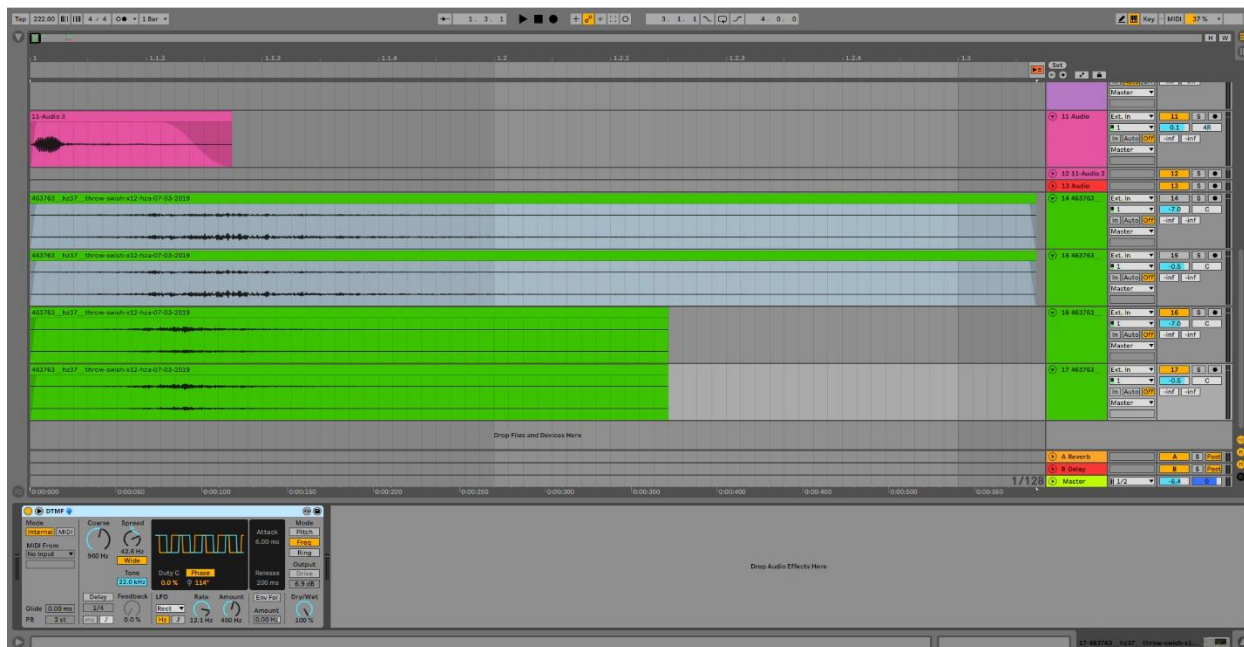


Figure 8.2: Adding different effects to folied audio to create unique UI sound effects.

Genshin Impact was a major inspiration for our combat, where we aimed to create sound effects with an anime influence. Combat sounds were dispersed to the other members of the audio team through a design-to-audio pipeline. Design would add what sounds characters needed in combat to the asset list and added descriptions to help direct the audio design. The asset sheet was then given to the audio team, and sound effects were assigned. Base sounds were either folied by the audio team or downloaded from online sound libraries. These files were then brought into Reaper and layered with each other to create richer sounds, and then had effects added onto them to give them extra juice. Movement, such as surface-based footsteps, and overworld sounds, such as location-based ambience, followed the same process, but with broader direction and more creative freedom from the audio designers.

Barks were developed by the audio team members who had writing and design overlap. During voice recording sessions, writers consulted the moveset sheet, and gave the actors

direction on what sounds to make based on the moves they were performing. Lineless action-based grunts, such as getting hit and hitting, were improvised by the actors, and line-centric grunts were decided by the writers and given to the actors. Several takes were recorded, with rerecording requested if the performance or audio quality needed to be modified. During our recording sessions for barks and general voice acting, we aimed to keep the volume between -16 and -10 dB to avoid distortion. After the lines were recorded, they were packaged in the characters' voice acting folder to be reviewed by the writer, who would trim the files to the best takes, and upload them to Sharepoint for an audio team member to edit into final renders.

8.3 Voice Acting

For our PAX build, we added placeholder voice acting for our quests to see how difficult it would be for the final game and how long it would take to travel through the pipeline. Audio that needed to be captured was full lines for the quests, emotive grunts to be used in less important dialogue, and barks for combat characters, as previously described. At the end of C Term, members of the writing team reached out to fellow WPI students, WPI alumni, and non-WPI-affiliated friends that they felt could best represent their characters' voices and asked them if they would like to record. During this time, writers created Voice Actor Bibles for each character, giving actors some direction in achieving the right tone. These documents contained images of the character, general information about each character, their backgrounds, their route information, and their voice comparables.

During C Term break, two writers who had audio team overlap ran recording sessions with each of the Voice Actors in the Fuller Audio Lab's whisper room, where they facilitated the

sessions and provided feedback to actors. All recordings were captured over two (very long) days and then passed onto writers to comb through and select their favorite takes. The process was the same for recording barks, and the selected recordings were then given to an audio team member who edited and rendered them to be used in the game.

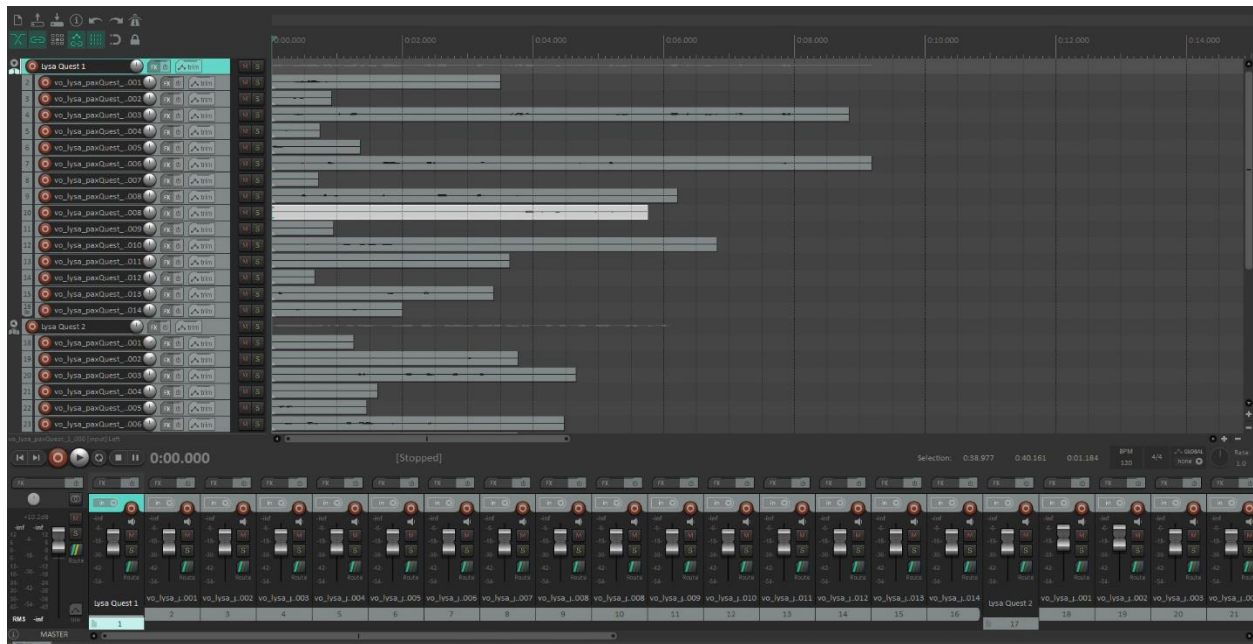


Figure 8.3: The line selection process in Reaper.

Overall, the voice recording process was quite arduous, and for such a small team, it bogged down progress on the build due to the overlap. It did make us narrow our focus for the final published build to include voice acting only in the main story and routes and use character grunts in the less important dialogue (overworld, general quests, hangouts, etc). We do still plan to use voice acting in our published build, and the process did help us find some great voice actors for characters. We will be holding official auditions and move through the recording process this summer.

8.3 Implementation

Implementation for each type of sound took various approaches. Universally, each sound needed various components to be properly imported. Each sound needed a sound class and attenuation. A sound class allows us to apply modifications to every sound that inherits the class. The main use of sound classes allowed us to categorize our sounds. We created a music class and a sound effects class to apply to all sounds. We also utilized a sound class mixer. A sound class mixer allowed us to adjust the volume of each sound class in our settings widget.

Sound attenuation allowed for sound effects to have distance, getting quieter the farther away the player is from it. In a 3D environment, this detail is necessary to convey a sense of space and depth to the player. We planned on creating multiple attenuations for different categories of sounds, but due to time limitation, we created one universal attenuation. In regard to implementation, there were three main ways to import sound effects. We could play sounds through blueprint functions, through animations, and through being placed in a level.

Sound effects fell into all 3 categories. Certain combat sounds like receiving damage or dealing damage made sense to play in a blueprint function. Since animations couldn't tell if someone was hit, we needed sound effects to play on enemies or players receiving damage. Other sounds were placed directly into animations. *Unreal Engine* has an animation notify that plays a list of sound effects. We also created our own animation notify that linked materials the players stepped on to the corresponding sounds. The animation notify analyzed what material was under the player's mesh and played the corresponding sound effects for stepping on that surface.

Music was only played blueprint functions. We created custom functionality within our game mode to play certain music across scenes. All that was checked was if the music currently playing matched the one for the current game mode. Since we utilized two game modes, one for combat and

one for the world, there were only two types of music that could play: world music and combat music. For future development of our music implementation, we'd like to add a more complex system that can layer sections of a song. We aim to have that the better the player is playing, the better the music sounds; however, with time constraints, we decided that this feature was best saved for summer.

Dialogue voice lines and sound effects were implemented through blueprint functions. To make implementing dialogue as simple as possible and using what we learned from the errors of using our previous plugin, *Runtime Audio Importer* (more info can be found in section [7.3.7](#)), we created a custom data asset and table for dialogue voice lines and sound effects. With our initial plugin, we tried to play sounds in folders, but it didn't work. To make transitioning from the old system to the newly created one for writers while also making it easy for audio implementation, we created a table that held onto maps that contained a category name and dialogue data asset. We essentially virtualized the folder structure through the tables and data assets. The dialogue data asset simply contained soft references to dialogue, only needing audio to be dragged and dropped into the data asset. Soft references are references to a file that is unloaded. They were used to minimize ram usage.

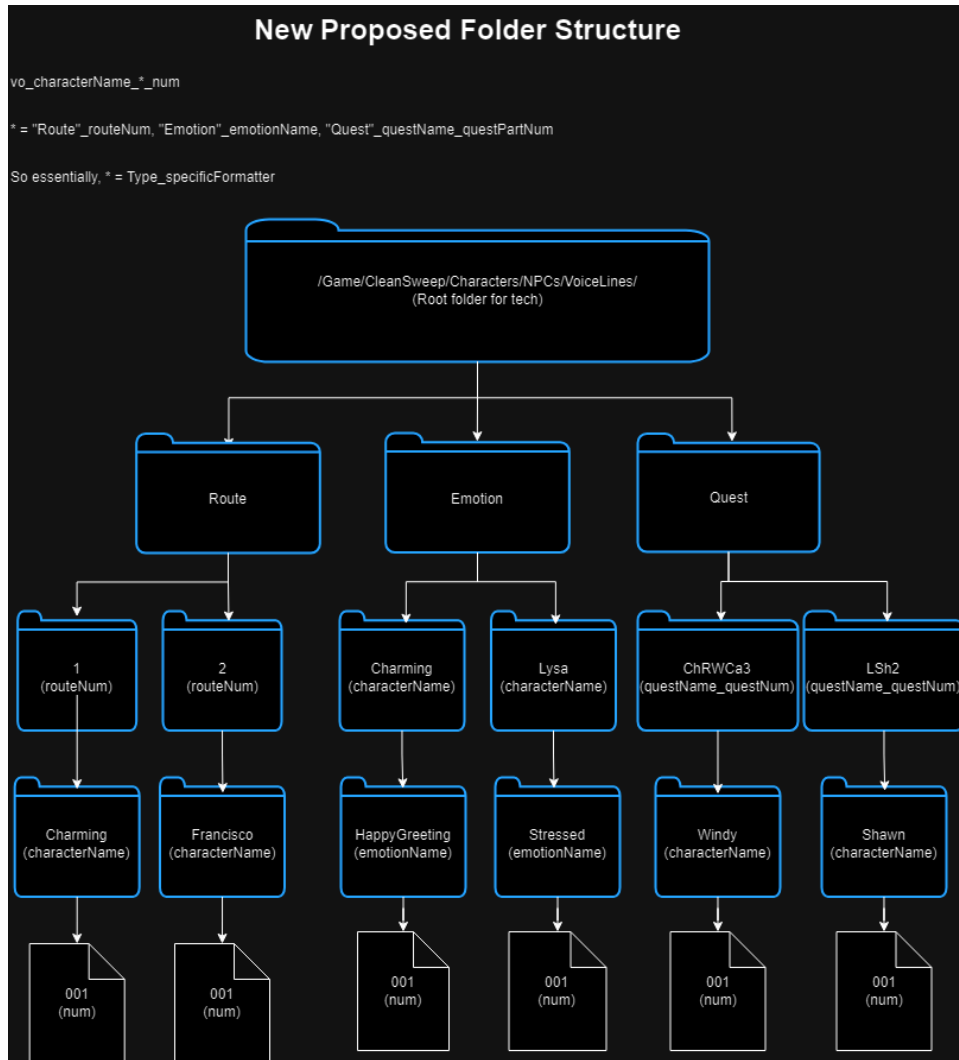


Figure 8.4: The proposed folder structure with *Runtime Audio Importer* that became the foundation for the current implementation of dialogue voice lines and sound effects.

