Raveling Dreams

A Major Qualifying Project (MQP)
Submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the

Degree of Bachelor of Science
in Interactive Media and Game Development (IMGD)

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

Authors:
Ethan Chau
Aidan von Conta
Merveille Wol

Faculty Advisors:
Prof. Rose Bohrer (CS)
Prof. Farley Chery (IMGD)

May 2024

Table of Contents

Abstract

Raveling Dreams is a 2.5D isometric puzzle adventure game built in Unreal Engine 5, primarily through the Blueprints visual scripting system. Combining 2D sprites into 3D modeled space, you control the player character traveling through past core memories, reflecting on the things you wish you could have or should have at the time, and repairing your past through these memories. The player character's appearance and identity is randomized each playthrough as part of our exploration into what representation can look like in games, creating a unique experience for each player. The game emphasizes diverse representation and mental health, especially the mental health of queer and neurodivergent people, which is rarely portrayed in video games.

Acknowledgments

We would like to thank the following people for their support and contributions throughout this project:

- Our advisors, Professor Rose Bohrer and Professor Farley Chery, for their guidance, advice, and encouragement during the development of *Raveling Dreams.*
- Ellie Kim, an Independent Study member, for her crucial contributions to most of the 2D art and some tech implementation.
- Ashe Neth, an Independent Study member, for her crucial contributions to much of the tech implementation.
- Abigail Rauch, an Independent Study member, for her contributions to the art direction and concept art.
- The Fest IQP team for organizing IMGD showcase events to exhibit and playtest *Raveling Dreams.*
- All the playtesters who provided valuable feedback about *Raveling Dreams*.
- TheDyingSun for their feedback and support to the design of Fursona.

1. Introduction

Raveling Dreams is a puzzle adventure game about personal growth and diverse representation among minority groups, with a focus on mental health and development. It's a 2.5D isometric game with two-dimensional sprites styled to fit in the three-dimensional levels, with each level representing a specific core memory of the player character. These characters act as foils for the player character to work through their core memories and extract an important lesson revolving around an action they took in the past, and how that affects them now. After initial excitement and what seemed like appropriate scoping, we ran into some initial hurdles that set expectations for the remaining three terms.

We went through dozens and dozens of hurdles in this project, ranging from our small team having to divvy up roles to team members with little familiarity in their new discipline, encountering tremendous issues with source control and file management systems, dealing with deep personal struggles and fighting to stay afloat amid the ever-encroaching deadlines, and climbing over each hurdle just to face an even taller one. Through strong project management, critical interpersonal communication, and immense persistence, we were able to overcome each obstacle placed in front of us. This ranged from making important decisions about what content was or was not necessary, re-scoping to fit our needs and ability when needed, where resources should be allocated at each moment in the development cycle, and inviting new talent on to the team in areas where the core team was unfamiliar or far too overextended to comfortably fit themselves into.

2. Vision

2.1. Concept

Our original vision was to create a game that told a complex story about personal growth through dreams that would have the player relive a memory, in an effort to locate what exactly it was that they felt they needed to learn from that moment in their life. These dreams would take place in parallel to a real-world plot where the player character would have their personal growth from aforementioned dreams put to the test when a scenario involving those skills would arise. This would culminate with the dreams and real-world stories becoming connected as the player character is faced with a situation that would require all of their newly-developed skills to navigate successfully. This idea was inspired by similar walking simulator games & role-playing games like *Omori* and *Yume Nikki*, in which players take control of someone diving deep into the recesses of their mind to explore and compartmentalize, in the hopes of escaping their own head and finding their first steps on a journey to recovery and growth.

Personal growth made up the majority of the narrative, with every interaction based around an important point in the player character's life that would serve as a good moment to teach them something about themselves. This meant the story would entirely be driven through dialogue, and as such we put a heavy focus on creating succinct, effective dialogue that was short enough to take only a few clicks to navigate through while still communicating the narrative stems we wanted.

The core gameplay loop consists of three core elements. One of these is how the player navigates the environments. The players control a character that is only able to walk around each

level, but we ensured that each level had distinct enough areas and weren't too large so that navigation didn't feel like a chore. The second element of the core gameplay loop is interaction with the environment, whether direct or indirect, and interaction within the puzzle systems. We have interactable items in our levels that players must pick up or otherwise interact with in order to initiate the puzzles in each level. These include a light switch in Level 1 that is directly interacted with, scraps of paper in Level 2 that must be directly picked up, a block puzzle in Level 3, and a blackboard in Level 2 that can't be interacted with directly, but is simply observed, deciphering what it means. The final core element of our gameplay loop is interaction with NPCs. They are the vessel for narrative progression and side quests, talking to them to push the story forward, trigger subsequent levels, and initiate side objectives.

2.2. Design Goals

The pillar of our gameplay was to create a short, relaxing experience with limited gameplay mechanics to allow our game to be played by players of any skill level. The pillar of our narrative was to deliver a complex narrative about self actualization and personal development through effective and succinct dialogue to not slow players down with hundreds of dialogue boxes. The pillar of our design was to create a player randomization system that allows for anyone to be represented and for players to not have the guarantee that they will play as someone who looks like them, and as such experience a story about mental health through a lens that they may not be familiar with. The pillar of our art was to create and implement a character randomizer that is able to represent a broad enough group of people and encourage players to view mental health through a unique lens that may not perfectly overlap with their experiences with mental health, well-being, and personal growth.

Personal development is an incredibly important theme that we wanted to ensure every player would understand, so we integrated it into every aspect of our game to various degrees to ensure it felt like the strongest overarching theme. The themes in our game design and gameplay were very similar, giving the player the same framework for their puzzle system in each level, and it is intended to feel like the player is taking skills they learned in the first level and building on those to complete the puzzles in each subsequent level.

We were very easily able to implement a narrative of personal growth through strong dialogue that builds upon itself in a similar way, where the first level comfortably introduces a moment where they are incentivized to ask for help in an uncomfortable environment. In the second level, this is built upon by presenting them with another uncomfortable situation and encouraging them to use the same problem solving mindset to rationally work through the case of the missing sketchbook, and deliver a calm verdict without resorting to anger or violence. In our final level, the player is presented with a far more difficult situation, needing to make the decision between difficult work and taking a fresh step out of a toxic work environment and allowing themselves to start somewhere else without such toxicity.

Our art was intended to show the same personal development in a different way, instead serving to aid in the display of each distinct stage of life. This worked hand in hand to show the player moving closer to the present in their dreams and, as a result, developing in other ways. It helps demonstrate to the player that the main character has grown physically and mentally in general, and they are at the point in their life where they are able to make these complex connections between core memories and personal development. Level 1 resembles their childhood home, with sprites resembling crayon and construction paper drawings. Level 2 looks like a school hallway, with characters that look like they were taken right out of the margins of a

school notebook. Level 3 resembles an apartment and office complex, with sprites that look like semi-professional art assets, meant to indicate that our character is at the point in their life where they are the one making decisions for themself and to suggest they are ready for the weight of the process of self actualization.

An additional art note was how we wanted to handle representation. Personal growth and mental health are not exclusive to a certain race or gender identity, and we wanted to represent that as such. We accomplished this by creating a handful of randomized body types, skin tones, and hair types, randomizing the player character independently, and giving the player their character to play with. The goal was never to allow the player to customize their character, but instead to offer them a different perspective on an experience they may have gone through.

Queer and neurodivergent people do not get fair or appropriate representation in any form of media, generally having their gender identity, sexuality, or neurodivergence being the core of their identity. These kinds of people need proper representation where they are layered and complex characters with complex identities that are not centered around their gender, sexuality, or diagnoses. We intend to offer players the freedom and space to fill parts of their character in with their own identity by offering somewhat blank randomized templates, dropping pieces of dialogue that give players room to identify with our character, and associate their own identity with the open space in the character. We also worked with dozens of references for Afrocentric hairstyles to give appropriate and accurate representation to our Black players, as Black hairstyles have been a frustrating aspect of games for years (Beyond!, 2024; Dowd, 2024). Developers seem to latch on to three Black hairstyles and never explore further than that, so we took it upon ourselves to create several that were interesting, unique, and provide Black players with plenty of options.

2.3. Target Audience

Our target audience consists of queer and neurodivergent people, specifically because they aren't properly represented in many forms of media, especially not their health or well being. They have so often been portrayed as two-dimensional caricatures with their queer identity or neurodivergence being their defining, or only, character trait, and we instead wanted to provide players with the opportunity to fill our character in with their own experiences to create a character that accurately represents them. Our character randomization also wanted to encourage people to practice empathy and experience the world from another pair of eyes with an identity that may not wholly match theirs. We wanted to appeal to queer players who don't often see a multi-faceted, complex character with their queerness being only one aspect of their layered identity. We wanted to appeal to neurodivergent players who often struggle in social settings and hyper-analyze every single social failing they've ever had. We wanted to capture those feelings of regret and turn it into something a player could recognize as a moment for learning, instead of a moment to relentlessly criticize.

We tried to appeal to these kinds of users by creating systems that would accommodate as many different kinds of identities, both physical and mental, as we could in the time constraints. We worked with broad strokes, creating body types and skin tones that were limited on their own but could be filled in by the player to breathe more life into them. We created 4 body types that did not conform to realistic body shapes, but instead were more characteristic of a caricature, utilizing odd shapes to represent body types instead of filling in all the detail ourselves. By using these odd shapes, we are leaving players to see a shape that somewhat resembles them, and fill in the smaller details that would fully complete the picture that represents them.

3. Project Management

3.1. Work Structure

In order to manage a group that struggles immensely with time management and staying on top of assigned work, we had daily group meetings to ensure at least some work would be completed when we were meeting. The goals of these meetings changed with each stage of development, but their underlying purpose would remain the same.

At the beginning of the year, these meetings were all about pitching ideas. Anything from narrative ideas to art directions to system designing were all on the table. We used this time to take thorough notes about which ideas the team liked, which ideas we found feasible, and which ideas were good as stretch goals should we be left with time at the end of production.

Deep in the production cycle, our meetings turned into times where the team would hold each other responsible for getting things done. These times could be spent being productive on any number of things, ranging from the direction of the project to collaboratively focusing on one element of the project that was proving troublesome. There was always room to be flexible and work on whatever was pressing or possible in the moment, and we would frequently have discussions about where we were and what we were actively working on, as well as next steps. The project manager would take charge and ensure the entire group was being as productive as they could in our daily two-hour blocks and that we would have some tangible progress to show at our weekly meetings with our project advisors. These weekly meetings would serve as time to keep our advisors updated on the state of our project and get feedback and advice about the direction of our game.

These weekly meetings would be accompanied by a bit of goal setting and breaking down tasks as we took our advisor feedback and turned it into a plan for the coming week. They acted as weekly goalposts for us to reorganize from the previous week and get the ball rolling in preparation for the next meeting. In addition to weekly goal discussion, there were also long term goal meetings at the beginning of each term, where tasks for each department of our project, those being design, art, writing, and tech implementation, would be discussed and broken down into manageable sizes for us. We experimented with two-week sprints for B-term and a part of C-term with Jira, but its effectiveness was limited and frequent check-ins needed by some team members directly conflicted with the self-directed nature of these sprints. Our team contained members who had been struggling with burnout and a host of mental health struggles before the project even started, so getting momentum on things was incredibly difficult when those team members could unexpectedly have difficult days where very little would get done. Accountability was lacking at times, which led to tasks in these sprints being pushed off to the next sprint, and so on until it would eventually be completed after eating into however many sprints it took over. This meant it was very easy for team members to fall behind and fail to communicate their status to their team members, which would cause a great deal of confusion and frustration when time came for a weekly checkpoint and someone would arrive having done very little and not having communicated how much they were struggling.

Adding to Jira's limited usefulness was how ineffective kickoffs and retrospectives were. These were small meetings at the beginning and end of each sprint where sprint goals and productivity were discussed. Team members had difficulty appropriately planning for how much work should be expected for each sprint, and the project manager did not have enough expertise in each of these fields to make an appropriate judgment either, and as a result kickoffs were

usually just short planning meetings that could be handled without Jira or a sprint system. Retrospectives at the end of each sprint were not helpful because of how often goals were not met and effort went in different directions, and there was never enough energy after each sprint to want to dive into why sprint goals weren't met and pinpointing weak areas that needed to be addressed.