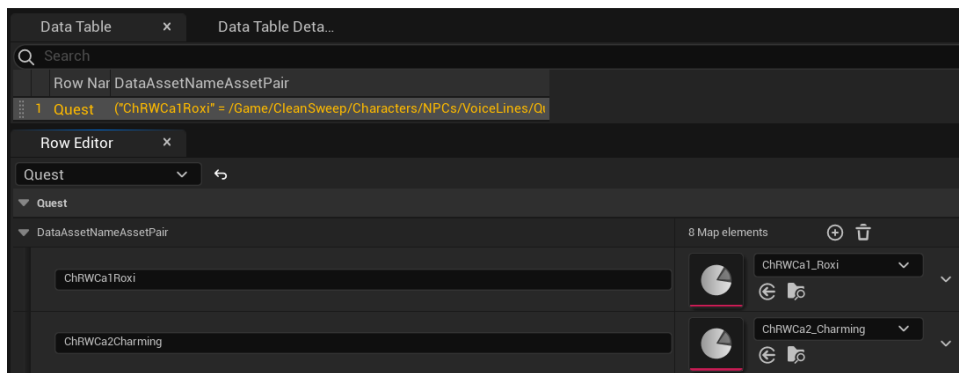


Figure 8.5: The lookup table for dialogue voice lines being implemented.

Finally, we implemented ambient sounds through placing them in the world. These sounds were in levels and contained their own size and attenuation settings.

8.4 Recommendations

Before jumping into work, ensure that all ISPs are on the same page with the project vision. A bottleneck encountered in early C Term was a lack of understanding of the project's concept from non-core team members. Audio team members would create sound effects that were not parallel with the vision and required multiple iterations to capture the desired sound. This can be ameliorated early on by providing ISPs with introductory material to acclimatize them to the project and the feel of the game.

Keep your asset sheet thorough and updated. The asset sheet was a weak point for audio production, especially because the game designers did not know how to describe combat sound effects in a way that was easily understandable to the audio team. This led to miscommunication, confusion, and more work on all sides. Accurately squaring away and communicating between teams early and often is a surefire way to avoid this problem.

Find team member strengths early and assign them accordingly. This recommendation works in tandem with the above, but figuring out what each team member excels at early on will increase productivity and team morale.

Create an Audio Bible. Making an easily accessible live document that serves as an Audio Bible can make the onboarding process much smoother and can help the team have a more unified approach to sound design.

9 Marketing

9.1 Social media

Early in the MQP process, we created social media accounts on several platforms to advertise our game including *Instagram*, *Tik Tok*, and *YouTube*. We had ideas for posts to promote our game and build an audience, which we kept track of in our meeting minutes and in a designated marketing *Discord* channel. During C term, we created a posting of three posts per week, each core member creating one every other week. The plan was to focus on short form video content, as that is a popular format on our three primary social media platforms. Unfortunately, sticking to this schedule was exceedingly difficult, as marketing was the lowest priority action item for group members.

9.2 Events

The events at which we present our game, Protofest, Alphafest, the IGDA Clark Showcase and soon to be Showfest, were all advertising opportunities. Even at PAX, an event where our game was not selected to be presented, we were able to speak with fellow game lovers and professional game developers, as well as garner interest in *Clean Sweep*.

9.3 Handling Copyright

Having a game with a heavy reliance on popular cleaning products, concerns about copyright were constantly on our minds. This primarily manifested in the naming of characters, locations and items. For almost the entire project, the team called what is now *Scrub-A-Lucious Boba*, *Scrubbing*

Bubbles. Other instances included the item Clorox-cola, which was changed to Cloro-cola. Mayor S.C. Jenkins was simply S.C. Johnson, and Francisco was Fabuloso, which although is a word in and of itself, was changed with consideration regarding both potential reinforcement of stereotypes with this being the name of a gay character, and to mitigate the chance of a copyright claim.

9.4 Recommendations

If you want to publish, consider getting a marketing ISP. If marketing is nobody's priority, it will constantly be put on the back burner. Having someone dedicated to marketing is ideal.

Make sure your marketing aligns with the game that you are making. Don't advertise things that are not representative of the game or things that won't be in it. You don't want to make false promises.

Establish your target audience and market with them in mind. Establishing a target audience early on will also aid in the overall development of the game.

10 Results and Analysis

Throughout the year, we held nine playtesting sessions. These sessions provided invaluable insights into what aspects of our game were functioning well and what needed improvement. Early on, we discovered that combat needed to be faster paced to keep players engaged and to better match the energetic tone of the game. Additionally, feedback from playtesters highlighted certain areas of the user interface and controls that were cumbersome, prompting us to streamline and refine these elements for improved usability. These playtesting sessions not only helped us identify areas for improvement but also validated the effectiveness of the changes we implemented based on player feedback. For example, after increasing the pace of combat from one of our earlier playtesting sessions, we were able to see a marked improvement in our survey results on the topic (figure 10.1). Playtesters were also a great source of bug and QA testing, revealing several errors we wouldn't have otherwise found, such as hitboxes being improperly sized and problems with UI interactions.

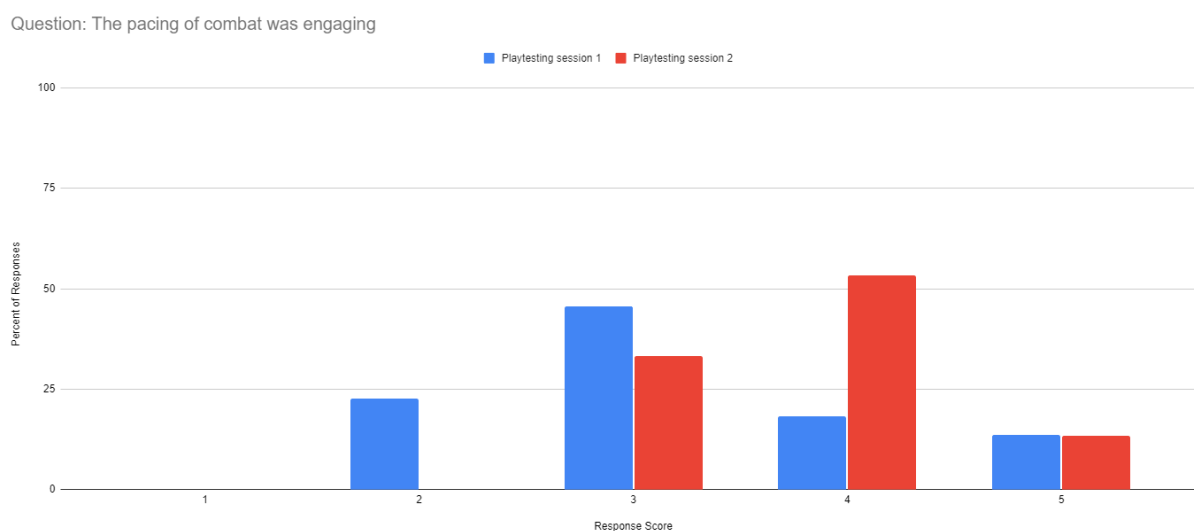


Figure 10.1: Responses for combat pacing from 2 different playtests

11 Conclusion

11.1 Accomplishments

Overall, within our four terms of working on this MQP we were able to establish a playable core gameplay loop, which we view as a huge accomplishment. Although our scope had to be significantly cut down, we have developed a product that is fully playable and able to be iterated on. Additionally, we were able to do several playtesting sessions throughout the year. Because of this, as well as other deadlines and conventions, we were able to develop polished builds that represent a vertical slice of the future product. Even though our game is not complete and ready to be shipped, we have established a solid foundation that can be expanded upon.

Even though our deliverable for this project is a vertical slice of the final product we plan to release, we were still able to produce an impressive amount of content while maintaining our standards of originality and quality. Over the course of the year, we created:

- An explorable town containing eight environments
- 312 modeled 3D props
- 32 modeled 3D characters
- 62 2D sprites
- Over 776 sound effects and voice lines
- Seven songs
- 70 animations

- Fifteen technical art tools for motion capture
- Nine written character routes for our final game
- And, of course, too many cleaning puns to count!

In terms of raw numbers, these far exceed the expectations of MQPs in the past. Being able to produce this amount of quality content while having a functional product is something that we are very proud of as a team. Our artists pushed the boundaries of what was possible, and having these elevated expectations challenged other teams. This forced all of us to develop a more thorough understanding of the entire 3D game development process in a short period of time.

11.2 Setbacks and Failures

Along the road of developing *Clean Sweep*, we dealt with numerous setbacks and failures. Some of the biggest setbacks included modulating scope, teammate oversaturation, communication, and time management.

One of our biggest failures was not having our game showcased at PAX East. Every year, WPI sets up a booth at PAX to feature student made games. This year, there was only space for four games to be shown, so teams had to submit a build to a panel of students who would decide which projects would be shown a few weeks before PAX. Unfortunately, the build we submitted had a bug that crashed the game, and for that reason we were disqualified from having our game featured. This was enormously disappointing for the team, as PAX would have been a spectacular opportunity to highlight our work to a wide audience. Luckily, we were able to fix the bug shortly after the disqualification. This benefited the team's approach to scope and Showfest goals, as we

decided to use the expo-goer friendly PAX build as a format for our D-term deliverable demo. This helped us reprioritize our goals and polish an approachable vertical slice that otherwise would have been oversaturated with content.

Early in the MQP process, we created social media accounts on several platforms to advertise our game. Unfortunately, our social media presence diminished beyond our expectations, as we could not commit to a consistent schedule due to the interruption of higher priority tasks. Instead, the team decided to postpone the effort until post college. Upon reflection, having a team member or ISP dedicated to marketing should have been a consideration.

11.3 Advice for Future Projects

We could have benefited from more regularly evaluating our scope. It's a widespread problem for IMGD students and the game industry to overestimate what can be done with a finite amount of time, and our team fell victim to this. If we had realized earlier that our project was too ambitious, we could have made scope cuts before too much work was done, resulting in less time and resources being lost. In the future, teams should be aware of the sunk-cost fallacy and be more willing to scrap work for the greater good of their project before it's too late. Additionally, teams should set realistic and measurable goals during the year, and thoroughly evaluate their progress often.

Build often and build in advance. We frequently put off building our game up to its deadline to get everything implemented. This prevented us from effectively testing for bugs that would only arise once built. With their issues not being immediately apparent, these errors took longer to fix. The result of this was a large lack of sleep for members of the team trying to fix issues hours before

a playtesting session or showcase. Building a couple of days beforehand or even a week before would have fixed numerous issues we have faced over the year.

Be prepared to let things go. The hard truth about game development is that you will never have a perfect project. Not everything you make will be able to go into the final deliverable. While this is disappointing, it is important to take a step back and look at what you did learn and achieve in the process. It is also important to prioritize your well-being; letting your work become and define your life never leads to a good outcome.

11.4 Future Development

As stated earlier, what we created this year is a vertical slice we will expand on. Luckily, we were able to make most of the big game decisions with our limited time working together, and now the game just needs to be filled out. We only pushed release back a month and are still on track for summer release, aiming for August 2024.

Of our ten playable characters, three are playable in the current build. We plan to further develop our unfinished characters to make sure that each character has a unique moveset and story route. More non-playable characters are also on the docket for development to build the world of *Clean Sweep* and provide more representation for underrepresented groups.

For writing, route work has been done, with some routes completed, but none are present in our Showfest build to prioritize a polished vertical slice over content. We also plan to create more quests to develop the world, content, and comedic presence of the game.

The audio deliverables, especially in terms of soundtrack, will be worked on further, aiming to achieve the goals described in the audio section of a full soundtrack with multiple combat themes, location-based music, and route-based music.

For environments, the mansion can currently be entered and explored, but more levels need to be designed and modeled to complete the location for the rest of the main story. Additionally, the plaza needs to be finished, which contains Scrub-a-Licious Boba, Ph Level Up, and The Periodic Table. Lastly, the park has not been started yet, which is where some of the side story routes take place. Of course, in the future the existing work will continue to receive further polish and detailing.

11.5 Final Remarks

Overall, *Clean Sweep*, while it faced its fair share of trials and tribulations, defied expectations and broke new ground of what an MQP could achieve. We are immensely proud of what each one of us has produced this year and want to extend our sincere thanks to everyone who has supported us through the development process. This is only the beginning, and we cannot wait to see what IMGD will do next with the tools we have left behind. We thank you for reading our paper, and most of all, we hope that you enjoy *Clean Sweep*!

Works Cited

- “A MAN OF FEW WORDS, BUT MANY Muscles.” *Household Cleaning Products | Mr. Clean®*, Procter and Gamble, www.mrclean.com/en-us/about-mr-clean. Accessed 14 Apr. 2024.
- Awad, Germiné H., et al. “Beauty and Body Image Concerns among African American College Women.” *The Journal of Black Psychology*, vol. 41, no. 6, Dec. 2015, pp. 540–64. PubMed Central, doi.org/10.1177/0095798414550864.
- “Differentiation of Self.” *The Bowen Center for the Study of the Family*, www.thebowncenter.org/differentiation-of-self. Accessed 12 Apr. 2024.
- Gami, Pankit Nepotism in the Workplace: Concern, Types, Examples & Cons. 19 Oct. 2023, knovator.com/blog/nepotism-in-workplace/.
- Herbox. Stan Lee - “Anyone Can Be Spiderman.” 2022. YouTube, <https://www.youtube.com/watch?v=aHA4JaU48L0>.
- Holland, Maggie, et al. “17 Manipulation Tactics Abusers Use.” *Choosing Therapy*, 3 May 2022, www.choosingtherapy.com/manipulation-tactics/.
- Karie, Kaufmann. “Recognizing and Managing Nepotism in the Workplace.” *Coaching Blog*, 6 Sept. 2023, kari Kaufmann.com/nepotism-in-the-workplace/.
- Kim. ““Can i Save Them?”” *The Hotline*, 15 July 2021, www.thehotline.org/resources/can-i-save-them/.

Marineli, Filio, et al. "Mary Mallon (1869-1938) and the History of Typhoid Fever." *Annals of Gastroenterology : Quarterly Publication of the Hellenic Society of Gastroenterology*, vol. 26, no. 2, 2013, pp. 132–34. PubMed Central, www.ncbi.nlm.nih.gov/pmc/articles/PMC3959940/.

Pham, Erik. "Council Post: Toxic Positivity at Work: Examples and How to Deal with It." *Forbes*, Forbes Magazine, 5 June 2023, www.forbes.com/sites/forbesbusinesscouncil/2023/06/02/toxic-positivity-at-work-examples-and-how-to-deal-with-it/?sh=16eeb24673e6.

Rahman, Insha, and Benjamin Troy. "How to Leave a Toxic Relationship." *Choosing Therapy*, 26 Aug. 2022, www.choosingtherapy.com/how-to-leave-a-toxic-relationship/.

"Rabies." *World Health Organization*, World Health Organization, Sept. 2023, www.who.int/news-room/fact-sheets/detail/rabies.

Reynolds, Graham. "Toxic Positivity." *Anxiety and Depression Association of America, ADAA*, 23 Sept. 2022, adaa.org/learn-from-us/from-the-experts/blog-posts/consumer/toxic-positivity.

Ricketts, Adriana, et al. "Burnout Response for Leaders." *Workplace Strategies for Mental Health*, 1 Jan. 2016, www.workplacestrategiesformentalhealth.com/resources/burnout-response-for-leaders.

“Religion and Coming out Issues for African Americans.” *Human Rights Campaign*,

www.hrc.org/resources/religion-and-coming-out-issues-for-african-americans. Accessed
23 Apr. 2024.

Saxena, Silvi, and Kristen Fuller. “Signs You’re Socially Awkward & How to Deal with It.” *Choosing
Therapy*, 26 Jan. 2023, www.choosingtherapy.com/socially-awkward/.

Rosenthal, Lisa, and Marci Lobel. “Stereotypes of Black American Women Related to Sexuality
and Motherhood.” *Psychology of Women Quarterly*, vol. 40, no. 3, Sept. 2016, pp. 414–27.
PubMed Central, doi.org/10.1177/0361684315627459.

Saxena, Silvi, and Naveed Saleh. “What Is a Toxic Relationship? Signs, Impacts, & How to Fix It.”
Choosing Therapy, 2 Sept. 2022, www.choosingtherapy.com/toxic-relationship/.

Schoch, Conrad L., et al. “NCBI Taxonomy: A Comprehensive Update on Curation, Resources and
Tools.” *Database*, vol. 2020, Jan. 2020, p. baaa062. DOI.org (Crossref),
doi.org/10.1093/database/baaa062.

Shafir, Hailey. “Work Anxiety: Signs, Causes, & 13 Ways to Cope.” *Choosing Therapy*, 21 Nov.
2022, www.choosingtherapy.com/work-anxiety/.

“Signs of Love Bombing.” *The Hotline*, 26 Jan. 2024, [www.thehotline.org/resources/signs-of-love-
bombing/](http://www.thehotline.org/resources/signs-of-love-bombing/). Accessed 2 Sept. 2022.

Sipe, Danielle. "Turning Negatives into Positives with Cognitive Behavioral Therapy!" *Pathway Family Services*, pathwayfs.org/turning-negatives-positives-cognitive-behavioral-therapy/.

Accessed 6 Oct. 2023.

Tashiro, Ty. *Awkward: The Science of Why We're Socially Awkward and Why That's Awesome*. HarperCollins, 2018.

Taylor, Paul-Roy, and Rajy Abulhosn. "Differentiation of Self: An Overview & Why It's Important in Relationships." *Choosing Therapy*, 31 Oct. 2023, www.choosingtherapy.com/differentiation-of-self/.

"The AI of Marvel's Spider-Man | AI and Games #74" YouTube, uploaded by AI and Games, 9 November 2023, www.youtube.com/watch?v=KVUh6vPFN4k.

"What Is Emotional Abuse." The Hotline, 15 Dec. 2023, www.thehotline.org/resources/what-is-emotional-abuse/.

"Writing People of Color by MariNaomi." Midnight Breakfast, midnightbreakfast.com/writing-people-of-color. Accessed 23 Apr. 2024.

Writing Poc 101 - Black American Characters - @jetsilks - Wattpad.

www.wattpad.com/173768010-writing-poc-101-black-american-characters-jetsilks.

Accessed 23 Apr. 2024.

Shirom, Arie, et al. "Burnout and health review: Current knowledge and future research directions." *International review of industrial and organizational psychology* 20.1 (2005): 269-308.

"Self Neglect | DSHS." www.dshs.wa.gov, www.dshs.wa.gov/altsa/home-and-community-services/self-neglect#:~:text=Adult%20Protective%20Services%3F-.

Vashi, Neelam A. "Obsession with perfection: Body dysmorphia." *Clinics in dermatology* 34.6 (2016): 788-791.

Appendix A: Asset Production Excel Sheet

Example excel spreadsheets containing lists of assets and their status. There are several sheets within this document, each pertaining to a type of asset.

	A	B	C	D	E	F
1	Character File	Modeler	ART STATUS	modeling issues	TECH STATUS	tech issues
2						<20,000 tris, high low, Vector Color
3	Characters					
4	3dc_janitor	Jess	Complete	low model + SubD levels for all meshes	In Progress	Optimizatiomn, UVs, Baking, Textur
5	3dc_beau	Jess	Complete	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
6	3dc_blanche	Jess	Needs Approval	low model + SubD levels for all meshes	Not Started	hand bones, Clothes Baking...
7	3dc_charming	Jess	Complete	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
8	3dc_lysa	Jess	Complete	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
9	3dc_roxi	Jess	Complete	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
10	3dc_cain	Jess	Needs Approval	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
11	3dc_vic	Jess	Complete	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
12	3dc_shawn	Jess	Needs Approval	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
13	3dc_tidy	Jess	Needs Approval	low model + SubD levels for all meshes	Not Started	
14	3dc_soap	Jess	Complete	low model + SubD levels for all meshes	In Progress	hand bones, Clothes Baking...
15	3dc_windy	Jess	Complete	low model + SubD levels for all meshes	In Progress	redo skeleton
16	3dc_arman	Renee	Needs Approval	low model + SubD levels for all meshes	Not Started	hand bones, Clothes Baking...
17	3dc_maya	Renee	Needs Approval	low model + SubD levels for all meshes	Not Started	hand bones, Clothes Baking...
18						
19						
20						
21	Enemies					
22	3dn_baseGerm	Jess	Needs Approval		In Progress	finger joints, issue merged verts on

Appendix B: Playtesting Feedback Form

This survey was taken by playtesters. Respondents varied in gender and age with a majority being WPI students.

Clean Sweep Playtesting Feedback

Thank you for playing our demo! Please rate your agreement with the following statements regarding your gameplay experience.

cleansweepthegame@gmail.com [Switch account](#)

Not shared

The controls were intuitive.

1 2 3 4 5

Strongly disagree Strongly agree

The pacing of combat was engaging.

1 2 3 4 5

Strongly disagree Strongly agree

The character swapping mechanic enriched combat.

1 2 3 4 5

Strongly disagree Strongly agree

The frequency of enemy attacks was *not* overwhelming.

1 2 3 4 5

Strongly disagree Strongly agree

The sizing of spaces were appropriate for combat scenarios.

1 2 3 4 5

Strongly disagree Strongly agree

Please leave any additional comments or feedback about *Clean Sweep* in the space below.

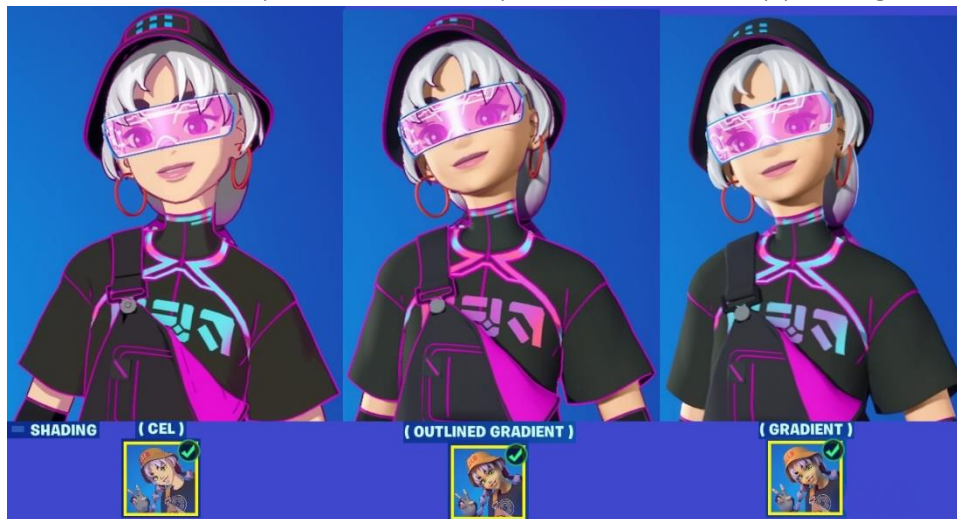
Your answer

Submit [Clear form](#)

Appendix C: Art Interview Feedback Questions

These questions were asked to playtesters. Respondents consisted of WPI students who varied in gender and age.

1. How well does the art style align with the playful concept of the game?
2. How well does the music contribute to immersion in the town?
3. Were there specific environmental elements that caught your attention, either positively or negatively?
4. Which of these three styles of shaders do you find most visually pleasing?



Placeholder image

Appendix D: Focus Group Structure and Questions

The structure of recruiting and running out focus group sessions. These questions were asked to focus group participants. Respondents consisted of WPI students who varied in gender, age, and ethnicity.

Goal of focus groups: Receive feedback from a diverse group of students on video game characters and story.

Reaching out to participants: Email diverse groups on campus with a sign-up form and character descriptions (includes, pronouns, age, ethnicity, sexual orientation, etc.). They will be able to sign up for a time/character of their choosing.

Structure: Total session is 1 hour. There will be 2 groups going at once, each with one moderator and one note taker. Within each group, participants will discuss 2 characters (30-minute session per character). This will allow us to get feedback on 4 characters per session.

Questions:

1. How would you describe the personality of the character?
2. What elements of the character design and writing resonate with you?
3. Based on your experience, are there any changes or additions you would recommend for the character to enhance the overall gaming experience?
4. In terms of cultural identity, do you think the character is well-represented?
 - a. Do you see any elements of the design that appear insensitive?
5. Do you think the character has broad appeal across different demographics?
6. What items and furniture would you expect to see in this character's personal room?

Appendix E: Character Bible

Information of the playable characters. This was created by the writing team to develop and character document the characters. It was later sent to focus group participants so they could make informed decisions about the characters they wanted to discuss.

Vic



Gender & pronouns: He/him/his

Age: 22

Birthday: February 29th

Ethnicity: Korean

Occupation: Former actor and boyband member

General description: Charismatic and cocky, Vic thrives in the spotlight as one half of an identical twin heartthrob duo. When he's not singing to adoring fans, you'll often find him engaged in a tennis showdown against his brother, Van. While it may appear to be all fun and games, the constant pressure of maintaining his image leaves Vic torn between his public persona and authentic self. Will he stay in sync with his twin or daringly pursue his own aspirations?

Adjectives: Charming, arrogant, expressive

Verbs: Interacts with fans, concerned with his public image, games with his brother, teases everyone.

Background: Vic and his identical twin brother Van have always done everything together. These nepobabies grew up in the entertainment industry doing modeling, acting, singing, all the things. Their first big break was playing the same child on a soap opera (child labor laws) called Brighter Days. From then on, it was almost like they were the same person.

Most recently Vic and Van were in a boyband called the ScentSations best known for the hit single, Fresh Start Fantasies. After a long band tour, Vic and Van decided to take an indefinite break from the industry. Now back in Cleanland, the twins spend their days hanging out and playing tennis but are unable to fully escape the public eye.

Goal of route: Individuality and finding the courage to pursue things independent from his twin brother Van.

Needs to achieve goal: Embrace self-discovery and pursue his personal interests. Confront Van about his feelings and put aside the public's expectations.

Other details:

- Members of the ScentSations: Vic: the goofball, Van: the sensible one, Jax (Ajax) - to be changed: the bad boy, Clif (Cif): the quite/sensitive one, Kim (Vim): the heart throb
- Was on a sitcom called *Clean Humor* about two big families with back-to-back backyards that don't have a separating fence. The parents are friends, but the kids have a secret rivalry and are constantly messing with each other. Vic and Van played the mischievous identical twins who spoke in unison and always wore the same clothes.
- Serious acting is his true passion

Inspiration for the character:

- Zack Morris from *Save by the Bell*
- Lucifer from *Lucifer*
- Boybands/k-pop groups
- Spic and span

Example Dialogue:

- “With you being new to Cleanland, I thought I could answer any questions you might have about town. Or any questions you might have about me. I’m sure you have a lot of those.”
- “I love my fans, but some of them can be a bit... you know. But don’t worry, you seem like a normal one.”