Jira added on extra weight to an already mentally taxed group that was far more of a burden than a benefit, as the project manager observed that it was difficult enough for the group to stay on top of their tasks let alone update them in Jira, remember what was to be completed for each weekly checkpoint, and stay organized within the shared google drive. A simpler way to divide tasks was to operate with a Trello board, where team members were in charge of their own boards for task organization and the project manager was in charge of an overview board that broadly tracked progress from each discipline. One area which was updated regularly and kept on top of was bug tracking, also making use of a Trello board. This was the lowest-bandwidth way to keep track of bugs, as the programmer already had familiarity with it on top of it being a very approachable piece of software with easily scalable systems that meant bugs could very quickly be added, marked as resolved, and filed away after fixing.

3.2. Asset Management

We organized everything within a Google Drive folder shared with every team member, as seen in Figure 1. This was where we kept our weekly meeting notes, design documents, art assets, dialogue scripts, and other documentation of our development process. Everything was organized into sub-folders named after their purpose or discipline, dividing our assets and notes

into appropriate categories. These folders were named for their contents and split into sub-folders to organize further, like splitting our art folder into 3D assets, 2D sprites, and UI elements.

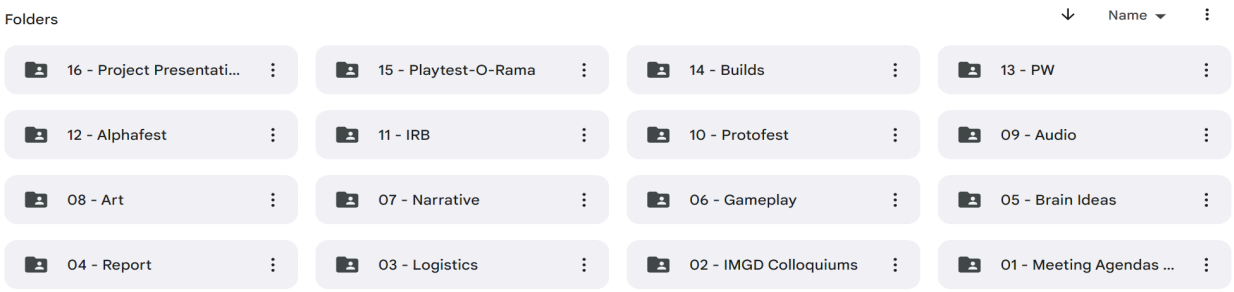


Figure 1. The *Raveling Dreams* Google Drive folder with sub-folders.

Our art assets were kept in a folder specifically for art, which was further divided into 2D sprites, 3D levels, 3D assets, menu assets. etc.. Files followed a naming convention to make organization easier, in this case denominating certain files as background assets, background sounds, character sprites, or assets for the puzzle menu. This was accompanied by spreadsheets to keep track of what assets were created, where they belonged, and what their implementation status was. This was important to keep up to date, as it made planning and dividing work much easier when knowing exactly what remained and what the status of each asset was.

We had folders for our meetings, plans, notes for weekly IMGD meetings, brainstorms, and every playtesting session we would get. These kinds of things were equally important to keep track of for a number of reasons. We would plan out each meeting with our project advisors and record as much of their feedback and suggestions as we could. We would take down important notes from our own meetings whenever possible, catch every bit of information our playtesters would give, and store it all so it would be easily accessible in the future, when it was ready to be processed. It’s the place where we could put in-progress documents, unfinished reports, spreadsheets with no assets completed, etc., and it would be very easy to revisit them.

We also stayed organized inside of Unreal Engine, creating folders for assets for our old and new builds and keeping each Blueprint in its relevant folder so anyone could quickly and easily navigate the project.

Blueprints specifically were stored in separate and multi-layered folders so it was easy to tell which Blueprints related to what systems. Our player folder, for example, had multiple subfolders for each of its systems, with an individual folder for the player randomization system, player movement, and player interaction with other assets. Each character in the game also has their own content folder, which contains all their sprites, data, blueprints, and any other assets. This made it easy to duplicate and add NPCs and edit each individual NPC dialogue, as they were all already separated and simple to replicate.The UI of the game was contained in its own folder so that it may be modified with ease, as was essentially every backend system of our game. We encountered dozens of bugs as we developed each system, and having them split made it very easy to diagnose where issues arose and address them.

Our art assets that stood alone were grouped into a large folder where they could easily be accessed, split into what assets belonged in each level. These assets were the 3D models and levels themselves, put into three different folders corresponding with the level they are featured in. This same organizational system was implemented for audio, where each sound effect and MetaSound were put into folders depending on their purpose and associated level.

3.3. Game Engine and Source Control

Unreal Engine was chosen as the workspace for *Raveling Dreams* because collectively the team members had the most experience with it over engines like Unity or Godot, and it seemed the best choice for our 2.5D game with its powerful rendering, accessible Blueprint and

MetaSound system, and easy 3D development tools. It also gave us access to dozens of powerful plugins that would aid in our development cycle, like Paper2D and the NotYet dialogue system.

Originally we used GitHub Desktop as our source control, with one main branch and a branch for each team member to work on their tasks in-engine. However, we ran into issues with merging our content with files becoming clobbered, deleted, or duplicated. There were several instances where our unfamiliarity with GitHub Desktop would lead to enormous merge conflicts that would unintentionally destroy entire levels and duplicate hundreds of files, quickly bloating the project file and deleting important pieces of our game. We were using Github's Large File Storage (LFS) system, which was something none of us had a large amount of knowledge of, so that led to further complications when trying to move and merge a quickly growing project file.

We decided to switch over to Perforce, this time with one main branch for our final changes to be implemented, and a singular development branch for all team members to work within. Perforce was chosen because of its "check-in, check-out" system that prevents other team members from opening and editing files that one team member is currently working on, which would save us from any merging headaches in the future. Perforce has integration with Unreal Engine, making it quick and easy to check-out files and identify which files you, or other team members, have checked out for work, and are not updated to the latest revision. With this integration, Perforce will automatically check-out files that you try to save edits to. This meant a lot of the issues we ran into with GitHub Desktop, like save conflicts, merging issues, and multiple people working on the same file at the same time, were all solved automatically by Perforce. There was still a learning curve while getting accustomed to the software, but it was a far shallower curve than learning to effectively use GitHub Desktop.

We still ran into our fair share of problems though, especially with some users forgetting to check their files back in for others to use, or dealing with problems unique to each member's perforce client, but these problems were far easier to solve and led to many fewer headaches than with GitHub.

4. Research

4.1. Influences

Figure 2. A screenshot from *Super Paper Mario*, one of the influences on *Raveling Dreams* (Walker, 2007).

Our game was inspired by puzzle adventure games that have succeeded in implementing a unique and stylized 2.5D isometric art style, like *Super Paper Mario* and *The Wild at Heart*. These are two games that operate with an isometric perspective, *Super Paper Mario* combining 2D assets with a heavily stylized 3D background, and *The Wild at Heart* placing 2D sprites on a 2D background with special camera work to give the illusion that these characters are moving around a 3D space. *Super Paper Mario* operates on two planes, those being 2D and 3D, where the titular character, Mario, is always facing the camera no matter which perspective the player is playing in. While this was a very interesting approach, we immediately were able to recognize it would have been far too much work for us to handle. We took some notes from *Super Paper Mario*, specifically how Mario is always facing the camera, as we moved forward with planning our systems.

Figure 3. A screenshot from *The Wild at Heart*, another influence on *Raveling Dreams* (Wickens, 2021).

This brings us to our second largest inspiration for art direction and functionality, *The Wild at Heart*. This game uses entirely 2D assets, and instead toys with the layout of each asset to give the illusion of three-dimensional space. We didn't take the same approach, considering we had no dedicated artist on our team, and we kept the 3D assets in *Super Paper Mario* in mind as we explored. We drew more inspiration for the art direction from *The Wild at Heart* as we noted how strong the art direction was, and how far it carried a relatively simple concept of 2D sprites walking on an angled 2D plane. We took inspiration from this to lean very hard into our simple art style and extract as much out of it as we could. Having no dedicated artist limited our direction at first, but once we were able to onboard a temporary artist, they were able to push our art further and we were able to lean very hard into our art style and take what would be very simple and empty levels and inject some variety to break them up with our character sprites.

For our core gameplay, we took inspiration from two role-playing/walking games, *Omori* and *Yume Nikki*. These are both games that discuss mental health in complex and interesting ways, and they were relatively simple bases that were expanded into deep and expansive games. The core of both games is simple cardinal movement with the W, A, S, and D keys and exploration of fantastical two-dimensional spaces meant to simulate the bizarre nature of dreams and the human mind.

5. Narrative

5.1. Themes

The narrative of *Raveling Dreams* is one about self-reflection and personal growth through the exploration of a set of core memories and experiences, aiming to teach the player about empathy for others and towards oneself. The player navigates three distinct levels that dive into connected themes about one's mental health and well-being, those being seeking aid in times of crisis, healthy conflict resolution and communication, and putting oneself first.

In the first level of the game, a core memory taking place in the player's childhood bedroom, they are placed in a dark and ominous room with frightening creatures hiding in the corners and under the bed. Intended to evoke fear, the player character's natural reaction is to be scared and to want comfort and reassurance from a safe figure in their life. The player character thinks back to this point in their life, where they elected to hide under the covers and wait out the night and comes to the decision that they should go find their mother for comfort. They make a comment expressing excitement at the prospect of seeing their late mother again, and are eager to make a better decision by asking for help in a difficult moment. This leads them down an equally unsettling hallway to the bright and comforting light of their mothers room, where she comforts the player character and acts as a safe space free of monsters and scary imagery.

The second level, a moment from middle school, takes the player to a hallway populated by their peers and their fursona from middle school. The issue that presents itself this time is that the player character's sketchbook has been stolen from their locker, and they must set out to find the culprit. Through various paper scraps scattered around the level, a mysterious cipher given to

them by their fursona, and dialogue from the four classmates placed around the level, the player can come to the conclusion that their sketchbook was taken by their goth classmate. The player has a conversation with Goth about what happened, and they make a conscious choice to resolve the issue in a better way than they did. They mention that when they were in middle school and this happened, they ended up getting into a fight with Goth over their sketchbook, and they understand that there are healthier ways to resolve the issue. Goth took the book because they liked looking at the drawings that the player character made, and the player can acknowledge this by asking for better communication next time. They tell Goth that all they need to do is ask to see it and they would have done it, and though it is just a dream, the player makes the conscious choice to communicate clearly and solve the problem instead of resorting to arguments and physical violence.

In the final level, the player finds themself in their first apartment and the office of their first job. They now have to deal with a difficult landlord and a frustrating boss, and they have to navigate both of these complex issues while being evicted and laid off simultaneously. The player must leave their packed-up apartment, still not completely done moving in1, head to their office, and navigate the maze of cubicles to reach their Boss. They are reprimanded for being a single minute late and given a hefty and bizarre task to complete, where they must deliver office supplies to the various empty cubicles in the maze. Once this is complete, they return to their Boss, who promptly delivers a corporate spiel about how they don't fit into the company environment, they aren't a good fit, and more bumbling excuses before firing them. They then must head home, knowing full well that once they arrive home, their dream will direct them to their Landlord, who will evict them as soon as they enter the door. They navigate through the blunt and hostile dialogue of their Landlord, and are left with nothing but their belongings. The

player has a moment to sit and take everything in, before taking a deep breath and figuring it's not worth it to get angry at their Landlord a second time. They take what they can in stride and simply prepare their stuff to be moved out, and look forward with as much enthusiasm as they can reasonably muster.

6. Setting

6.1. Dream World

The game takes place in three different dreams, manifested as small fragments of spaces floating in a void. Each space is only supposed to resemble the real world, but not match it one-to-one. Each of the three levels is set to resemble the environment at the specific points in these three dreams, with the first level looking similar to the player character's childhood house, the second level appearing similar to the player character's middle school, and the final level resembling the player character's first apartment and job office. These levels are intended to appear as the player character remembers them, so a simply dark room appears like a room full of monsters, a hallway full of peers turns into a confined space where they are the only ones the player can interact with, and a confusing office space turns into a full blown maze.

The first level that looks like the player character's childhood house consists of three rooms, one being their bedroom, one being a long and dark hallway, and the final room being Mom's room. These rooms are designed to look similar, as they are two bedrooms of two connected people, but the lighting turns one of them into a hostile environment while making the other a welcoming and homey environment. The player character's mind has warped their normal bedroom into a scary place that is full of monsters, as seen in Figure 4, but the reality is that it is a harmless dark room.