Soap



Gender & pronouns: They/them

Age: 29

Birthday: November 22

Ethnicity: Indian

Occupation: Entrepreneur

General description: For Soap, business is their business. They grew up knowing that money equaled success, a lesson learned from humble beginnings. Soap's awkward charm makes them instantly likable, but their relentless focus on business has left them oblivious to the nuances of casual interaction. Now realizing that there's a world beyond work, Soap is on a journey to discover the simple joys they've been missing out on.

Adjectives: Resourceful, awkward, curious

Verbs: Be confident discussing business, be unsure in casual conversation, prepares for every event

Background: Soap N'Water grew up lower class and their family environment emphasized hard work and success. Soap's desire to fulfill their family's dreams became a driving force in their life. This sense of obligation fueled their unyielding pursuit of success, resulting in the

development of an overachiever mindset. This mindset served as a coping mechanism, helping them navigate the stress of striving for success. In their relentless quest, they sacrificed a balanced life and basic life experiences.

Goal of route: Stop living for success and start living life to the fullest.

Needs to achieve goal: Start engaging in the simple experiences of life. (A bucket list!)

Other details:

- Invests in Bubblecoin
- Owns the vending machines in the arcade
- Likes classic fairytales and fables
- Always get sanichai boba when they go to Scrubbing Bubbles

Inspiration for the character:

- Dr. Spencer Reid from *Criminal Minds*
- Dr. Temperance Brennan from *Bones*
- Castiel from *Supernatural*
- Soap

Example Dialogue:

- “Based on my research, shorter karaoke songs are best for audience engagement.”
- “Am I overdressed for a ‘casual conversation?’”

Mr. Tidy



Gender & pronouns: He/him

Age: 52

Birthday: December 1st

Ethnicity: White

Occupation: Chef and Restauranter

General description: Meet the head chef and owner of the Periodic Table. While the dining may be fine, Mr. Tidy is in a mid-life crisis. Between managing his restaurant, the challenges of finding culinary inspiration, and thoughts of his estranged ex-wife and teenage son, Tidy's got a full plate. Let him cook!

Adjectives: Driven, introspective, passionate

Verbs: Searching for inspiration, cooking up a storm, reminiscing about the good old days with S.C. Jenkins

Background: Mr. Tidy grew up in Cleanland with S.C. Jenkins and moved away to pursue his passion for the culinary arts. Along the way, Tidy got married and had a son. As Tidy delved into

the demanding world of professional cooking, his pursuit of greatness took a toll on his domestic life.

Working inconvenient hours strained communication within his family, breeding resentment from his wife and son. The guilt of being a bad husband and father caused Tidy to bury himself deeper into his work to avoid confronting his family. This cycle eventually led to divorce, with Tidy's Wife receiving full custody of their son.

After the Divorce, Tidy needed a change. A return to Cleanland to open up his own restaurant. Tidy decided to open his own restaurant, using his excellent culinary reputation to attract investors like Goctor and Pramble. He opened the Periodic Table to great success where he continues to bury himself in work. On rare occasions he can be seen shopping for ingredients at the market or visiting with his family friend S.C. Jenkins and his family.

Goal of route: Addressing burnout and embracing support.

Needs to achieve goal: Realize his lifestyle is unhealthy and allow himself to make new connections and grow existing ones.

Other details:

- Jesse is his sous chef
- Sings and cries in the Periodic Table walk-in freezer
- A fan of the SentScations
- Practically lives at the Periodic Table (sleeps on a cot in his office)

Inspiration for the character:

- Walter White from *Breaking Bad*
- Mr. Clean
- The Bear TV show

Example Dialogue:

- "The quality... It isn't good enough..."
- "Knock knock."
-

Lysa



Gender and pronouns: she/her

Age: 23

Ethnicity: Black

Occupation: Aspiring musician

General description: Forever the tortured artist, music has always been a sour subject for Lysa. Despite her dreams of pursuing a music career, resurfacing memories of her ex have left her with a broken heart and a broken spirit. Stripped of her self-confidence, she struggles to simultaneously stay silent and be heard. It's up to Lysa to face the music of her past, realize her talent, and step into the limelight!

Adjectives: Cool, superstitious, enigmatic

Verbs: Leans heavily into New Age beliefs and phrases in her speech and life. Gets anxious when someone mentions her music/singing. Is REALLY weird... but has such a thick aura of coolness no one bats an eye. Is obsessed with her ex and what she thinks about her. Will sometimes snap out of the heavy New Age vernacular to "explain in common terms".

Background: Coming from a long line of creatives, Lysa's life had always revolved around music. Her earliest memory was watching a VHS tape of her father's featured appearance on an episode of Soul Drain, where he performed live as the drummer in a special guest 5-piece band. Since that moment, she wanted to become a musician like her father, though he had long ago retired his drumsticks in hope of a more stable career for his family. Lysa's parents built their financial stability from rock bottom, and wanted their children to continue in the same vein. This well-meant intention took the form of strict parenting and traditional values, pushing Lysa and her older brother Ty to put immense effort into their studies. Lysa excelled in school, but always found herself drifting to creative activities, much like her older brother. They did not support their creative endeavors, telling them that music and art are a dead end street, and ultimately are not financially stable. Instead, their parents encouraged them to pursue STEM related careers, since Lysa and Ty "had worked so hard" for their grades... and they were paying for their tuition. So, Ty went to college, and when Ty's fashion dreams were suddenly realized by a talent scout from New York in his senior year, he dropped out to pursue his passions in NYC. Her parents were aghast and appalled at his decision, cutting off all contact and said that they had poured the money they had fought their whole lives for down the drain. This resulted in her parents tightening their grip on their daughter's life, and looked down upon any way Lysa deviated from their perfect vision of her.

This became increasingly hard on Lysa. In high school, Lysa began to question everything in her life, and felt as though she never truly belonged in her parents' religion. She believed in a higher power, but her sexuality was not welcome in their belief system, or her home. Thus, she dove deep into superstitions and New Age oddities, like tarot, crystals, and energies. She felt as if her passion, her identity, and her beliefs all conflicted with who her parents wanted her to be. So, she appeased her parents, got the STEM degree, told them about her authentic identity, and when things quickly went south, left. Throughout her whole life, Ty had been her rock, and he, after spending a few years in NYC with extreme success under the moniker Wave, offered to give her a place to stay in Cleanland.

During college, Lysa had met her now ex, and was instantly head over heels. Also a musician, she suggested that they form a band with a few of her friends. Things began to take a turn for the worse when practices began. Her ex was classically trained, perfect pitch and all, while Lysa was self-taught, and "played by feel". This resulted in a lot of clashing between them, mostly resulting in her ex being passive aggressive and slowly but surely shattering Lysa's confidence in her music. Ultimately, it resulted in Lysa getting removed from the band for someone with "more experience", and the band leaving to tour without her after she graduated. However, Lysa remained disillusioned; she was, no, IS convinced that they are SOULMATES, DESTINED to be together, even after their breakup.

Lysa is no longer in contact with her parents or her ex and instead lives a few streets away from her brother and his fiancé, but sometimes, she is reminded of the fonder moments with her family when she plays her keytar.

Goal of route: Gaining confidence in her music and not caring about what others think, especially her ex.

Needs to achieve goal: build up confidence in herself, get over her fear in performing in front of others

Other details:

- VERY superstitious
- Gullible
- Looks up to her brother and his fiancé, Olive
- Works at Wave and Olive's market stalls during the day
- Eats dandelions for good luck (just straight out of the ground)
- Music is REALLY WEIRD and only instrumental, but leaves people feeling obscurely moved....If abstract art were music

Inspiration for the character:

- Willow Smith
- Lysol

Example Dialogue:

- "I read about it in the Cleansmopolitan. It's not a tabloid. Unlike all the other fakes, Cleansmopolitan is real. It's helped me through most important life decisions... like what color I should wear to prom, if I should bother publishing my song, whether or not my ex still secretly loves me and writes my name with her fingerprints on her shower wall and then splashes off the words so no one knows, if I should scrape the mold off my raspberries and still eat them because I spent good money on them... The important stuff."
- "Your vibe seems... tangled. Perhaps your spirit requires a cleanse. No worry. I have a spare aged rosewater cricket asparagus energy purifying tonic on my person. Open wide..."

Roxi



Gender & pronouns: she/her

Age: 26

Birthday: September 21st

Ethnicity: Half white, half Chinese

Occupation: Town Hall Secretary, Future Mayor

General description: As the face of the town, Roxi has let it all get to her (objectively beautiful) head. In Cleanland, running the town is a family business, and Roxi is convinced she's next in line. I mean, how could she not be? She, like always, has it all, including others to do her work for her. However, a good mayor keeps an eye on everyone, and SC hasn't let this go unnoticed, and might have changed his mind...

Adjectives: aloof, arrogant, dramatic

Verbs: slays, needs things to go her way, nice to people to their face on the surface, judges others

Background: Ever since middle school, Roxi had always gotten whatever she wanted because she was the prettiest girl in the room. Her parents spoiled her throughout her childhood... and adulthood. She is an absolute daddy's girl – her father (SC's younger brother) pulled some strings to get her the secretary job at town hall. She also likes to facetime her mom over a glass of wine and gossip about their lives. Her parents are the two people that Roxi would never say anything negative about. Roxi cares about her cousin, Windy, and wants the best for them, but doesn't show it as much as she thinks she does. She also is sometimes critical of Windy "from a place of love", but only makes them feel worse about themselves. Her toxic bestie Clara has bullied Windy for as long as they can remember, but Roxi seems not to notice. In all seriousness, Roxi's only "close" friend is Clara, though she's a bit of a backstabber and leeches off Roxi. Also, while Olive and Wave were getting serious, Roxi tried to break them up by flirting with Wave just to see if she could. It didn't work, and she only recently has begun to feel like she might be in the wrong (she definitely is). Clara also tries to enable Roxi to make bad decisions, secretly burning with jealousy and getting satisfaction from her little failures.

Goal of route: Get back into SC's good graces and become next in line for mayor.

Needs to achieve goal: End friendship with Clara, reconcile with Windy, take responsibility for her actions, take more responsibility in the Town Hall, begins to be more genuine and humbler

Other details:

- Expensive taste
- Daddy's girl (loves her mom, too)

Inspiration for the character:

- Regina George from *Mean Girls*
- Oxi Clean

Example Dialogue:

- "Of course I'm the face of the town! I moisturize daily~."
- "I'm not *judging* people, I'm brainstorming constructive criticism. Feedback is essential to growth, after all."

Shawn



Gender & pronouns: he/him

Age: 23

Birthday: April 14th

Ethnicity: Filipino

Occupation: Unemployed

General description: In a town settled by the rich and successful, Shawn feels like the odd one out. Running away from a personal crisis has left him out of college, work, and most importantly, cash. Despite everything going on in the background, Shawn still manages to live in the moment and make friends with everyone he meets. Can he find a way to rake in some revenue, or will he have to say “paalam” to Cleanland?

Adjectives: Easy-going, magnetic, comedic

Verbs: likes westerns and anime, skateboards, gets st0ned, does graffiti art, takes it eeeasy

Background: His mother being a doctor and his father a nurse, they’ve always pushed Shawn to study hard and go to medical school. After all, the family lived in a big house in a nice neighborhood for a reason. Doesn’t Shawn want that same comfort? Maybe, but he wasn’t

gonna work for it. Shawn embodied “C’s get degrees” in middle and high school. Hell, he embodied “D’s get degrees.” This put him in conflict with his parents constantly—always pressuring him to do better, always nagging about learning the family’s Tagalog, always nitpicking on Shawn’s friends. The last part stings most. In high school, he befriended classmate Lysa, and the two have remained close. Through Lysa, he met her brother Wave, who taught Shawn graffiti art. Not wanting to be home with his parents, Shawn spent a lot of time skateboarding around and tagging places in graffiti. Shawn decided himself to take a gap year after high school, not wanting to deal with any more school, at least for a little bit. That meant constantly being home with his parents with a new topic to nag him about. In this time, Shawn picked up weed to take the edge off while at home. Shawn eventually went to college, for art, but the lazy gap year and lack of effort in high school caught up to him. He dredged through four years, constantly pressured by his parents to switch to nursing and feeling “behind” all his high school peers. After a bad break and a fight with his parents, he dropped out and dipped.

Goal of route: To have a stable source of income for the foreseeable future so he can stay in Cleanland

Needs to achieve goal: Recognize that he can’t give up and get out --- the people need him!

Other details:

- Wants to reconcile with family
- Wants to transition into full-time art

Inspiration for the character:

- Dolly Parton’s 9 to 5 and Other Odd Jobs
- Dawn dish soap

Example Dialogue:

- “Heyyyyy, no worries, be happy! Am I right? Ahaha... Ha.”
- “Man, why you being such a tryhard, mannnn? Yo, I was thinking, we could take a couple of these, grab a bite, and watch some Cleant Eastwood at my place? C’monnn, just for a little bit—my treat!”

Windy



Gender & pronouns: They/Them

Age: 21

Birthday: November 11

Occupation: ~~Gamer~~-Student and Journalist

Ethnicity: White

General description: Don't let Windy's nature fool you; despite being a very nervous and awkward person, they are always thinking about their passions... unfortunately they do to a fault. You can usually find Windy in the arcade gaming or at their desk writing for hours. Just make sure you remind them to take a break and take care of themselves every once in a while.

Adjectives: Introverted, skittish, focused

Verbs: I.e. Be shy around strangers, talkative when discussing passions.

Background: Whether it be being bullied by people at school or by Roxi's friends, Windy is severely lacking in confidence. They've fallen into a cycle of constant self-neglect and isolation from others. After the death of their mother, their life has only been on a negative spiral downwards. To drown out the constant pressure that is life, Windy turns to both gaming and

writing. Windy spends a lot of time on their own at the arcade or in their room, only leaving it for food and bathroom breaks. Windy tries to make connections with others, but it usually never turns out ok. Anyone who interacts with them needs a great deal of patience and understanding. They typically avoid others due to crippling anxiety and a fear of being judged as weird or annoying. With being the mayor's child, people give Windy too much attention, compare them to Roxi, or sometimes even just forget them all together with how much Roxi is the face of town hall. With the only connection they truly have being Roxi and SC, and with Roxi hanging out with her own friends and SC being busy running the town, things look very grim for Windy...

Goal of route: Realize the spiral of self-neglect and isolation they've thrown themselves into and start taking care of themselves.

Needs to achieve goal: Seeing that people are there to support them; self-reflection and addressing of the issues at hand

Other details:

- Windy is the writer for the local paper, but they don't want anyone to know.

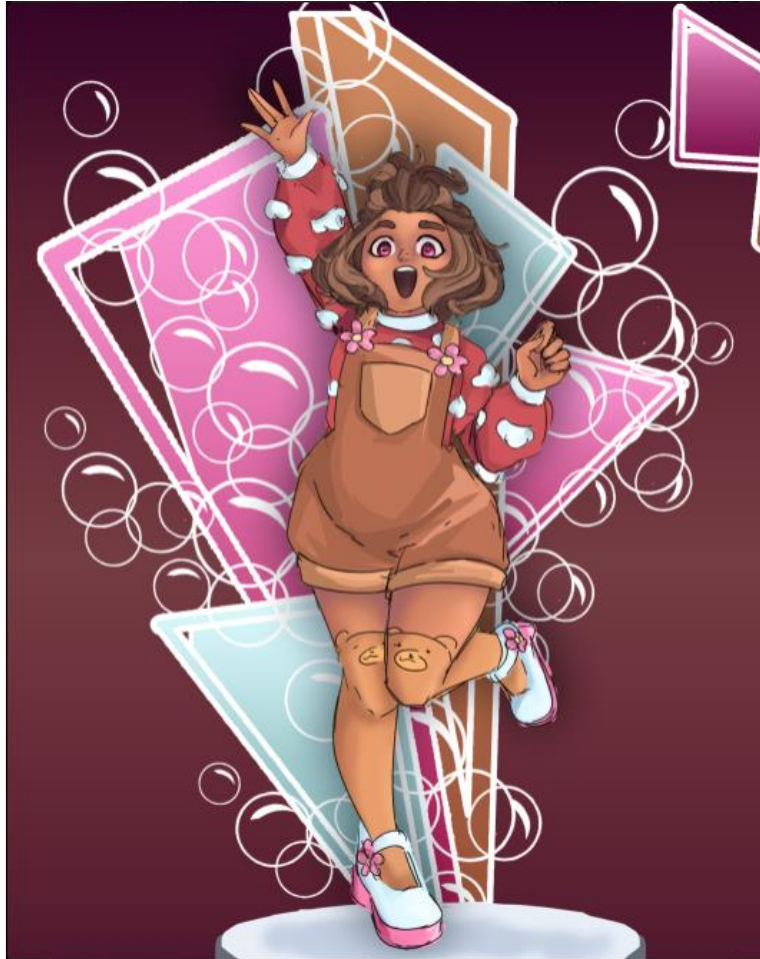
Inspiration for the character:

- Anime weeb
- *Beauty and the Geek Australia* season 3
- Windex

Example Dialogue:

- "I LOVE the arcade! There's this new game they just added, and I've been playing it for hours, and basically you play as a mouse trying to get pictures-- ...oh u-uh... Sorry."

Charming



Gender & pronouns: She/Her

Age: 24

Birthday: May 16th

Ethnicity: Mexican

Occupation: Barista

General description: Charming is as go-getter as it gets. When faced with any type of hardship, she always pushes forward and gives it 110%.

Adjectives: Bubbly, tenacious, optimistic

Verbs: Be nice and friendly to people, be a pushover, be overly optimistic

Background: Charming's biggest role model is her mama bear. Charming has seen her work hard every day in her life to ensure that Charming would live without the strife that she had to go through. Watching her mom work has rubbed off on Charming a bit too much. She's very hard working almost to a fault, never asking for breaks. This has led her to being a pushover, working hard for approval of others while leading herself to burnout. By needing to stay on top

of everything, she's been paying the price for the last couple of years, not realizing just how much overworking herself has been leading her to burning out.

Goal of route: Start to put herself forward more, take a break, and realize when people are taking her for granted

Needs to achieve goal: Someone to ground her

Other details:

Inspiration for the character:

- Charmin

Example Dialogue:

- "I hate when people say, 'It's like taking candy from a baby.' Why would anyone want to take candy from a baby!? Should babies even have candy...?"

Cain



Gender & pronouns: He/Him

Age: 25

Birthday: July 8th

Ethnicity: Half white, half Puerto Rican

Occupation: Kelvin Clean Model

General description: If there's one person all old ladies in town fawn over, it's Cain. He's always putting his best foot forward, helping just about everyone he can. When he's not at the gym trying to maintain his chiseled physique, you can see him modeling for Kelvin Clean. He can lift just about anything you throw at him, but he can't seem to lift his own negative self-image off.

Adjectives: Airheaded, thoughtful, jolly

Verbs: I.e. Be very encouraging to others, lifts so many weights, hypes up just about everybody.

Background: In the past Cain was bullied for his body, being larger than most of the people around him. The constant judging from others and the negative feelings he started to express towards his own self-image left him hating his own body. Cain, motivated by a hate for his own body began to work out extensively and use it as an extreme coping mechanism. After his glow

up, he was approached by recruiters for Kelvin Clean and began modeling. Now he's entered a position where everyone is looking at his body, and he constantly feels like he needs to upkeep his "perfect" physique. The constant pressure of his desire to look perfect has left him hating any imperfections he has of himself, trying to overcompensate his workouts, eating habits, and daily routines to make sure he won't slip back into being unattractive again.

Goal of route: Help Cain acknowledge and learn to take action against his body dysmorphia

Needs to achieve goal: Acceptance from the player, support in his own self-help practices, motivation from the player

Other details:

- The gym is basically his second home
- Grandmas' little goober

Inspiration for the character:

- Gain

Example Dialogue:

Appendix F: Example Character Route Event One

Example draft of a character route event one. Route event one is where the player is officially introduced to a playable character. The following text was written in the narrative scripting language Ink in the editor Inky.

```
//Ink file: vic_event1:
```

```
//Scene: Park
```

```
//Stage: Vic
```

```
~Name = "Vic"
```

Hey, you! Ooo, Is that a new hazmat suit?

```
~Name = "Janitor"
```

Uh-

```
~Name = "Vic"
```

Why don't we walk and talk? Let me show you my favorite patch of grass!

```
//VIC
```

Just pretend like we're friends and keep walking.

```
~Name = "Janitor"
```

What's going on?

```
~Name = "Vic"
```

I needed an excuse to get away from that fan. She wasn't getting any of the hints.

```
/*
```

Choice

1. You have a fan?

2. What do you mean by hints?

```
*/
```

```
//Choice 1:
```

```
+ [You have a fan?]
```

```
~Name = "Vic"
```

Of course I do. What kind of a question is that?

~Name = "Janitor"

Do you find yourself running away from fans often?

~Name = "Vic"

Hey, we're walking. I would never disrespect a fan by running away from them.

~Name = "Janitor"

Well, there are always exceptions...

//Choice 2:

+ [What do you mean by hints?]

~Name = "Vic"

Well, I said 'it was so nice to meet you' three times. Three! Didn't register.

//VIC

When I looked at the time and told her I had to leave, she pulled a, 'Wait, before you go...' Boom!
Trapped for ten more minutes.

~Name = "Janitor"

How long was she talking to you?

~Name = "Vic"

Don't know. Too long.

//***same text for all routes at this point

~Name = "Vic"

- I love my fans, but some of them can be a bit... you know.

//VIC

But don't worry, you seem like a normal one.

~Name = "Janitor"

I'm not.

~Name = "Vic"

Are you saying you're crazy?

~Name = "Janitor"

No, I'm saying that I'm not a fan.

~Name = "Vic"
 Not a fan? So, you are crazy!

//VIC
 Why aren't you a fan?

/*
 Choice
 1. I don't really know who you are.
 2. I don't really know what you do.
 */

//Choice 1:
 + [I don't know who you are.]

~Name = "Vic"
 You don't— I must speak to my publicist!

//VIC
 I'm Vic, the better half of the twin duo Vic and Van.

//Choice 2:
 + [I don't know what you're famous for.]

~Name = "Vic"
 You don't— I must speak to my publicist!

//VIC
 Let's see, I've done modeling, acting, singing, you name it.

//***same text for all routes at this point

//VIC
 - Have you ever heard of the ScentSations?

~Name = "Janitor"
 Doesn't ring a bell.

~Name = "Vic"
 It's the boy band me and Van used to be in. We had the hit single 'Fresh Start Fantasies.'

~Name = "Janitor"
Haven't heard of it.

~Name = "Vic"
Huh, maybe I should speak to my publicist.

~Name = "Janitor"
You said you used to be in the band?

~Name = "Vic"
Yeah, we started getting tired of the lifestyle and decided to move back here.

~Name = "Janitor"
I recently moved to Cleanland.

~Name = "Vic"
Now that makes sense. It's the only way to explain how you haven't heard of me.

~Name = "Janitor"
Or maybe you need a new publicist.

~Name = "Vic"
Yeah, maybe... Oh no.

~Name = "Janitor"
What is it? Germs?

~Name = "Vic"
Worse...

//Camera: Close-up on VIC

//VIC
Paparazzi.

//VFX: Camera shutter

//Camera: Back to VIC and PLAYER

//VIC
Earlier I told you to walk, but now I'm going to need you to run.

//VIC

I'll go this way, you go that way. Rendezvous at the tennis court. Go!

//FX: Fade to black
 //Scene: Tennis court
 //Stage: Vic

//VIC
 There you are! You didn't talk to them, did you?

/*
 Choice
 1. No, I didn't.
 2. I don't know.
 */

//Choice 1:
 + [No, I didn't]

~Name = "Vic"
 Thank goodness.

~Name = "Janitor"
 Why? Do you think I would've said something bad about you?

~Name = "Vic"
 Well, you're not a fan, so... the possibility was there.

~Name = "Janitor"
 I wouldn't do that.

//JANITOR
 But it might have been better if I had spoken to them.

//Choice 2:
 + [I don't know.]

~Name = "Vic"
 What do you mean you don't know?

~Name = "Janitor"
 It all just went... dark?

~Name = "Vic"

Huh. Well, do you think you said something?

~Name = "Janitor"

Probably not, but it might have been good if I had.

//***same text for all routes at this point

~Name = "Vic"

- What do you mean?

~Name = "Janitor"

A picture with no context? they can speculate whatever they want.

~Name = "Vic"

Suds, that's what I was trying to avoid.

/*

Choice

1. I thought you'd be better at this.
2. What do you think they'll speculate.

*/

//Choice 1:

+ [I thought you'd be better at this.]

~Name = "Vic"

I've never had to deal with this sort of speculation. I'm always with Van.

~Name = "Janitor"

Always? Are you not allowed to do things without Van?

~Name = "Vic"

I am, but that's not what the fans want. They want to see the duo.

~Name = "Janitor"

What do you want?

~Name = "Vic"

To play tennis. Van can't play because he hurt his shoulder.

~Name = "Janitor"

Is there no one else you can play with?

~Name = "Vic"

Nope. Anyone else and it'd be too easy.

~Name = "Janitor"

You don't know anything about me. I could be a professional tennis player.

~Name = "Vic"

You're right, I should never make those kind of assumptions...

//VIC

Janitor.

//Choice 2:

+ [What do you think they'll speculate?]

~Name = "Vic"

Probably that we're dating. Lucky you.

~Name = "Janitor"

Lucky me? I don't want people to think we're dating.

~Name = "Vic"

Sure, whatever you say.

~Name = "Janitor"

I've got enough going on in my life right now.

~Name = "Vic"

You're just saying that because you would never have a chance with me.

~Name = "Janitor"

You've got it wrong. You would never have a chance with me.

~Name = "Vic"

Oh really? And what makes you so special?

~Name = "Janitor"

Not much, but I don't date people who only love themselves.

~Name = "Vic"

I love Van too, I mean, have you seen him? He's so handsome.

~Name = "Janitor"

I rest my case.

~Name = "Vic"

That was a joke. Do janitors know what those are?

//***same text for all routes at this point

~Name = "Janitor"

- How did you know that I'm a janitor?

~Name = "Vic"

Besides the hazmat getup? When I went "Uh oh" and you're first thought was "germs!"

//VIC

It was pretty obvious.

~Name = "Janitor"

Yeah, germs have been on my mind a lot.

~Name = "Vic"

You know what's been on my mind a lot? Getting caught with you by the paparazzi.

//VIC

Something I really hope doesn't happen again...

~Name = "Janitor"

Is this one of those hints you give when you want to get away from a fan?

~Name = "Vic"

It would be, but you're a hater.

~Name = "Janitor"

I'm not a hater, I'm just not a fan.

~Name = "Vic"

Ouch. They say the opposite of love isn't hate, it's indifference.

//VIC

Now I must recover from my bruised ego.

//VIC

I'll see you in the presses, Janitor.

-> END

Appendix G: Character Voice Descriptions

Sample character voice descriptions created by the writing team to develop the characters and provide instruction for voice actors.

Mr. Tidy

Voice: Deep and commanding. Grumbly, grounded.

Tone: Passionate, humble, jaded, contemplative, controlled, sarcastic, forthright

Speech patterns: Blend of vulnerability and strength. His speech is slow and deliberate, conveying a sense of authority and certainty. He articulates his words with purpose. Typically maintains a stoic, confident demeanor, but in moments of vulnerability he speaks with a more gentle, almost hesitant tone, often trailing off at the end of sentences.