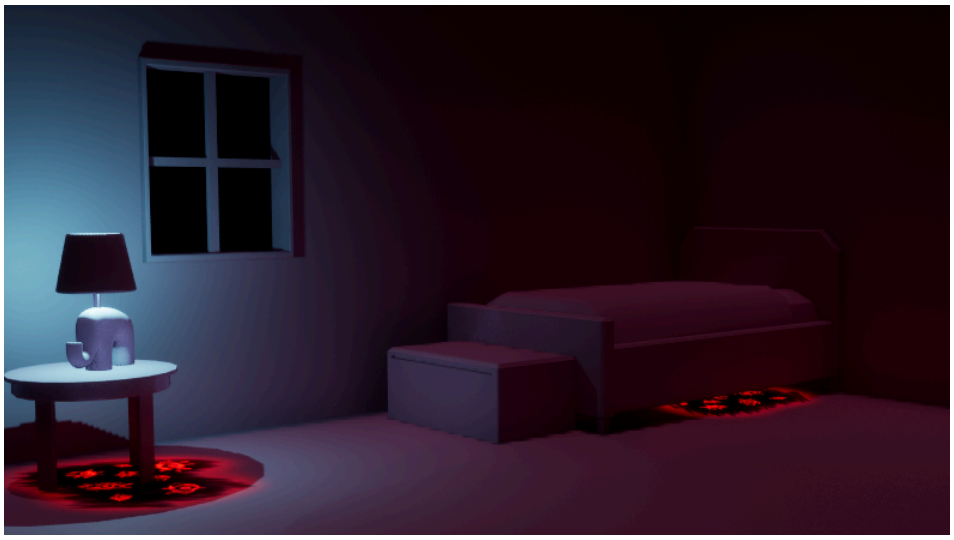


Figure 4. The player character's bedroom with the lights off, revealing the Shadows.

The second level takes place in a condensed version of the player's middle school, Shi-Dee Middle School, altered by their mind to fit into two short hallways and one small room as the homeroom. While the real-world middle school would be a far larger building, the dream world has turned it into a very small and confined space where the player has no choice but to interact with their peers. The player character was not very well known by their peers, nor did they know their peers, and as such, the levels reflect how they have relegated their four classmates to their own corners of the map, as if they own their own regions of the school. Their homeroom, seen in Figure 6, is the only place where they are alone and felt most safe, as they were always able to hide in the back of the room.

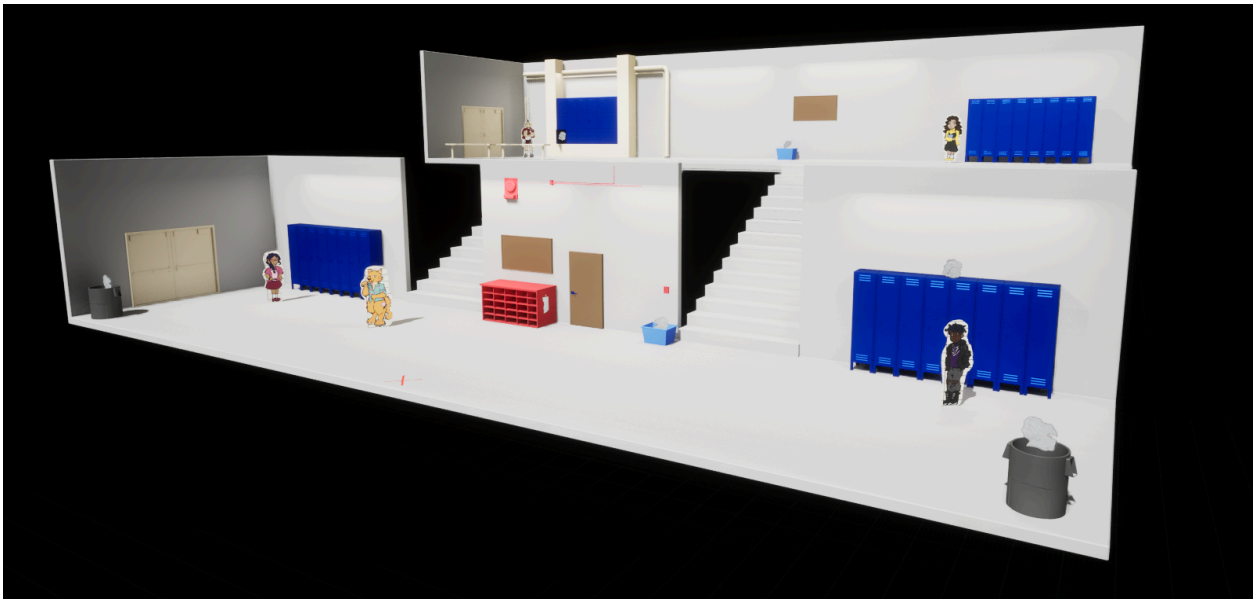


Figure 5. Overview of Shi-Dee Middle School's hallways in Level 2.

Figure 6. The player character's homeroom in Level 2.

The final level takes place in an office and the player character's first apartment, seen in Figures 7 and 8, where they navigate through the office building and the cubicle maze to report to their Boss's office. Their apartment acts as a similar space to the homeroom in Level 2, where they feel the safest there, but the mess in the corner conveys the disorganized state the player character is in, knowing that once they get through their work for the day, their memory will bring them back to an eviction notice.

Figure 7. The Boss's office in Level 3.

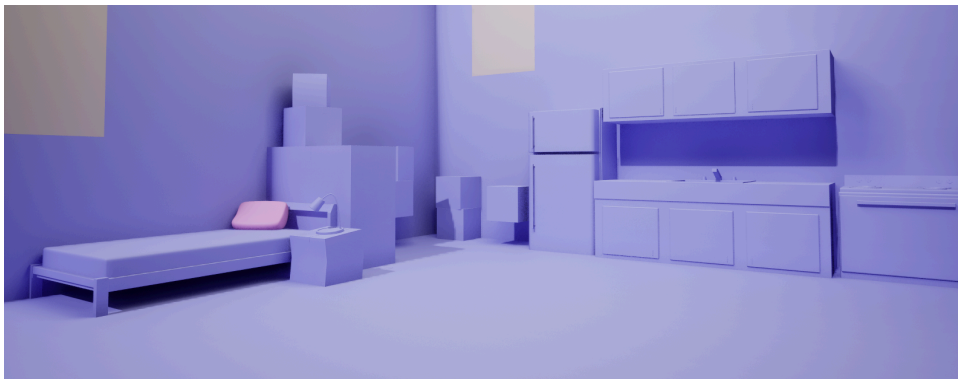


Figure 8. The player character's first apartment in Level 3.

6.2. Characters

The characters in *Raveling Dreams* are designed to evoke memories from each era of life and feel as if they have been ripped straight from that era of the player character's life. Their designs are intended to fit right in with the current level and clue the player in to the point in the player character's life at which they were met, and how the player character views them.

In Level 1, the only NPC that the player can interact with is a drawing of their late mother, who is placed into the level to act as a source of comfort for the character as they reminisce about the time they spent with their Mom. In Level 2, there are five NPCs that the player can interact with. These are the four archetypal characters that the player character had the most interaction with when they were in middle school, and they are designed to lean heavily into those archetypes, as the player character did not know them very well, and as a result, does not think of them as very complex characters. In Level 3, the only two NPCs that can be interacted with are the player character's landlord and boss. They are monocolor and faceless in their depiction to convey the distance and hostility that the player character felt towards them.

7. Design

7.1. Rooms and Levels

The first level of *Raveling Dreams* consists of three rooms, the player's room, a hallway, and their mother's room. The player starts in their room, eager to escape from the frightening Shadows scattered about their room, but they find the door leading out is locked. They must complete a block puzzle and spell the word "OPEN" with their four letter blocks in order to unlock the door, at which point they will transition into the hallway. This is the most straightforward section of the level, where the player must walk down the length of the hallway, occasionally getting some lines of dialogue to emphasize the loneliness and fear the hallway evokes. After making it to the end, they enter their mother's room, where the lights are on and she is ready to speak to the player. She acts as the goal of the level, the safe space as opposed to the eerie atmosphere present in the player character's room.

The second level, taking place in a middle school, consists of only two rooms, those being the main hallway with all five NPCs the player can interact with and the paper scraps for the level's puzzle, and a homeroom that has a blackboard puzzle and no NPCs. The original level layout had four additional separated rooms, one for each NPC, and the player would teleport between each of them. When the first iteration of Level 2 was played through at Protofest, an IMGD event in which very basic iterations of games are tested by members of the WPI community, this was not a very big issue. Players found it somewhat difficult to remember exactly where everything was, but the transitions weren't jarring and didn't hinder player movement. However, when implemented digitally, the transitions between levels were very

jarring and the layout of the space was even more difficult to remember. This led to the current iteration, where the school hallway is one large space, with the only teleporter being between the hallway and the homeroom. This teleport was far less disorienting because it is triggered by interacting with a door, as opposed to the old system of red markers on the floor. This gave players a stronger sense of connection between the two spaces.

The school hallway is split similarly to the Alphafest build, but instead of being four separate areas, they are all connected in the same level. The four NPCs that each occupied a separate room are placed in each corner of the large hallway room, those NPCs being Jock, Nerd, Prep, and Goth. Jock and Nerd are on the upper floor hallway, while Prep and Goth are on the lower floor. The player character's fursona stays in the middle of the lower hallway, and they act as the initial contact point when the player transitions from Level 1 to Level 2. The original plan was to have Fursona move from the center of the hallway to some empty space near Nerd's wing, but ultimately we decided it would be better to leave Fursona in the middle of the level to reinforce the idea that the player character was not particularly close to any of their peers and instead is very close to a character of their own creation.

The final level has four spaces, one of them being the player character's studio apartment and the other three being spaces in the office at the player character's job. The player starts in their apartment, partially unpacked and disorganized, and exits through their apartment door to find themselves transported in an instant to their office break room. The move through to a large cubicle maze, a section of which is seen in Figure 9, that has been designed to feel hostile and easy to wind up lost in. The player must navigate this maze multiple times, and as a result, they must become familiar with its layout if they want to avoid struggling. Once the maze has been conquered for the first time, the player enters their Boss's office, where they are meant to feel

small and unimportant in the dimly lit room. They will be scolded by their Boss, sent on a task in the maze, and brought back, only for their position at the company to be terminated.

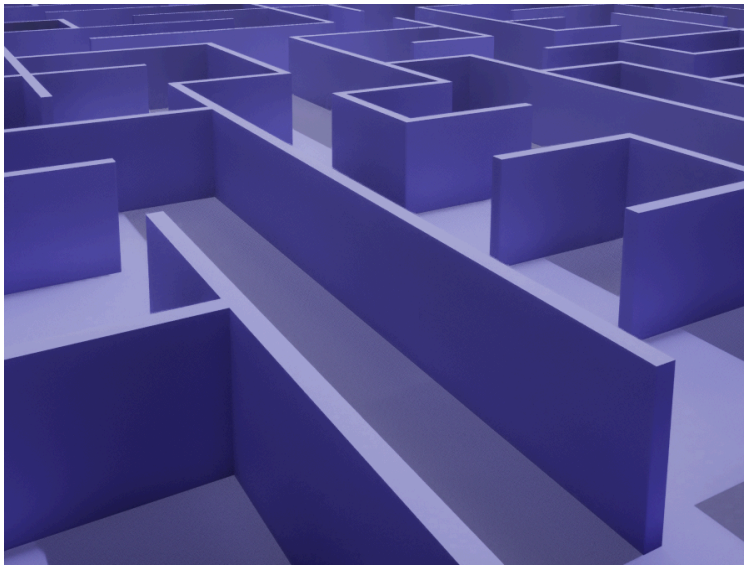


Figure 9. A section of the “cubicle maze” in Level 3.

7.2 Inventory

The inventory system was initially created for Level 2, as it is the only level where the puzzle items are not given to the player all at once. In Levels 1 and 3, the player either finds all the necessary materials to solve the puzzle in one collectible, or they are given all the materials they need. In Level 2, however, they must collect various paper scraps scattered about the school hallway. The inventory system keeps track of which paper scraps the player has collected with a menu in the corner, which has a different icon contextual to the level the player is in, and it will show the player if they have collected the “Supply Closet Key” and “Dino Grabber” key items.