Voice comps: Walter white (Breaking Bad), Castiel (supernatural)

Soap

Voice: Clear and precise, with a hint of intensity. Intelligence and inquisitive. flat.

Tone: Forthright, logical, earnest, sincere, formal, naive, informative, candid, sensible, supportive

Speech patterns: Timid and uncertain in regular social situations but find their strength and voice while being informative and discussing business: “business mode.” In business mode they speak with a clear, articulate tone, often using complex vocabulary and technical terms. There is a lack of intonation and emotion, giving the impression of someone who is more comfortable with facts and logic than with social cues. Around people they are comfortable with, speech is a casual version of business mode with earnestness and sincerity. They speak clearly, without much rhythm (relatively flat/monotone), and at a faster pace.

Voice comps: Dr. Spencer Reid (Criminal minds)

Vic

Voice: Smooth and youthful, with a hint of cockiness and mischief. He speaks with a casual, almost breezy tone, effortlessly sliding from one word to the next. There's also a warmth and sincerity in his voice. Raspy, abrupt, musical.

Tone: Animated/lively, confident, effortlessly charming, playful, sarcastic

Speech patterns: His delivery is often quick-paced, reflecting his sharp, sarcastic wit. He enunciates and speaks with clarity, adding to his charisma. Variations in tone/inflection to convey sarcasm or playfulness. Minimal pauses or hesitations. Uses succinct phrases and clever one-liners to convey his thoughts. Can also convey seriousness or concern when faced with more dramatic or emotional moments.

Voice comps: Zach Morris (Saved by the Bell), Kendall (Big Time Rush) - most band members can be referenced

Additional voice information: Vic's experience as an actor and singer contributes to his enunciation.

Appendix H: Rough Concept Sketch to Character Model

Examples of modeled characters and their rough concept sketch to illustrate the first and final steps of our concept pipeline.

Ambar “Brandy” Barr



Charming



Appendix I: Character Sprite Expressions

Examples of different characters' emotive sprite sheets in the Showfest build, which will be largely expanded on in the final published build.

Roxi



Lysa



Appendix J: Technical Art Pipelines and Tools

Link to all pipelines and tools developed by the technical art department. This information is accessible to anyone with a wpi.edu email.

[Clean Sweep Project](#)

[pipelines](#)

[tools](#)

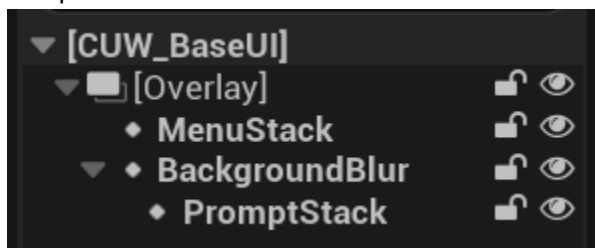
Appendix K: Common UI Documentation

Documentation created by the programmers to allow the team to fully understand common UI.

Custom Classes

CUW_BaseUI

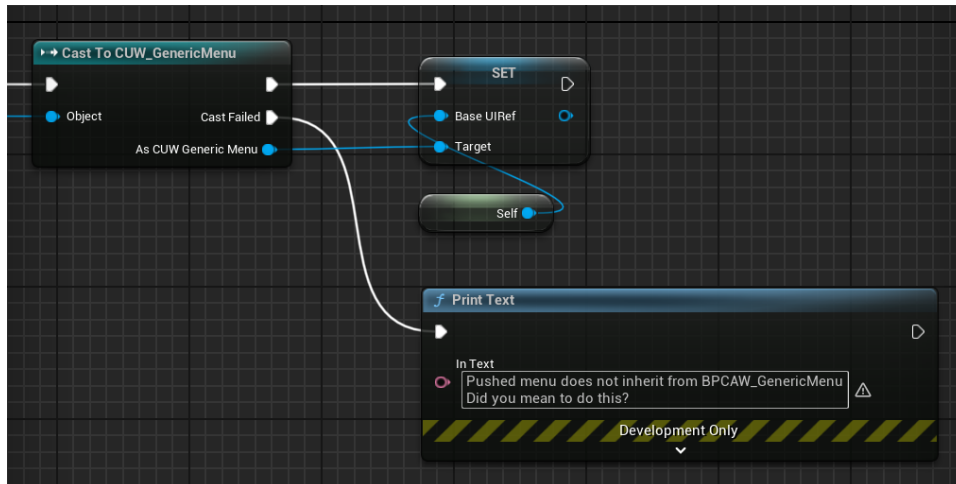
Base UI is the holder for all common activatable widgets. Right now, we have functionality to reuse this, but if you need to create your own, take a look at the implementation. Let's take a look at its individual components.



- Overlay
 - This is just a normal overlay which holds all elements. Overlays just allow for widgets to be stacked on one another
- MenuStack
 - This is a CommonActivatableWidgetStack. This holds all Menus. It functions as a normal stack. Menus pushed on are put on the top, and you can pop menus off the top of the stack. **The menu at the top of a stack is the only one visible.** Menus underneath the top are hidden. When on top they are activated, when hidden, they are deactivated.
- BackgroundBlur
 - This is both for visual clarity on blocking out the background for choices and for functionality. Since the two stacks are separate, **they can both be visible at the same time!** This does come with some minor headaches depending on your implementation. When routing from one button to another, both stacks can be navigated to. This leads to having a prompt on top of menu stack that can be navigated back to and thus create more prompts or unintended functionality. The background blur prevents navigation from the Prompt Stack to the Menu Stack
- PromptStack
 - This is another CommonActivatableWidgetStack. It's the exact same as the menu stack except it holds onto prompts.

Now that you know what everything is, let's go into functionality!

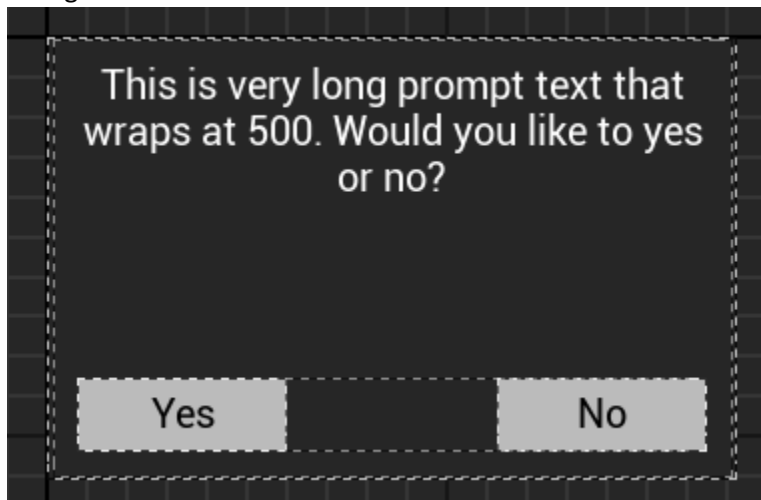
There are two functions: PushMenu and PushPrompt. PushMenu simply pushes a widget to the top of the stack. It also tries to cast the widget being pushed onto the stack into a CUW_GenericMenu and save a reference to the BaseUI onto the GenericMenu.



Push Prompt actually does something a bit different. This function takes in an owner which should be a GenericMenu, the prompt text, and the index of the prompt. It also binds an event to when the widget is deactivated to handle the background blur and enabling/disabling the menu stack. Now what does all of this mean?

CUW_GenericPrompt

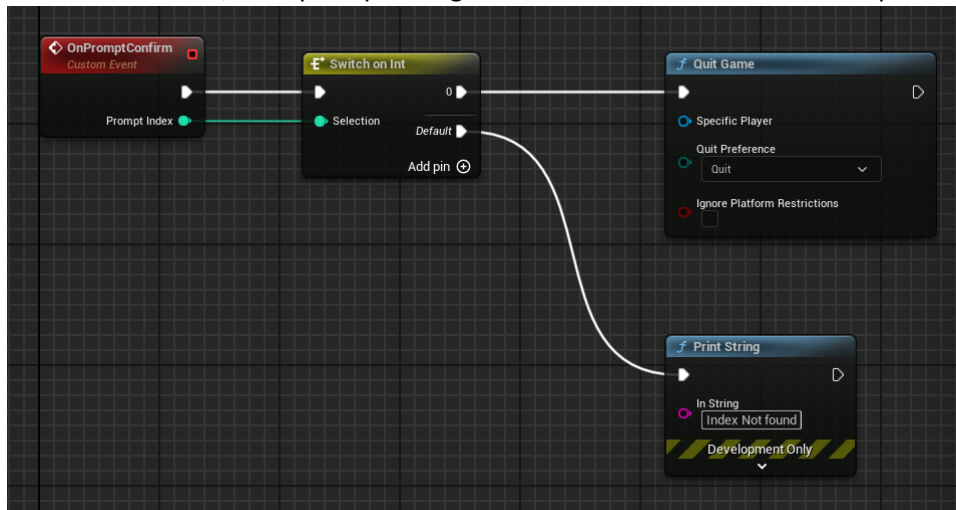
The prompt is actually something I already created so we can have a universal prompt design and application everywhere. The index for each prompt that a Generic Menu creates should have a unique index. More information about why and what we do with it can be found in CUW_GenericMenu. The prompt has two options, yes, and no. Unless we want to, those are the only two options. We could pass in different text for yes and no, but the function doesn't allow for that. Let me know if we'd like to change this.



CUW_GenericMenu

GenericMenu is a class I created that all menus should inherit from. It saves a reference to the BaseUI in case the menu you create needs to have a reference to it. The Generic Menu also has an additional function, OnPromptConfirm. This function has an index as an input. When a prompt confirm is clicked,

this function is called by the prompt. Normally, a delegate should be passed into the prompt when creating, but that requires C++, and that's stupid. So, this is a worse, but effective alternative. As mentioned earlier, each prompt a single menu creates should have a unique index. Here's an example:

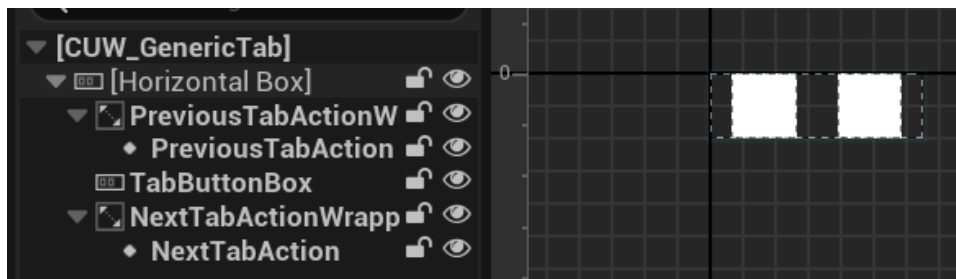


This OnPromptConfirm receives the index from the prompt (the same index we created the prompt with), and calls the function QuitGame when confirm is hit on the prompt. When “No” is clicked no a prompt, you’d think OnPromptCancel would run, right? Wrong! This function is currently disconnected because in most cases, at least most I can think of, canceling a prompt should just close it, and the prompt does just that.

In order to use the prompt system, your menu MUST inherit from CAW_GenericMenu!!!!

CUW_GenericTab

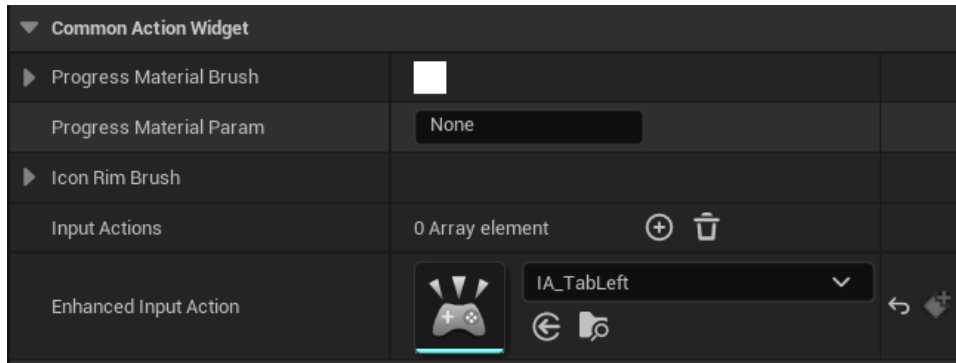
If you’re looking to create a tab system, there’s one resource you should look at. This is something I created for general tabs that go left to right. CUW_GenericTab is a tab system that has a couple of important things we should look at:



Above we can see this is just a basic horizontal box with 2 CommonActionWidgets wrapped with scale boxes along with a container for tab buttons. Let’s discuss everything important about this class.

Common Action Widget

A CommonActionWidget is built into common UI. It is a very useful tool for gaming with multiple platforms. It allows for immediate switching of icons for different inputs. For example, with the tab system, one would want q and e to be displayed indicating to the player that they can use those keys to change tabs while on controller, they’d want left and right bumper to indicate changing tabs. Common Action Widgets automatically switch the second any input is detected from the player.

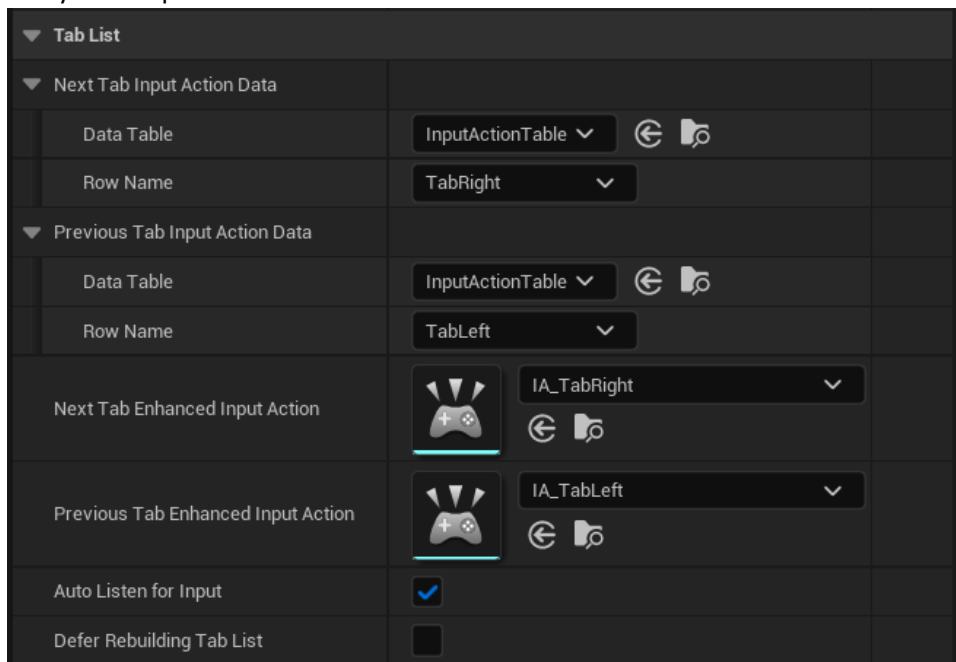


In the specific details for a common action widget, the only thing we need to modify is Enhanced Input Action, i.e. what action we want to use for this common action button. We might want IA_Interact for whenever the player needs to interact with something or, in the case of the tab system, IA_TabLeft to show this is the button the player should click to tab left.

One important thing to note: If using Enhanced Input Actions (like we are), if any inputs are consumed before making it to the UI like e mapped to interact and go right in tabs, the button won't display because the widget will think the button is unavailable and will display nothing.

Common Tab list Widget Base

The Generic Tab is actually a child of a common UI element called “Common Tab List Widget Base”. This has a couple of really neat features that we will make use of. First and foremost, the Tab system links to a **widget switcher!** This is **IMPORTANT** and will be discussed further in the functions section. Now, let's analyze the specifics of this class:



First, there's space for the Next and Previous Tab Input Action Data and the Enhanced Input Action versions of both. The Action Data versions are currently not in use as we have switched over to using Enhanced Input Actions for UI control. All you'd need to set is what action triggers tab left and tab right. Next is Auto Listen for Input. This allows the Tab System to automatically listen for the input next and previous tab actions we give so the player can hit the buttons anywhere, and it'll swap tabs. Finally, Defer Rebuilding Tab List is just for rebuilding the tab list to the next tick, so if you'd want to add

multiple tabs at once, you can just wait until the next tick, so it only needs to run once. This isn't checked now, but it shouldn't cause significant performance drops with it off.

There's also a number of functions that are built into the CommonTabListWidgetBase. We will go over the important ones as we continue to cover CUW_GenericTab.

Custom Functionality and Built In Functions

There's a number of specific functions to the Generic Tabs:

Refresh Next/Previous Visibility:

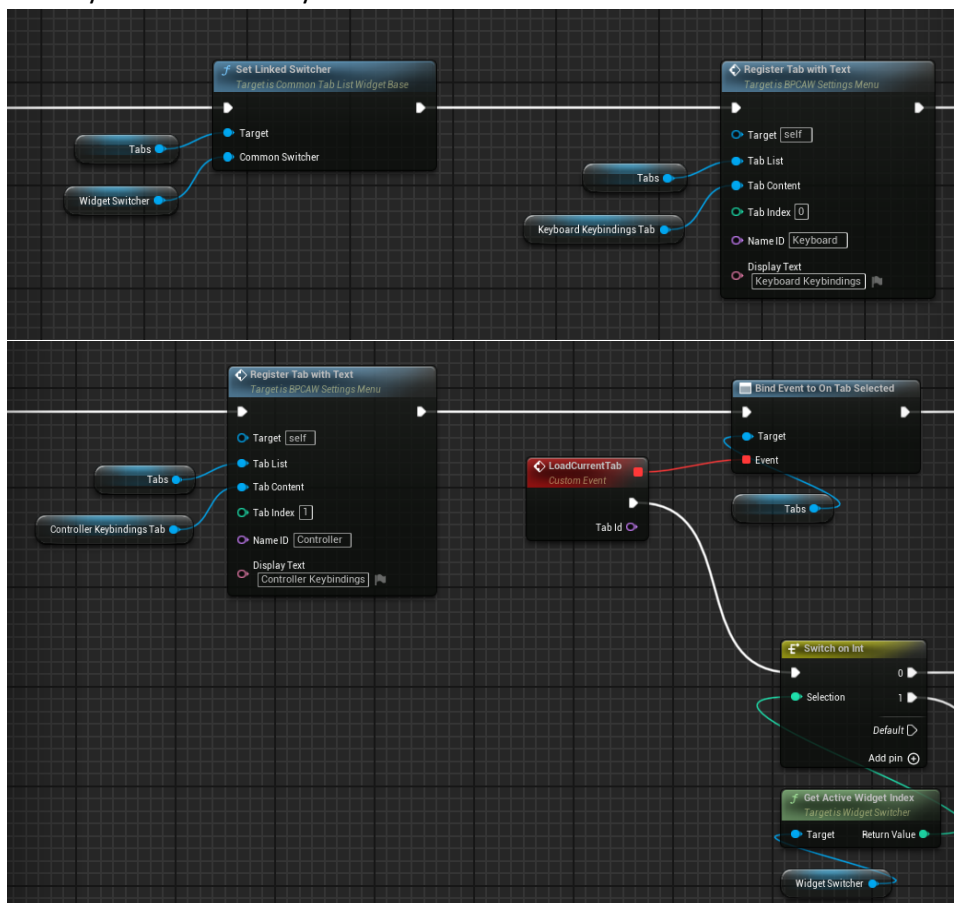
This is a simple function that gets run every time a tab is selected or removed. If there's only a single button visible, this disables the visibility of the common input action widgets because they shouldn't be visible for only a single tab.

Update Tab Styles:

This adds a button to the tabs, sets padding and styles for the button, and calls the function above.

Register Tab:

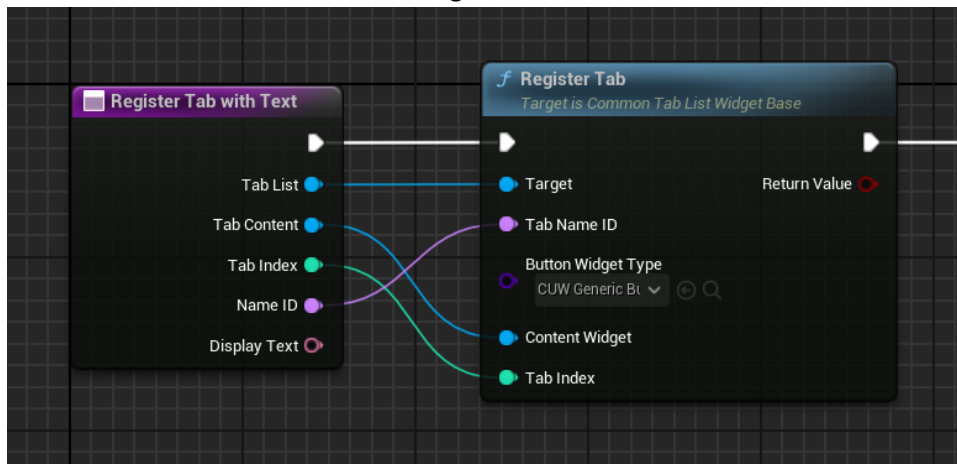
This function is not an overridable one, but one we call every time we want to register a new tab. In order to fully understand what we're doing when we register a tab, let's look at the settings menu which actually utilizes the tab system.



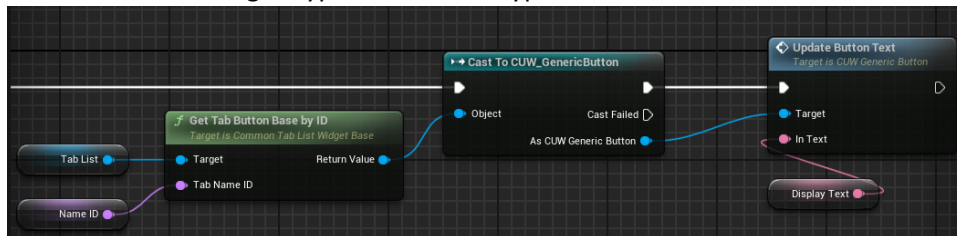
This is run off of the Construct event of the settings menu. First, we run the function SetLinkedSwitcher. This links the Widget switcher in the settings menu (the two tabs are for keyboard controls and controller controls). Next we run the function Register Tab With Text. This is actually a custom function that runs a built in function. There's a couple of things we should take note of. First, we pass in the tabs,

next the specific tab's root content, next a unique index which should increase from 0 upwards, a unique name ID, and the display text of the tab button. Finally, we bind a function to the on Tab Selected dispatcher that is run by the Tab system whenever a tab is switched to. On the settings widget, what we do is load the keys for the correct tab and refocus correct widget. More on what that means can be found in the What makes a Common Activatable Widget So Cool? Section.

Let's now take a closer look at the Register Tab With Text function:



First we can see that this actually just calls the Register Tab function. In order to register a tab, we need a unique name, a button type, the content of the tab, and the index of the tab. It's important to note that the Button Widget Type **MUST** be of type Common Button Base. Normal buttons won't work!



Once the tab is registered, we find the button by its ID and update the text of the button to whatever we need it to be. Registering a button does not automatically set the next, even with the name ID. We have to set it ourselves, so remember to do this if you create your own custom tab system!

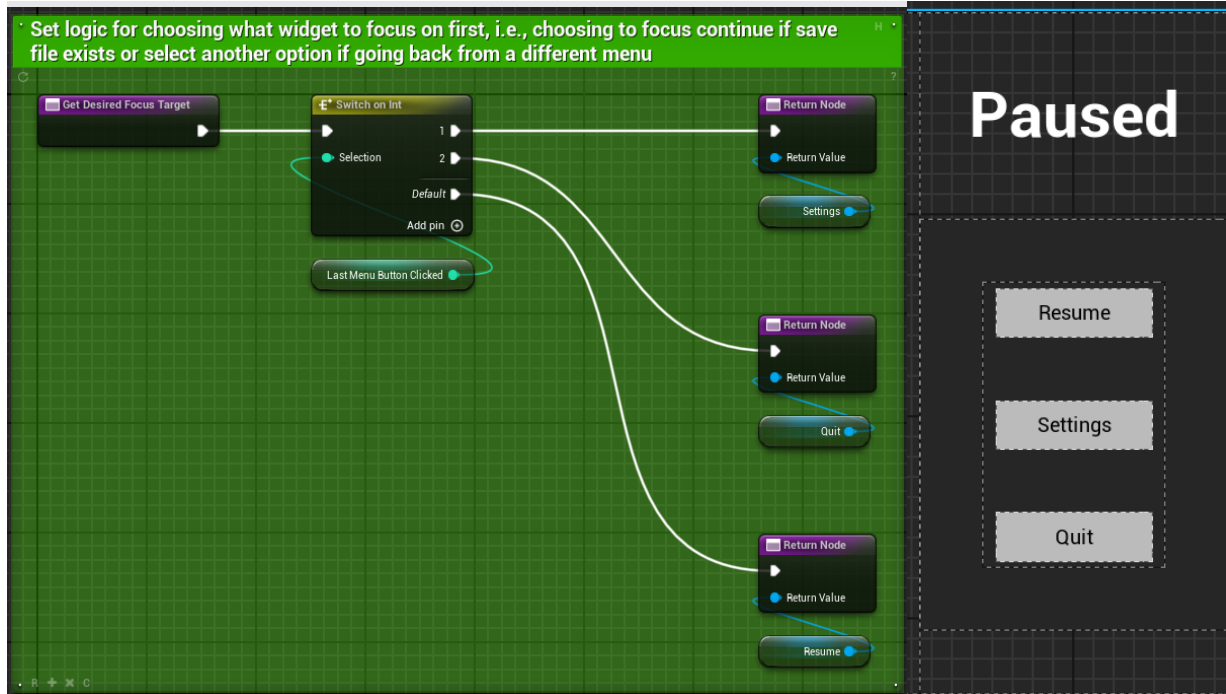
What makes a Common Activatable Widget So Cool?

Well, for starters, they can be **Activated**. What does that mean?

When put on a stack, the widget is considered activated. When taken off the stack or when not on the top of the stack, the widget is considered deactivated. This adds two new events, onActivated and onDeactivated. So whenever activated, on activate runs, and when deactivated, onDeactivated runs. Very simple!

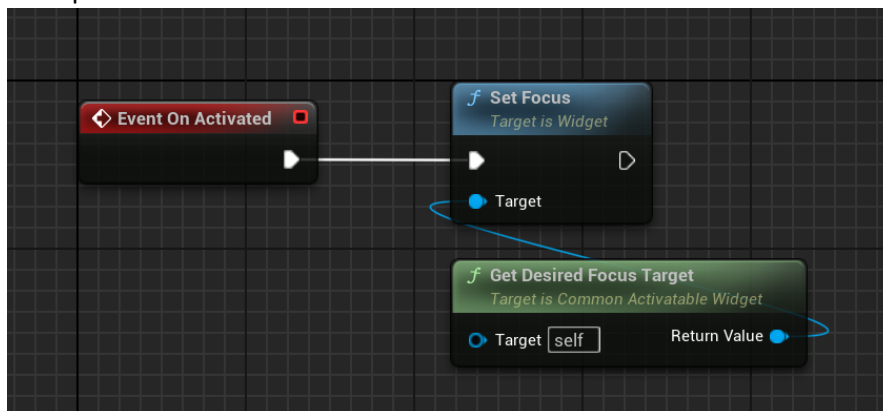
There's also a couple of new functions that Common Activatable Widgets add as well. These are normal functions, not events. The only 2 we will use (there's only 3 and one doesn't matter to us) is BP_GetDesiredFocusTarget and BP_OnHandleBackAction. There's are built in by Unreal Engine and are pure functions.