7.3. Puzzles

Each level in *Raveling Dreams* has a unique puzzle that builds on the puzzle system from previous levels in an interesting way. They are all variations of drag-and-drop puzzles, where the correct pieces must be dropped in the correct relative positions to solve the puzzle, but in each subsequent level, they get a bit more difficult.

In Level 1, the door out of the player character's room is locked. They can find four letter blocks, "O", "P", "E", and "N", stacked on their wardrobe. These introduce both the puzzle menu and the drag-and-drop puzzle, as seen in Figure 10. The solution to this puzzle is to arrange the letter blocks so that they spell the word "OPEN", after which the door will open.

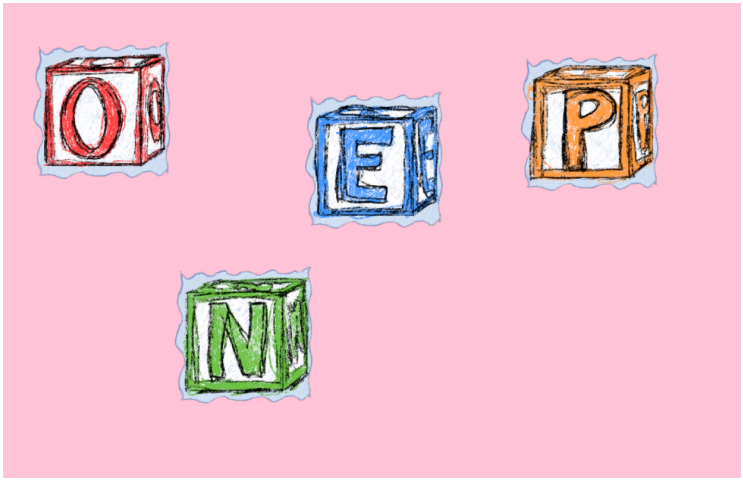


Figure 10. The 'letter blocks' puzzle screen for Level 1.

In Level 2, the puzzle gets an added layer of complexity with the addition of differently shaped pieces that must be fitted together correctly. In this level, the player's sketchbook has gone missing from their locker, and they must figure out who took it. They must pick up scraps of paper scattered around the level that, when fit together, make a number combination. The player must be able to spot which scraps of paper fit together, and which ones are duds and do not have a match. Once they've collected them all and deciphered the number the paper scraps

hide, they will end up with a locker number and locker combination that leads them to their sketchbook.

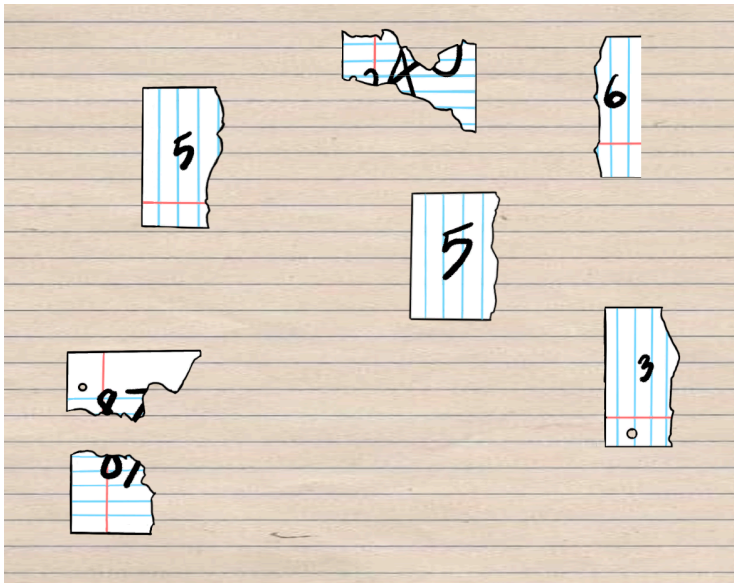


Figure 11. The ‘paper scraps’ puzzle screen for Level 2.

In Level 3, the intended puzzle now uses *Tetris*-like shape blocks and the player must slot them into a grid in such a way that all the blocks are able to fit. The player is instructed to fit office supplies into a little handcart, so the goal of trying to fit all the blocks into the grid evolves into delivering as many office supplies at once as possible. There would be a second part of this puzzle where the player would head back to their apartment and be faced with eviction, which would require them to organize all of their belongings in a similar grid puzzle to the office supplies puzzle. These puzzles would be more complex than the previous two in that there would be more than one solution, and it would prompt players to experiment with multiple solutions instead of getting frustrated trying to make one solution work. The initial puzzle would only feature L-blocks, I-blocks, and a square block, to allow the player to ease into this new puzzle system. The second puzzle would involve more complex shapes, like a U-block or a T-block, and

these would provide the player with a greater challenge for the final puzzle of the game. As with the first puzzle, this second, more complex, puzzle would have multiple solutions.

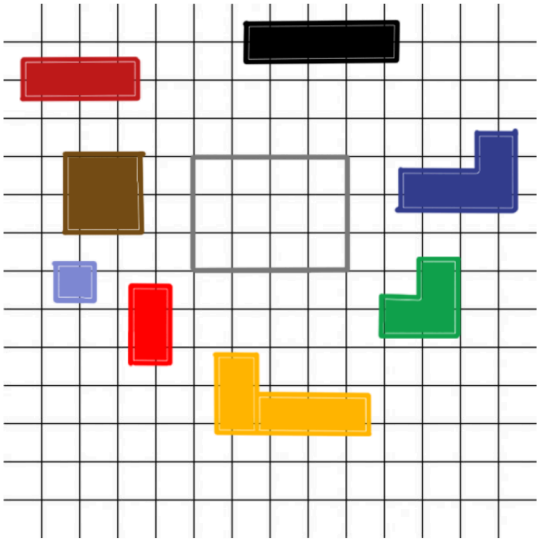


Figure 12. Mockup of the puzzle screen featured in Level 3.

7.4. Dialogue System

Raveling Dreams makes use of the NotYet Dialogue System plugin for Unreal Engine 5 to streamline dialogue implementation and functionality. The NotYet dialogue system utilizes proprietary Dlg files to organize and display dialogue nodes as one-off dialogue lines, conditional dialogue paths, or randomized sets of dialogue. They are then seamlessly attached to our NPC Blueprints so that these systems can be activated when NPCs are interacted with. The participants in dialogue are defined in the Dlg file as the player and the NPC that the Dlg file is attached to.

By default, the NotYet Dialogue System comes with boolean checks for if the dialogue node has been visited yet. This is useful for playing a dialogue sequence only once upon first interacting with the NPC. We use this in all implemented levels, notably with the Light Switch in

Level 1, the Nerd in Level 2, and the Boss in Level 3. The system also supports custom conditional dialogue paths. Dialogue nodes can perform a check against a boolean in a participant Blueprint, either the player character or the NPC in the current dialogue sequence. This is useful for playing dialogue sequences that only appear after the player starts a quest and before they finish it. End nodes in dialogue can also execute an event, which is useful for updating booleans in a participant Blueprint and ensuring that the correct dialogue progression will be played for the current level or quest. Boolean checks and updates for quests-in-progress and finished quests are most notably used in Levels 2 and 3 to create cohesive narrative progression.

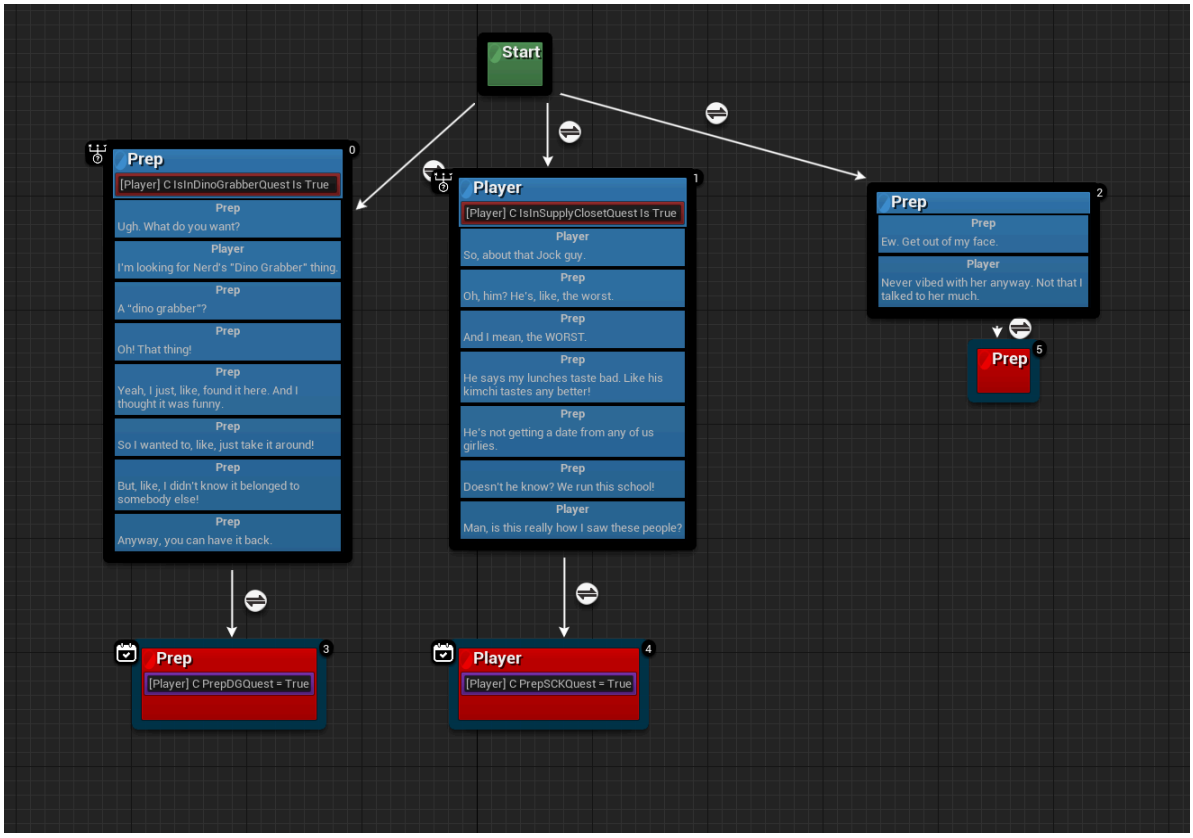


Figure 13. Screenshot of the Dlg file for Prep, as implemented in *Raveling Dreams.*

8. Art

8.1. Art Direction

There were two primary drivers that pushed us towards our simple art direction of character with stick limbs and simple shapes as bodies. The initial idea for an art style was to have the characters appear as if they came straight from a dream journal, where the player character would have drawn them as if they had the same artistic ability as anyone from our core team. The other driver fits very neatly with the first driver, that being none of the core team were dedicated artists. We had a writer and manager, a programmer, and a sound designer. None of us had the experience or time commitment to create art assets for the game, and the members on ISP we were able to enlist for help didn't have the same time to commit to our project, so scalable and simple characters were very beneficial to their output.

This was also one of the reasons we went with 3D environments and 2D sprites. 2D backgrounds would require an extensive time commitment from artists that we either didn't have or couldn't commit that kind of time to our project, and 3D characters would have equally sucked time away from other disciplines, as none of us were familiar with rigging or 3D character design.

The isometric style of game was what we eventually settled on, for both stylistic and limitation reasons, and this is a style of game where the player gets a ~45 degree view from above and to the side of a space, and they control the character as if they are looking down on them and the world. *Hades* is an example of an isometric game that does it to great effect. In *Raveling Dreams*, we created something more akin to *Super Paper Mario*, where 2D sprites

move around a 3D environment. Unlike *Super Paper Mario* however, our camera is static and just moves with the player. The only time it changes is when it hits certain camera triggers, where it moves to a different fixed location.

Figure 14. A screenshot from *Hades* with its isometric camera view (Supergiant Games).

Every level was modeled with this isometric view in mind, taking advantage of how the camera would operate and saving on modeling time by not modeling certain parts of objects or making other parts of them low detail, as the player would never see them. For example, the backside of a bed in Level 1 can have no detail or legs because the player would never see them under normal circumstances.

The 2D sprites that would be navigating these 3D environments were drawn to appear as if they were taken straight out of a dream journal. They are drawn to contextually fit into each level as well, with each character looking like they were taken from different stages in the player character's life.

In Level 1, the childhood bedroom, the only interactable NPC is the player character's mother, and she appears to be drawn with crayons on construction paper. Her design is very simple, with her defining characteristic being the large heart drawn on her chest. However, the level also features the non-interactable Shadows, which are abstract manifestations of the player character's fear of the dark as a child.

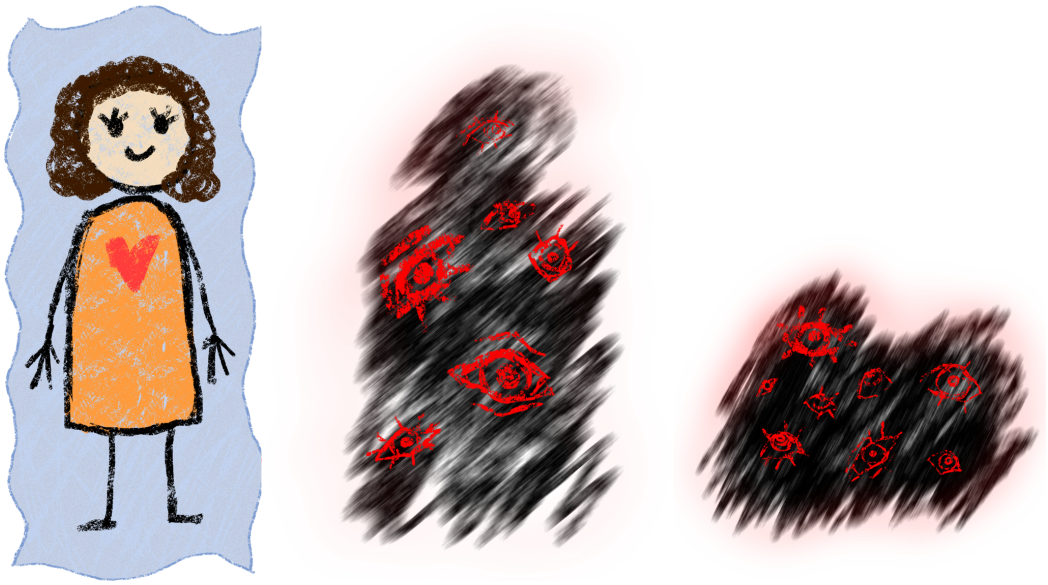


Figure 15. The Level 1 NPCs: Mom and the Shadows.

In Level 2, the middle school, there are five NPCs who are drawn to look like they were taken out of the margins of a school notebook. This style is intended to resemble the anime-style art that comes out of middle schoolers. It is supposed to show the shaky lines and mildly poor anatomy of those drawings. One of our primary inspirations for these assets was the art style of the animation studio CLAMP, namely their animation work done for the anime *Code Geass*.

Figure 16. The Level 2 NPCs: Jock, Nerd, Prep, Goth, and Fursona.

The style for Level 3, the office and apartment, drew inspiration from Corporate Memphis, which we dubbed “soulless”, “corporate”, and several other synonyms that denote it as terrible. We wanted the player to not feel entirely comfortable around the two NPCs in Level 3, as if the people you interact with aren’t entirely human. We created a landlord and a boss who both felt very distant and cold, with the player stating such in the dialogue with them. These two characters were designed to look like they were taken from semi-professional art sheets, as if whoever drew them had been working as a freelancer or been drawing professionally.

Figure 17. The Level 3 NPCs: Boss and Landlord.

8.2. Character Designs

8.2.1. Player Character

Our original character design was intended to be a very androgynous-looking character who was not easily identifiable as any particular gender identity to allow for anyone to associate with that regardless of how they present themselves. Once this goal was met however, we figured we could go further with it and create a system for randomizing the player character. We took some notes from how *Rust* handles player randomization, specifically how it avoids character customization by giving the player a random skin tone and forcing them to play with it (Lopez, 2016). The primary reason we chose to pursue player randomization was that we wanted people to experience this game about self-actualization through a lens they may not have familiarity with, like seeing the journey from the eyes of someone else who doesn't share the same outward identity.

Figure 18. The very first sketch of the player character, introduced in pre-production.

We made careful decisions about the scope of our character randomization system, as implementation proved more difficult than anticipated. We ended up with 16 different randomization combinations of body type and hairstyle, taking advantage of four body types and four hairstyles. We went with Afrocentric hairstyles that were not commonly seen in games, like cornrows and afros, and we took steps to ensure these hairstyles were appropriately designed and properly represented their real-life counterparts. We created more hairstyles testing and eventual implementation, but time constraints limited how many we could implement successfully. These hairstyles included fades, afros, locs, undercuts, as well as a few hairstyles that represented different hair types, those being wavy, curly, straight, and coily. This system allowed for broad representation that many people would be able to slot themselves into by filling in some of the details themselves. Our audio lead can connect through hair with a character that has wavy hair, despite the difference in length between their hair and the hair of our character. These kinds of broad strokes are what we utilized to ensure our rudimentary character randomizer would still be able to give players a degree of connection to the character they play as.

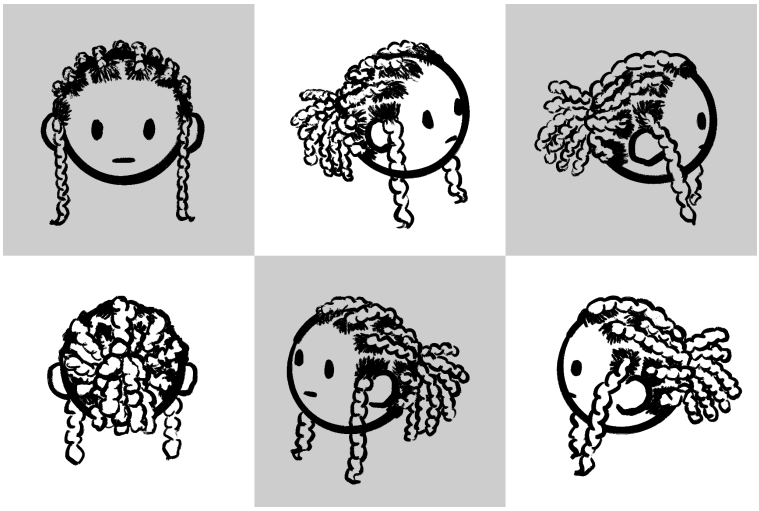


Figure 19. Concept turnaround for one potential hairstyle of the randomized player character.

Figure 20. Examples of our randomized player character as seen in-game.

8.2.2. Level 1 NPCs

Mom

Figure 21. Mom's sprite, as seen in-game.

The interactable NPC in Level 1 is the player character's mother. She does not share many physical features, if at all, with the player character, no matter the randomization. This was an intentional design choice to leave the player to interpret what the family history and relationship between the player and their mother is. There are plenty of explanations for this, including the possibility that the player was adopted, that they are mixed race and the parent not featured in the game is more closely visually associated with the randomized player character. This is intentionally left ambiguous to give the player room to imagine what's going on behind the scenes and fill in their own story for their character. It also reflects the real world, where certain features from parent to child may not accurately represent their relationship. Assuming the skin tone and hair style between mother and child would be exactly the same is not always the case in the real world, and we wanted to reflect just that.

Her design is meant to reflect the player character's perception of their mother at that stage in their life, that being when they were a child. She is drawn with brightly colored crayons and given a bright smile and large heart on her torso. This is indicative of the safety that the player felt with their mother when they were a child, and the safety and reliability they long for in their adult life, as it is revealed that their mother passed away at some point in between this level and their current point in life. Her position in the brightly lit and safe room at the end of Level 1 acts as a safe goal post for the player to reach after escaping the dark and scary room prior, where the player receives comfort and love from a figure they've always felt safe around and loved by.

Shadows

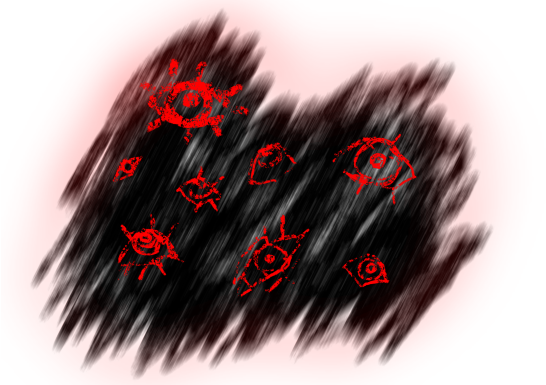


Figure 22. A Shadow sprite, as seen in-game.

Level 1 also features static antagonistic entities simply referred to as Shadows. These are a representation of a child's fear of the dark and the unknown, and how often odd shapes would be twisted into scary monsters in the absence of light. These Shadows are an interpretation of those fears turned into something far scarier by a frightened mind. We wanted to take the shapeless masses that deceive us into believing there's something frightening in a dark room with us and turn them into something visible, tangible, and unsettling, which led us to our current

design. The Shadows are completely flat against whatever surface they're on, adapting our perception of unsettling forms in the dark by turning the way we see those unsettling forms into a living shape that can move its form about. Our respite in the real world comes from how static the masses of Shadow are, so we instead sought to make them a very eye-catching and unsettling image. The Shadows are placed under furniture and in dim areas in the map to reflect where the darkest shadows would be in real life. These areas act as spaces the player would want to avoid, and it incentivizes the player to escape their bedroom. We also took inspiration from the many unsettling depictions of shapeless creatures with many eyes, as humans find an abundance of eyes or eyes in places we don't perceive as appropriate very unnerving.

8.2.3. Level 2 NPCs

Our character designs for Level 2 were all based on school archetypes, but we subverted a few expectations about the kinds of personalities that fit into each of these archetypes. We took the "American highschool" archetypes–the forever alone White or Asian nerd, the dim-witted and needlessly cruel White or Black jock, the White prep, and the White goth types–and twisted the kinds of characters who fit into each archetype. For example, we took the jock archetype and casted an East Asian masculine-presenting person into that role instead of the nerd. A huge issue in media that features these archetypes is that these characters get no room to exist or grow outside of their archetype. Their characters so often are lacking in depth and complexity, and we sought to improve on that when creating our archetypal characters. We changed up the racial identity of each of our characters, but left it at that. We gave them room to be multifaceted characters underneath their outward appearance, and instead played with which races might be an interesting pick for certain archetypes.

Jock

Figure 23. Jock's sprite, as seen in-game.

Our jock archetype is an East Asian, masculine-presenting person. This was chosen to directly counter the stereotypical perception of East Asian people that claims they are physically weak, one-dimensional nerds with terrible acne and a horrendous lisp, usually acting as the target of a show's humor (TV Tropes, n.d.; Jiang, 2023). We wanted to instead place our East Asian character in a role that would suggest higher strength, lower intelligence, and a standoffish personality. This is how the player will perceive them, as the player character was not very close with the jock character, and their understanding of his character is superficial. His design is meant to be somewhat intimidating, as it would reflect the player character's mild fear of the jock character when they were in middle school. He is the largest of the NPC sprites in Level 2, almost comically large for a middle schooler, indicative of a combination of his actual build and how our player character perceived him. He is on the school's wrestling team, a sport not commonly represented as the pick for a jock archetype, with that honor typically going to basketball or gridiron football. To reflect his affiliation with the wrestling team, he is depicted wearing athletic wear themed with the school's primary color, red. He was not designed to be a

conventionally attractive character, deviating from the norm of any school media making the jock characters generally attractive and eye-catching, instead making him an oddball with a messy and poorly cut mohawk that has been dyed. This is not to say he is an ugly character, but we intentionally designed him to be nothing particularly eye-catching. Once the player actually speaks to him however, they will discover that he is far from what his appearance may suggest. While he is all about sports, he is also a geek and loves anime. He is a very sensitive person who cares a lot about his outward appearance and how he is perceived by his peers. His design was initiated by our project manager, an East Asian man with a fondness for sports who has family members that fit into this archetype in much the same way.

Nerd

Figure 24. Nerd's sprite, as seen in-game.