GetDesiredFocusTarget helps us with navigation and is essential for controllers so they know what menu button to default to. When we want to focus onto a button/target, we use GetDesiredFocusTarget. This is normally used to select buttons that we want to highlight by default when using the widget. For example, with prompts, we may want to highlight the yes button by default. You can also add logic to selecting the button. Here's an example:



In this example, when a certain button is clicked on the main menu, it sets this LastMenuButtonClicked variable. Since the widget isn't destroyed when deactivated, it still retains all variables and states it had before deactivating. Use this if you want to go back to a certain widget after clicking it. I used a switch, but there's tons of other ways to set up logic.

Now that we have this function created, how do we go about setting focus? It's simple! There's a function actually called, Set Focus! Usually on the OnActivated, you should call this function. Here's an example:



The other function, BP_OnHandleBackAction is called when the Back Action is clicked. A Common Activatable Widget can handle **back actions**. In the details panel of a Common Activatable Widget, there's two settings you can enable. The first is isModal. This setting allows you to actually use the back

action. Sometimes the back action doesn't work, so if it doesn't, just enable `isModal`. The other option is `isBackHandler`. This function needs to be enabled in order to call the `BP_OnHandleBackAction` function. One important note! For the `BP_OnHandleBackAction` function, the default functionality is to deactivate the widget. If you want to add custom functionality, you can override, but you also need to make sure that the function returns true. If it doesn't return true, it will call the C++ function and deactivate. This can lead to unintended functionality so make sure you know what you're doing!

Other Classes That Are Important

There are numerous other classes that are very important and make `CommonUI` work. Below I'll cover a lot of them, but for starters, there's one thing that is super important for making our game more cohesive and unified, and it saves a lot of headache later down the line. That is `Styling`!

Styling

`Styling` is hugely important in `CommonUI`. There are 4 styles: `Common Border Style`, `Common Button Style`, `Common Text Style`, and `Common Text Scroll Style`. I think it's obvious, but `Border Styles` are for `Common Borders`, `Buttons for Buttons`, and `Text styles for text`. The only unobvious one is the `Text Scroll Style`. That one is a little confusing. `Styling` is useful as it allows us to make changes to multiple different UI elements all at once with only a single style change. It also allows us to make multiple styles for different common elements and easily drag and drop them into any element we use. One thing to note is that styles are essentially data blueprints, so you can remove every tab but the details panel when viewing a style. Let's go over every single text style in quick detail!

Border Style:

`Border styles` are used for, well, borders. They are essentially the background of widgets. You can either use an image or a color for the border. There're also specifics about the background like how do we draw it as, but that's something you should look up and I won't cover here.

Button Style:

`Button styles` are the biggest style we have. We can assign a single material to the button if we want to, but we won't be using this. Also the tooltip incorrectly says that it creates a drop shadow, but that is not the case. There's a TON of properties so I'm gonna go rapid fire: We can set minimum width and height for the button; normal, hovered, selected, selected hovered, and disabled text styles (read below); pressed, selected, locked, hovered, selected hovered, and locked hovered sounds. We can also set `Normal base, hovered, and pressed, Selected base, hovered, and pressed, and disabled button styles`, like what image or color to use for those specific button configurations.

Text Style:

`Text styles` allow us to change `CommonText` elements. This includes changing the font. `Unreal Fonts` have many options including font family, typeface, size, letter spacing, skew amount, font material, and outline. You can also change the color of the text, shadow (if you decide to use one), margins, and strike brush (if you decide to use one). For more info on strike brushes, google it. Even I don't fully understand them.

Text Scroll Style:

Last but not least, `Text Scroll Style`! For text that gets cut off, `text scroll styles` allow for that text to scroll from left to right before being looped back to the beginning. There are only 5 properties on `text scroll styles`: `speed, start delay, end delay, fade in delay, and fade out delay`. These properties are all self-

explanatory. This is mainly for text that is too long for a specific widget. This can be very useful in certain cases, but hopefully we won't have any cases where text does not fit!

Common Text

Common text is what you should use for all text assets within Unreal. There's a couple of reasons for this: with styling it allows for easy reusability, it allows for consistency across all Widgets, and it is more customizable.

Common Button

Common Buttons are buttons that utilize the button styles within Unreal. Right now, we have a custom class we created that you should use. It's called CUW_GenericButton. They also have custom code to support showing the hovered style for keyboard and controllers as those aren't natively supported, sometimes. It can be a bit buggy, but this button eliminates those bugs. It also requires two styles, one normal one, and a "keyboard" style that has the hovered state on the normal state. Also don't set buttons to selectable. It looks bad and keeps them "hovered" when off.

General Useful Things to Know

- Navigation works visually. It tries its best to navigate to the next element visually present. This can be an issue if you have a button too far away or below a scroll box as it may jump out. Try setting the scroll box's Navigation Destination to center in this case! You can also modify the navigation settings at the bottom of the widget, although it doesn't really work too well
- DO NOT SET A UI MAPPING ON A WIDGET!!! When you do, it auto removes it once the widget is removed. This **only** works if you want every mapping to have input and you don't add it yourself. If you have custom navigation that you want to enable for one widget then by all means go for it but if not, don't have it there

Appendix L: *BURST UR BUBBLE* Lyrics

The lyrics for a combat song, *BURST UR BUBBLE*, which serves as an example of how we themed our music to the project through goofy cleaning-centric lyrics.

BURST UR BUBBLE

VO: "Warning: Hazardous chemicals, contact may result in injury or death"

When I walk into the room,
Depend on impending doom.
Death comes carrying a broom:
One sweep wonder!

Water pressure getting high.
pH reading alkaline.
Tooth for tooth and lye for lye.
Spotless showdown!

Your deep clean is overdue.
Disinfecting rendezvous.
I will wipe the floor with you.
It's a floodbath!

Caution's knocking on your door.
(No) sign of life on the wet floor.
Call cleanup on aisle four!
Your time's washed up!

Did that sink in or will you need to soak?
How much can you bite off before every chew begins to choke?
See me (see me waiting) as you dive into the deep to drown.
This is how I'll burst your bubble no-o-o-o-ooooo-ooooow...

(SO SAY GOODNIGHT!)

Say your prayers to-night you're cooked, you're on the cha-ain!

(SO SAY GOODBYE!)

To the world before I scrub you like a stain!

(SO SAY GOODNIGHT!)

Swirl and spiral as I flush you down the dra-ain!

(SO SAY GOODBYE!)

Hello, welcome to your wo-o-orld of pain!

This is how I'm gonna burst your bubble.
 You said you could tango with the trouble.
 This is how I'm gonna burst your bubble.
 This is how I'll do it.

How more potent can I get?
 Turn't up like a turbo jet.
 Three in one: a triple threat.
 Locked and loaded!

Let me even out the odds:
 Turn the tide without the pod.
 Smited with the mop of God!
 Done and dusted!

Your demise is very near:
 No one left to save you here.
 I will make you disappear!
 Total wipe out!

Payback is the debt I owe,
 Roll the punches blow-by-blow.
 I'd step back before I go
 Soa-per-no-va!

Can you stand the soapy lemon scent?
 (You can) try to hide beneath the rug but I will find you, I am hellbent.
 You can (you can scrub it) but you'll never get the blemish out.
 This is how I'll burst your bubble no-o-o-o-ooooo-ooooow....

(SO SAY GOODNIGHT!)
 Say your prayers to-night you're cooked, you're on the cha-ain!
 (SO SAY GOODBYE!)
 To the world before I scrub you like a stain!
 (SO SAY GOODNIGHT!)
 Swirl and spiral as I flush you down the dra-ain!
 (SO SAY GOODBYE!)
 Hello, welcome to your wo-o-orld of pain!


This is how I'm gonna burst your bubble.

You said you could tango with the trouble.
This is how I'm gonna burst your bubble.
This is how I'll do it. (....UNH!)

Spray them down!
Pow-wer-wash,, like a pow-werhouse!
I smell your fear...
Let me make it crystal clear!


This is how I'm gonna burst your bubble!
You said you could tango with the trouble!
This is how I'm gonna burst your bubble!
This is how I'll do it.
This is how I'll do it.
This is how I'll do it!
This is how I'll do it!

Appendix M: Clean Sweep Computer Science Project Presentation Poster



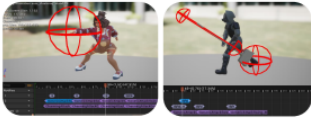
Clean Sweep

Nelson Pires (CS/IMGD BS), Austin Hyatt (CS/IMGD BS), Jessica Liano (IMGD BA),
 Conor Dolan (IMGD BA), Renee Cullman (IMGD BA), Zachary Adams (IMGD BA)
 Advisor: Professor Gillian Smith (CS/IMGD), Professor Farley Chery (IMGD),
 Professor Karen Stewart (IMGD), Professor Rodney DuPlessis (IMGD)




Clean Sweep is a 3D, comedic, story-driven, action-RPG where you play as a janitor in a world where everyone is a humanoid cleaning product. It's the start of summer, and you've recently inherited your late aunt's mansion. Germs have been appearing rampantly in town, and it's your job to clean them out. Meet people in town, assemble a cleaning crew, take on janitor jobs, and uncover the secrets behind your aunt's mansion which is filled with germs!

Hitbox System



To speed up the development of our game's combat, we developed a flexible system that can apply hitboxes to any animation. This system uses capsule-shaped hitbox actors which are adjustable in terms of size, rotation, offset, and damage attributes. These hitboxes are positioned on a timeline and linked to specific joints of the attacker's skeleton, so they follow the movement of the animation. The system allows for multiple simultaneous hitboxes to create more complex shapes. To make implementation easier and more accurate, we also implemented a feature within the engine to draw and visualize hitboxes, as seen above. When the hitbox overlaps an enemy's collider, it will trigger logic for damage, knockback, and hitstun. Hitboxes can also carry a variety of effects such as slowness and poison that are applied to the hit entity.

Enemy AI




Our enemy AI system uses a token-based approach, where enemies are given permission to attack based on their position, the number of enemies surrounding the player, and attack cooldowns. There are separate managers for each type of enemy. Features such as changing intensity and proximity-based token stealing give finer control over group behavior. Enemies are individually managed using behavior trees. Enemies will randomly roam before spotting the player using Unreal Engine's AI perception system. Melee enemies move to a position along the player's radius where they wait to receive a token while ranged enemies can request from anywhere. Upon receiving a token, enemies transition to their attack branch, execute their attack, and enter a cooldown phase before they can attack again.

Systems

There are numerous systems that run simultaneously that allow for Clean Sweep to function. Briefly, we'd like to discuss 2 main systems, the dialogue system, and the quest system.


Dialogue System

The Dialogue System was created using the plugin Inkpot, a game dialogue scripting language. Through ink, we created functions that perform various visual effects and integrated other systems into it like the quest system, the item system, and the scripting flag system. In addition, our game has full voice acting and character sprites with emotions. The ink files parse strings in the file for critical information like character name, emotion, quest, and file number; use tables to lookup locations of sprites and voice lines, and load soft references to the sprites and voice lines. Overall, this allows for a memory efficient method to find dialogue and sprites as the only sprites and voice lines loaded are the ones being requested to load.



Quest System

The Quest System is the most involved system within the entire game; it listens to various systems and actors since many different actions the player performs can update a quest. To solve this without having the quest system bind to every actor in the game, we utilized a decoupled event system. Any object in the game can send an event containing a tag and data. The quest system uses these tags to update any quests listening for it. They also implement a queue system for displaying updates, calling quests that receive multiple updates in order to limit how many notifications pop up for the player upon quest progress achieved.



UI and Navigation

Clean Sweep was designed to work fully with both controller and keyboard. Setting up navigation in Clean Sweep through UI was a difficult task. We utilized Unreal Engine's plugin, Common UI. Common UI allowed the game to listen for changes in input, updating certain input actions to match with the current input, i.e. changing press E for interact on keyboard to press A for interact for controller. Common UI allowed for global styles, easier formatting, controller and keyboard navigation support, and better organization of UI widgets. It allowed for menu stacks which created better flow on navigation and creating new widgets.

Debugging and Efficiency

With creating a massive 3D video game, we were constantly looking to find ways to optimize and debug our game to ensure that it runs at a high, constant frame rate. Our art team spent time optimizing both character and environmental models, employing processes like retopology.

On the tech side of the project, we dug into performance issues and frame drops. Unreal Engine provides numerous tools to help debug and find the issues on both the GPU and CPU where the game can be slowing down. We used GPU profilers to generate information dumps on what parts of the game were slowing down the render thread. We also looked at displaying live graphs of each thread to see what parts of the engine were slowing down and where we were experiencing bottlenecks.

Unreal Insights, a profiler built into Unreal Engine, gave insight into the game thread, render thread, and all background worker threads. It gave clear visuals on how much time each thread was taking per frame, where they were stalling, what classes were using the most frame time, and how that impacted the frames per second. Through using these debuggers, we were able to find bugs and problems with various systems including our lighting and player manager. The image to the right is from one of our trace files that allowed us to visually debug and correct a mistake with our Player Manager actor component.