The nerd archetype in *Raveling Dreams* is occupied by a feminine-presenting Latina who appears conventionally attractive and not entirely a fit for the nerd archetype. Latino/a characters do not usually get represented in school-centric media, and when they do, it is usually a White-passing character who does not exhibit many outwardly Latino/a features. Additionally,

feminine-presenting nerds are a rarity in American media, and they are usually accompanied by a "nerd to prom queen" story where they simply change up their outfit and take off their glasses, and they've gone from a nobody to the most stunning girl at school. We leaned away from this "stunning nerd" stereotype and instead created a character who is very happy with her relatively unassuming appearance, only revealing what archetype she "fits" into when she is spoken to, conveying the idea that these identities are not only about the outward appearance, but also about the personality and interests of each character. A character can be the most stunning one in the entire game, but the way they act will determine more about them than the way they look. She is dressed simply, to avoid the route of "chic" clothes and leaning too heavily into the "nerd to prom queen" trope. Her personality fits into the archetype in a much more obvious way, as she talks about her involvement with the science-fiction club and her burning interest in archaeology.

Prep

Figure 25. Prep's sprite, as seen in-game.

Our prep archetype was filled by a feminine-presenting South Asian girl, a typically underrepresented and misrepresented group of people in American media. South Asian

characters generally fall into similar tropes to East Asians, those being nerdy tech wizards who aren't very fashion-forward (TV Tropes, n.d.). They are rarely portrayed as pretty, preppy characters who are seen as conventionally attractive. Our design reflects the common prep look, with bright colors and accessories giving her an appearance that stands out from the other characters in the school hallway. Influence for her design was taken from *Mean Girls*, using pink as her main color, all while incorporating elements from her cultural background, like the bindi on her forehead. An intentional design choice we made was to give her glasses, as they are commonly associated with the nerd archetype, and are seen as unattractive things to be removed in a "glow up". All three core members of the team need glasses for daily life, and we were eager to flip that stereotype into something more positive. Much like the nerd character, we wanted to create someone who would be visually distinct from their stereotype and invite some extra thought about which archetype to fit her into, only to demonstrate who she is when she is spoken to.

Goth

Our goth character is a masculine-presenting Black character, designed to be reminiscent of the late Juice WRLD, an icon of Black emo culture and a cornerstone for the rise of that subculture online (Hutchins, 2019). They have ripped jeans and a paint-splattered hoodie, lifting these elements directly from streetwear trends and Black fashion culture. They have dreadlocks styled to resemble those that Juice WRLD sported. We chose a Black character for our goth archetype because goth/emo characters are not commonly represented as Black. We used this as an opportunity to further demonstrate the independence between archetypes and the characters that occupy those archetypes, and, as with the rest of the characters, ensure we swap certain archetypes with fresh identities that relate far more to us as developers. Our lead programmer, an

African-American who has been deeply intertwined with emo culture for years, made sure this design was appropriate, accurate, and made them feel represented. Our goth character wears primarily dark colors, but one design challenge that came from creating a Black goth character was how monocolor the color palette was. To remedy this, we broke up their black jacket with specks of white and a purple shirt, along with silver jewelry to clearly separate their head and torso.

Figure 26. Goth's sprite, as seen in-game.

One difficulty that appeared during implementation of these NPCs was how light reflected off of Goth. They were particularly dark when the other three characters were perfectly visible in the same light, so we had to slightly amplify the light over Goth in-engine.

Fursona

The final design in Level 2 was the player character's orange cat fursona. Our player character has a connection to orange cats as a comfort item/comfort character, and we wanted to include that in Level 2 as an homage to how common it is for middle schoolers to be developing their fursonas at that time. Nobody in the group has any true connection to furry culture, so we

were sure to enlist the help of our advisor, who was well-versed in furry subcultures and how to respectfully and accurately implement a fursona.

Figure 27. Fursona's sprite, as seen in-game.

8.2.4. Level 3 NPCs

Level 3, taking place in the office maze, utilizes a style reminiscent of Corporate Memphis, an art style commonly used by companies to seem relatable and "hip", but instead coming off as terrible and soulless. We wanted to capture that soulless and inhuman style and use it for our NPCs in Level 3 to demonstrate the disconnect between the player character and their Boss and Landlord. This is done mainly through the color scheme, utilizing bright and colorful monochromatic color palettes with non-natural skin tones. They are also both lacking a face to complete the distant and unsettling appearance.

Boss

Figure 28. Boss's sprite, as seen in-game.

The player character's boss is a tall, intimidating figure who looks down on the player and searches for anything to scrutinize them about. She is a feminine-presenting, bright red character who is designed to be unsettling, dominant, and unnatural. The bright red hints to her abundance of anger and frustration, which is frequently turned against her employees.

Landlord

The Landlord is a short, round, green, masculine-presenting character who is meant to appear like a freaky slimeball of a man. His green skin is intended to make him appear gross and sleazy and almost alien, harking to his lack of empathy for the player character for a situation that the Landlord himself created when the city deemed the apartment to be too dangerous for living.

Figure 29. Landlord's sprite, as seen in-game.

8.3. Rooms and Level Art

Each of our three levels would be created as 3D assets to ease the burden on our limited artistic skills, and it would also enable us to easily rearrange parts of the level that weren't working well and duplicate things to quickly populate a room.The simple nature of these levels made for easy repurposing, as evidenced by clever asset reuse in Level 3. The room itself is a slightly modified version of the room in Level 1, with the window hole sealed, which was used as the level model for the first and second room of Level 3. These rooms were easy to duplicate and modify for each new purpose, so when the same room was used again for the Boss's office in Level 3, it was easy to add little details to make it feel unique.

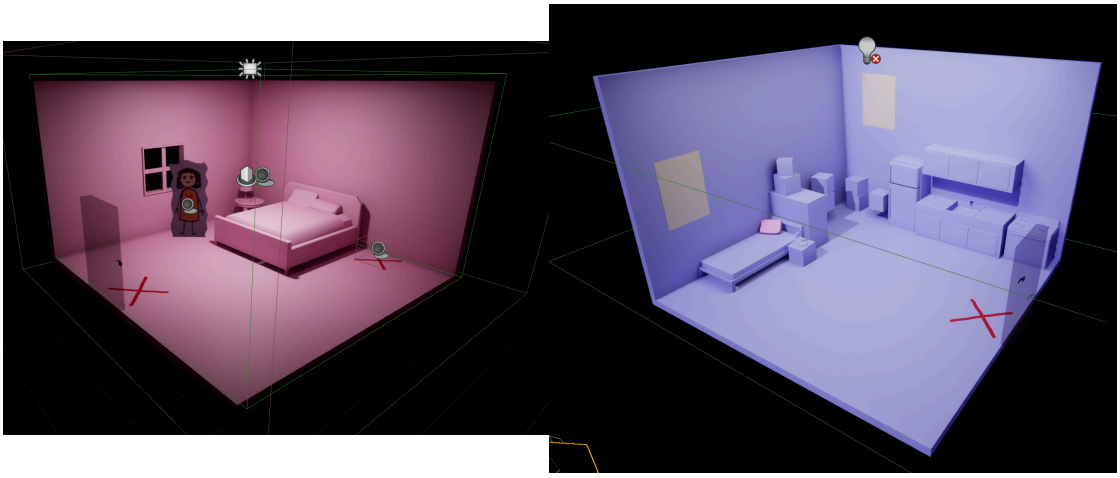


Figure 30. One room in Level 3 built with reused and repurposed 3D models from Level 1.

8.4. UI

The UI was a point of frustration for us because of the numerous technical issues we encountered and how difficult it has been to implement. This is one of the moments where C++ would have been an easier solution, as Unreal's Blueprints system has a special Blueprint for widgets, which is what we created the UI with. UI widgets are for creating menus that have static buttons however, and in our puzzles, these widgets would be overlapping. Unreal does not have a very easy way to overcome this issue, and so whenever a player attempts to connect two widgets in the puzzle menu, one of the widgets will disappear. The widgets are supposed to connect or move to one spot and stay there, but we could not eliminate the bizarre clipping behavior, and as a result, most of the puzzles are not working as intended.

Stylistically however, the UI is yet another great example of our art direction. The main menu is a journal with various drawing and journaling elements around it acting as menu navigation buttons. The button to open the puzzle menu is contextual based on the current level, with Level 1 having a toy chest, Level 2 having a backpack, and Level 3 having a briefcase as their icon. In addition, the dialogue box follows through with the overall "paper" aesthetic of *Raveling Dreams*, featuring a lined notebook paper background and a font that appears to be handwritten.

Figure 31. Example puzzle buttons, as seen in Level 2.

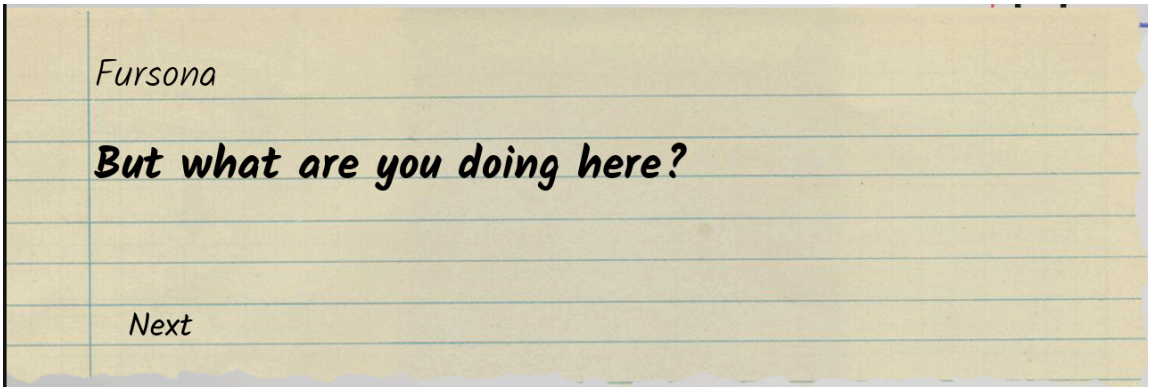


Figure 32. Screenshot of the dialogue box, as seen in-game.

9. Audio

9.1. Sourcing

Sourcing the sound effects was a mix of recording what was possible on a college campus and the surrounding area, and taking sound effects from Freesound.org under a Creative Commons license, either the Creative Commons Zero (CC0) or Creative Commons Attributions (CC-BY) license, both of which would allow the alteration and use of said sound effects for this project (Freesound, n.d.). These credits were listed in a README file inside the game folder.

The bulk of the sound effects were recorded on the Worcester Polytechnic Institute campus and surrounding area, utilizing buildings, open areas, and specialized sound-isolated areas on campus.

9.2. Mastering

All editing & mixing of sound effects was done in Reaper, a digital audio workspace (DAW) that allows for all sorts of modifications of sound files. Each sound effect, whether retrieved from a creative commons website or recorded on campus, and mixed and equalized to the same level so nothing sounds too loud, too quiet, too bass-y, etc. The most sound work was done for background sounds, where equalizing the high and low frequencies is extra important. These are the sound effects that will be playing as long as the player is in each level.

Unfortunately, there was not a ton of time for complex mixing or mastering, so sound effects are simply equalized and given a few small audio effects to improve the sound.

10. Tech Implementation

10.1. Character Movement

Our character movement was implemented using Unreal's Axis Mapping. Although Axis and Action Mapping are deprecated, we still decided to use it because the movement required for the character was very simple (WASD and/or Arrow Keys). Using the Axis Mapping, we created two Inputs, MoveForward and MoveRight. The player is able to use either set of keys to move in 8 directions. A function called SetMaterial was used to dynamically change the player's animation via materials.

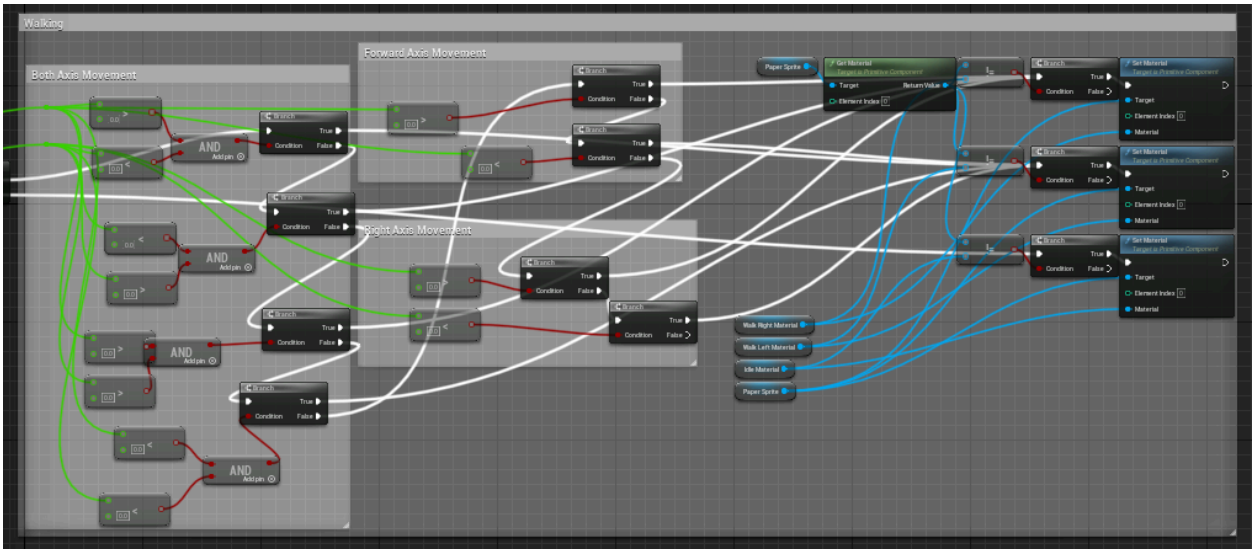


Figure 33. Character movement implementation through Blueprints.

10.2. Object Interaction

Raveling Dreams operates on a very simple interactable gameplay loop, with one key allowing for all forms of interaction. Every game-piece that the player can interact with stems from a parent Blueprint class called "BP_Interactable". Using this Blueprint, we are able to simplify common functionality such as allowing interaction only at certain ranges, selecting

which of the in-range items to actually interact with, and various indicators of item eligibility for interaction. This is achieved by giving the “BP_Interactable” class an “OnInteract” event, an “OnInRange” event, and “OnSelected” gameplay tag. Each of which gives opportunities for customization with the shared functionality that only up to one instance of the interactable super class will be selected at any given moment. NPCs, Paper Scraps, and light switches are all implemented using this system.

There was also a UI_Interact Widget which altered the interaction UI element to show the player which objects they could interact with, and how they could interact with them. These UI Blueprints came with variations for doors (enter), objects (pick up), and NPCs (talk). This way, the player would clearly tell what they could do with each interactable instead of just seeing an overall “interact” option.

In order to help the player differentiate what objects they could interact with from objects they couldn’t interact with, we created a system that allowed most interactable objects to spin when the player was in interaction range and the interact UI element appeared. When the player would leave the interaction range of that interactable, it would reset to its default orientation. The objects that this was not applied to were objects that would not have the freedom to rotate. These objects included a light switch, which would be screwed into the wall, and a scrap of paper taped to a mailbox.

Figure 34. Example of an object in rotation when within interactable range.

10.3. Player Character Randomization

The player character is randomized on each play-through to encourage the player to relate to various social groups that might not exist in their personal lives. This is done by having separate sprite sheets for each component of the player character's body, as seen in Figure . There are two layers, fill and outline. The fill layer is what changes colors for the hair color, and the outline layer would go above the fill layer. We had plans for skin color randomization, but we found it difficult to limit the randomization to believable skin tones, so we just stuck with three skin tone options. We also had multiple body types, those being spherical, rectangular, cylindrical, and triangular, which came with its own clothing layer that would go fill, clothing, and outline last, as pictured in Figure 35 .These are then mixed and matched when a new game save is made before being combined into what is essentially one sprite sheet for the gameplay. This is done using a custom shader, shown in Figure 36, that layers the sprites together into what behaves as a single image. This system helps demonstrate the shared experience of people from different cultural groups by making a character that is relatable to all sorts of people.

Figure 35. Sprite layers ready to be implemented, mixed-and-matched for randomization.

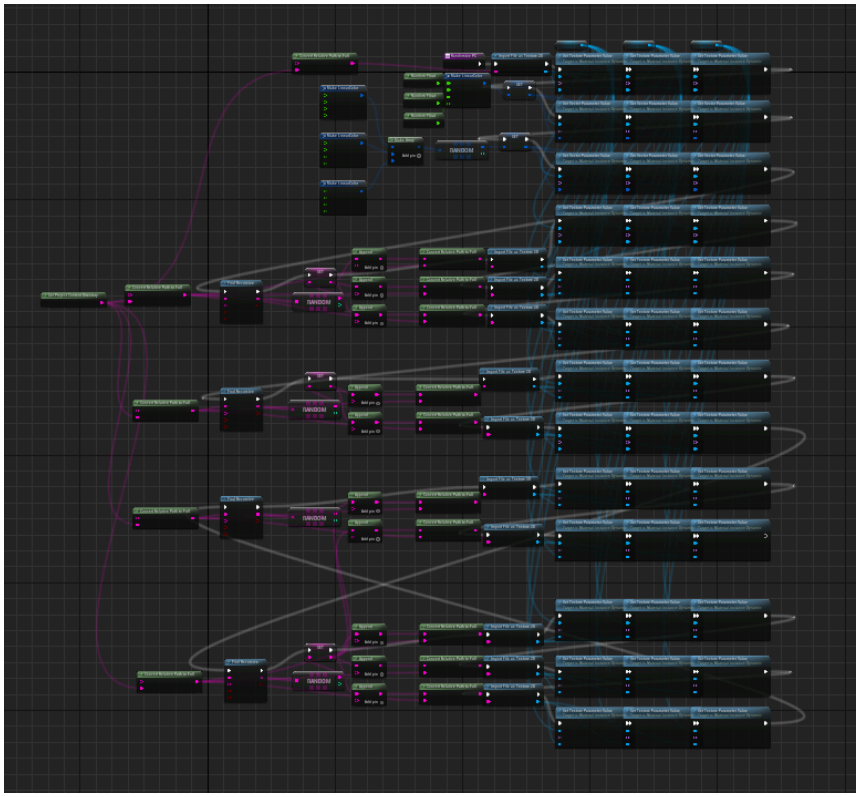


Figure 36. Player character randomization, implemented through Blueprints.

10.4. Camera

The game's camera is moved based on the player's position as opposed to being directly controlled by the player's input, allowing the level designers to have greater control over how the story is told. The camera is controlled in one of two ways. The first way is to have it face in a particular direction and sit a set distance from the player. This can be used to dramatic effect by, for instance, putting the camera very close to the player or to demonstrate the scale of a scene by pulling the camera farther away. Similarly, the other camera control method forces the camera to face towards a point in space, allowing levels to have central focal points around which the story can be told. These can be combined with other techniques to tell compelling stories with the current tools. In the example provided, the player character is navigating a maze in one of the figures, representing the overwhelming world that the character is moving through. By using an aerial view, the scale of the maze and therefore the character's distress is emphasized.

Figure 37. Default camera angle, as seen in the homeroom of Level 2.

Figure 38. Alternate low camera angle, as seen in the hallway of Level 1.

Figure 39. Alternate high camera angle, as seen in the maze of Level 3.

10.5. Limitations

Our core programmer had planned to solely use Blueprints in Unreal, as that was what they were most comfortable with, and it gave the other core members who were not as familiar with programming languages a chance to implement things on their own and assist in programming endeavors wherever possible. However, Blueprints were not a perfect system, and

there were times where our core programmer felt C++ would have been a far easier solution than wiring a few dozen Blueprints together to solve what could have been a simple C++ fix. However, there were also moments where Blueprints were easier to use than C++, and our programmer ultimately chose what was easier for them. Unfortunately, by the time we realized that C++ might have been an easier solution, we were already well into working with Blueprints and integrating C++ and Blueprints together is far more of a hassle than just continuing with Unreal's Blueprints system.

11. Evaluation (Playtesting)

11.1. Process

At each playtesting opportunity, we showed our current prototype of *Raveling Dreams* with fellow WPI undergraduate, graduate students, and guests for voluntary playtesting. We asked users to play our prototype for 10-15 minutes from the beginning of the level(s). Player activity included navigating the level, interacting with level objects and NPCs, and solving puzzles. Playtesting data was collected via observations and user surveys. We observed the player as they progressed, including expressions, comments, and where they were having trouble or showing interest. Observation notes were documented in a private spreadsheet accessible only to team members, separated by prototype version and player.

We then asked players to complete a short voluntary feedback survey. The survey asked users to rate certain aspects of the prototype's gameplay and aesthetics on a scale from one to six, with one generally meaning "very poor" and six generally meaning "very satisfactory." This included rating the cohesiveness of the level's art and how easy to understand were specific level puzzles. For each rating, there was room for optional comments for users to expand on their rating. The survey finished by asking for any additional comments or feedback. The user survey responses will be stored in a private spreadsheet accessible only to team members.

After a playtesting session, we would scan our playtesting data as a team and identify areas of strength and areas of improvement in our current design and implementation based on analysis of the playtesting responses. First, we would look at the feedback results and identify patterns and themes in the observations and survey responses. From there, we would identify

where changes are needed based on these feedback patterns and themes, or if any changes are needed at all in any specific aspect of the prototype. We must have considered how this feedback would inform our development decisions moving forward, including how it relates to our design and implementation and what the feedback addresses in our current design and implementation. For each prototype evaluation, we should be addressing concerns around fun, intuitiveness, navigability, and consistency.

The team participated in three total playtesting opportunities: Protofest, Alphafest, and the Playtest-O-Rama colloquium. Additional feedback and iteration was completed through internal playtesting among team members and playtesting by our advisors, which took a much more informal approach.

11.1.1. Protofest

Protofest was the first instance where we got playtesting feedback, and we were able to gather a physical Level 2 for players to walk through. It was created with cardboard decorated with pen and marker, with NPCs drawn on to sticky notes and held in place with weights. The paper scraps that players collected were small lego pieces, and when they grabbed them they would be given physical scraps of paper that fit together to give the player a code. We were able to create the entire level end to end and allowed players to observe their environment and choose what to do next. This open-ended method of playtesting allowed us to get some insight into how players wanted to interact with our NPCs, what their first thought was, and how they responded to dialogue differently from each character.

11.1.2. Alphafest

By the time Alphafest had come, we managed to get Level 2 into a digital format. It was a much more linear format as we could not get the puzzle implementation working in time, but it still allowed for players to go through the story we had planned out. Players enjoyed the art style, although the layout and teleportation between the wings and lobby confused players, which we took into account while redesigning. Some players asked for a map so they could see where they were, but our level designer imagined a simpler solution. All the segments of the level were condensed into one large room, which was far easier to navigate and far less confusing when players wanted to think about where they were.

11.1.3. Playtest-O-Rama Colloquium

We had our first end-to-end build ready for this hour-long playtesting session, and it was here that we got some of the most effective feedback. The puzzles weren't completely finished, mostly because of limitations with widget Blueprints, but we were still able to get very valuable feedback about the pacing of our game. People mentioned that Level 2 felt far too linear, partially because the puzzle system wasn't functional and was the most interesting part of the gameplay in Level 2.

11.2. Feedback and Implementation

Because we are following an iterative process standard to game development, the loop of receiving feedback and implementing changes based on that repeats multiple times. The cyclic process goes through the steps of conceptualization, implementation, playtesting, and evaluation multiple times until the conclusion of the project at the end of the school year. A new version of

our game prototype would be completed before going out for the next round of user testing until it is ready for final distribution. Each iteration should expand and build on the content of the game until it becomes a visually engaging experience with tight design, art, writing, and audio design in addition to addressing concerns of fun, intuitiveness, navigability, and consistency.

An example of this iterative design process in action is in regards to the layout of Level 2. During Protofest, when our first prototype made of cardboard was being presented, having each area in the level be a separate space was the easiest solution. Players found it a bit disorienting, but it was not noted as a common complaint, so it was not changed much for the first all-digital build for Alphafest. However, at Protofest, many players complained about how disorienting the transitions between each space was. This was rectified in the build for the playtest-o-rama event, where the feedback for the level design was far more positive, and there were no comments mentioning disorientation or confusion about where they were in Level 2.

12. Rescoping

12.1. Scrapped Content

Due to the limited development time coupled with delays in progress because of our other obligations and overall lack of development experience, some features had to be cut to re-scope the project to still be achievable with the time we had. The most notable piece of cut content is the entirety of our original Level 3. This level ended up being scrapped because it contributed the least to the overall story of the game and player character. We also wouldn't have had enough time to implement all four levels in time as originally intended. Our current Level 3 is actually the originally designed Level 4.

In the original Level 3, the player character explores a college frat party where they must find a female friend who has gone missing at the frat house, implied that the player character left them alone and failed to contact her that night in the real world. The main mechanical puzzle involved mixing drinks at the basement bar to create specific color combinations for party-goers, getting them drunk and causing them to inadvertently reveal dialogue clues hinting at how to find the player character's friend. Item clues were also involved, which could be revealed to NPCs for additional dialogue clues. While we thought that the design of the level on-paper was our strongest overall, we thought it was the weakest in gameplay because the level was poorly planned out with subpar level of detail by the end of pre-production in the fall. The team failed to solidly design the strings of dialogue and item clues that would lead the player to deduce that they need to unlock the front door because the player character's friend ended up outside the frat house. What made this design particularly difficult was justifying narratively and in-world how

the friend progressed through the frat house and ended up outside without the player character noticing.

Another set of cut content includes short playable segments set in the real world meant to supplement the main narrative in the dream world. These segments were designed as a way to show how the actions and revelations in the dream world levels affect the player character in their real world life. They were also meant to clarify the relationship and visual difference between the real world and the dream world in *Raveling Dreams*. During development, we recognized that these playable segments would have to be scoped down to short cutscenes of still images, but ultimately the entire idea of segments in the real world had to be cut due to lack of time.

One final feature that had to be cut were randomized paper scraps and locker locations for Level 2. The solution set of paper scraps on each playthrough was supposed to be randomized from a total set of paper scraps. Thus, the specific location of the solution locker to end the level was also supposed to be randomized, though it would always be in Goth's wing. Puzzle randomization was supposed to add some depth to our gameplay by creating different puzzle progressions on different playthroughs. However, we determined that this randomization is not imperative to our project as the existing, non-randomized puzzle functions perfectly fine as a challenge. Puzzle randomization was cut also due to the lack of time left in development. The team decided to allot the man hours left to work on much more important systems, such as audio and the puzzle interface, or much more personally interesting systems, such as player character randomization.

With how much time was spent on 3D modeling and project wrap-up by the sound lead, music was cut from the final build as it would simply take too much time to create, mix, and implement in a meaningful way. Sections of songs were created, but they remain unused.

12.2. Future Development

The team made the goal to create a cohesive, complete product by the end of the MQP and have stuck with that timeline. No decisions were made or discussed by the team to continue development past graduation and May 2024. While there is no professed commitment to continuing development on *Raveling Dreams* in the future, there are areas where we can improve or add project content. Most of the additional content we would like to develop is scrapped content. This includes the originally designed Level 3 and the real world segments. The real world segments would be cinematics about the player character's life in the real world that plays before the first level, in between levels, and at the end of the game to provide a more detailed and cohesive narrative; or, they would be short playable segments set in the real world that would serve the same narrative purpose as the cinematics.

Another avenue for future development is to clean up the gameplay mechanics, which include completing and polishing the puzzle systems and patching any bugs left.The puzzle system has reached a level where it is completely functional in Level 1, and the other two puzzles are not functional. Those two levels are able to be completed through dialogue and basic interactions instead, allowing for a playable end-to-end build, as the puzzle systems created for those two levels were not successfully and functionally implemented. While there are no major bugs that prevent gameplay progression, there are minor bugs tracked on our Trello kanban board that still need addressing.

Beyond these developments, any potential future development would focus on levels of visual and mechanical polish. One visual feature that would improve the game is an in-level cinematic sequence where dialogue can play automatically and NPCs can move in, out, and around the scene. Visual polish would also include better detailing on the 3D models, better texturing on the 3D models, more 3D props, better lighting, and improved UI with more thematic and visually clear buttons. Mechanical polish would include the implementation of puzzle randomization in Level 2, where the solution set of paper scraps will be randomized at each playthrough.

Our main focus at the conclusion of this project, however, will be distribution of *Raveling Dreams*. At the minimum, we want to publish independently through itch.io as it is one of the easiest platforms to distribute small indie interactive projects on. Ideally, though, we would distribute *Raveling Dreams* through Steam and/or Epic Games, which have much stricter and longer application processes but a wider reach to potential audiences. We are prioritizing distribution because we believe having at least one published game on our resume already will be an advantage in the application process to hopefully improve our chances at landing a job in the games industry.

13. Retrospection

Raveling Dreams was the first long-term interactive project for each individual member of the core team. Due to our inexperience, we consider our development in retrospect to have been inefficient and, at times, overwhelming. However, the team did succeed in producing a self-contained, reasonably scoped interactive experience that we can be proud of as our capstone project coming out of WPI's IMGD program. More importantly, we experienced first-hand the difficulties of the long-term game development process and, as a result, have learned and grown as developers. This is especially true for our small team with only three core members, since each of us has had to work in disciplines outside of our comfort zone. Overall, the team showed strong resiliency in doing so, in addition to the troubles that come with the MQP development process and the school year in general. Because of that, we each came away with a stronger, interdisciplinary skillset that will open up avenues for potentially undertaking positions at game studios that we would have otherwise not qualified for, if it were not for this project experience.

Moreover, we cannot understate the importance of the soft skills needed to be a game developer working within a team. While the project manager's responsibility is to keep the team on track, each member should have strong organizational, communication, and interpersonal skills that will allow them to keep track of their own tasks, work efficiently, keep the workflow smooth, and build rapport with peers. As rookie developers, there are many pitfalls that are difficult to predict and prepare for, but the building of these soft skills will aid in either avoiding or climbing out from these pitfalls. To supplement this report and provide a resource to future MQP teams, we have provided extensive recommendations for the development process below.

13.1. Recommendations

The development of *Raveling Dreams*, like many other game development cycles, was one that faced many challenges both in the process and in the personal lives of our team members. Through these challenges, however, we as a team and individually came away stronger and with valuable knowledge that we would like to pass on to future IMGD MQP teams, particularly on three topics:

- development within Unreal Engine,
- team management and the development process, and
- making important decisions about art and gameplay direction.

13.1.1. Developing in Unreal Engine

A game engine as powerful as Unreal Engine can be difficult to manage and work with, especially for beginner developers unfamiliar with working directly in game engines in general. A beginner developer unfamiliar with Unreal Engine will likely develop much of their project using Blueprints, Unreal Engine's proprietary visual scripting system. There are different types of Blueprints with specific use cases, like Blueprints specifically for player-centric design or environment-centric design. Some examples of player-centric Blueprints include player collision checks and triggers and binding movements and actions to certain keys. Environment-centric Blueprints include creating and placing actors around a level that the player can interact with, giving the player interactables to teleport them around a map, and placing trigger volumes that will change how a camera behaves.

Blueprints are a very strong and easy-to-grasp scripting language, but it was not without its fair share of difficulties. Our biggest struggle was surrounding UI implementation. While we

were easily able to create text boxes and interaction widgets that told the player what and who they could interact with, they were far less straight-forward when implementing our puzzle systems. We used UI widgets to display our puzzle pieces when they were in the puzzle menu, and we discovered that Blueprints did not have a solution to widgets disappearing when being stacked on top of one-another. They were created simply as UI elements, and were never intended to be used in gameplay. This problem plagued us for weeks, and we tried many fixes to no avail. This problem could have been fixed in a far simpler manner using C++, but cross-integration between languages in Unreal is a difficult task that we did not have the bandwidth or time to embark on.

Aside from their approachable nature that ensured our lead programmer was never too lost in syntax and always had a plan for how to proceed in the code base, Blueprints provided very easy integration for our sound lead. Blueprints were open ended, and could easily have extra pieces added on to their tail, meaning audio cues could easily be stitched on to an interaction Blueprint for opening a door. It also meant our sound lead and writing lead, who are both unfamiliar with C++, were able to create complex systems on their own, giving the programming lead more time to complete the core game mechanics.

The greatest strength of Unreal Engine 5 was its vast market of plugins that solved problems that we were not capable of solving. Our lead writer was able to use the Not Yet Dialogue system to allow for easy integration of complex and branching dialogue paths, and would easily plug in to our pre-existing Blueprints. Our art assets utilized a system called Paper2D, a plugin that solved most of the headaches of getting 2D sprites to render well in a primarily 3D engine like Unreal. Systems like these saved us hours and hours of from-scratch

development and gave us the freedom to fix our own bugs and develop our own necessary systems, like character randomization and the drag-and-drop puzzle menus.

13.1.2. Team Management and Development Process

All future IMGD MQP teams must understand as early as possible that a dedicated project manager will be imperative for a complete and successful project. This still applies even if no member has significant production or project management experience. At least one member must voluntarily step into the role or be assigned to it. Ideally, the dedicated project manager should be decided naturally based on team dynamics as the one member who seems most capable at organizing the team and has their full mutual trust. Dedicating multiple project managers would also be ideal, especially for larger teams of five or more members developing larger projects. The project managers can support each other and allow for multiple perspectives on the team's process, helping to alleviate any pressure that would befall a sole project manager.

Whoever the project manager(s) may be, there is a chance that the IMGD MQP is their first experience handling a project to such a large scale. In case an inexperienced project manager is worried about the responsibilities of the new role or doesn't know where to start, the project advisors will be a great resource to talk about solid starting practices. This also applies to other team members if they are unfamiliar with individually managing a large scale project. A faculty member who would be good resources is Professor Walt Yarbrough. We also recommend connecting with previous IMGD MQP teams for tips and advice on the development cycle and project management. Regardless of who the contact is, practice courteous and professional communication for the best chances of receiving a helpful response.

Now that the team's project manager has accepted the role, they must become familiar with different workflow management processes. This may include generally researching sprint

systems where the team breaks down individual tasks to be done over a period of one or multiple weeks, or learning about well-established processes such as Agile, Scrum, or Waterfall. Any of these processes will help a team execute an organized development cycle with internal deadlines. Every team works differently, however, and there is no guarantee any of these processes will be effective. Teams will respond to different processes in different ways, either favorably, negatively, or a mix of both. One of the duties of the project manager should be to observe how the team members prefer to work, how they react to deadlines, and how they react to currently implemented workflow structures and task-keeping. These observations should inform the team manager on what workflow management practices work best for that specific team. Early in development, the project manager should keep options open and never commit to one system in case the team responds negatively to one workflow management process. They must be able to adjust to a system that works for everyone if they notice the current process is not working favorably for the team.

Before making any systemic changes, however, the project manager should discuss with the team if they notice the current process is not working. The hope is that the individual members can provide valuable feedback on the current workflow management and identify any pitfalls the team are falling into. Some important workflow topics for the team to think about and discuss include:

- how well deadlines are met,
- how well individual members track, communicate, and update tasks,
- how well individual members can identify and break down potential tasks,
- how well individual members can estimate the length of time specific tasks can take,
- if tasks are consistently being delayed or pushed off,

- the energy and manner of team members during meetings,
- the general level of productivity during team meetings, and
- if team members work on their own time outside of team meetings.

If the team is not giving much feedback or is having trouble identifying exactly where work is falling short, take that as a sign it is probably worth trying a different process to observe if the team responds more favorably.

A general rule of thumb for project managers is to always assume tasks and implementation will take longer than initially estimated. The team should account for this and make time to potentially extend the deadlines for these tasks. To make this time in the development cycle and to better prioritize tasks, the team should discuss the most important aspects of the game to immediately work on and the least important aspects of the game to push off and potentially cut. Be ready to adjust when tasks should be completed on the timeline and to whom tasks are assigned to ensure the project is at least complete and interactable at the end. As such, do not become overly attached to any design element and be ready to cut planned content. Before making decisions to cut, however, first consider the possibility that work is not being broken down into small enough tasks or the team is not yet accurate in estimates of how long certain types of tasks will take. The team should expect to adjust the scope in this way multiple times during the development cycle until the team eventually settles on a project scope that is achievable by the end of the project timeline.

As a project manager, a continually diminishing scope may seem concerning, but as long as the team has identified a minimum viable product during the design phase, the most important development goals should still be met by the end of the project. Every team should document a clear vision of a reasonable minimum viable project. To assess if the team's vision of a minimum

viable product is reasonable, team members should discuss a few core questions and come to a consensus:

- How would the project interact at its bare functional state?
- What are the most important feature(s) that would be included in this state?
- What features would be left out?

With a clear and well-defined minimum viable product, the development team can utilize a “layered” design approach to iteration. A “layered” design approach involves defining additional tiers of content past the minimum viable product as prioritized benchmarks once previous tiers have been met. For example, on our project, the minimum viable product would involve a 2D sprite walking around 3D rooms. The next priority tier of content would include NPC sprites and NPC dialogue. The player character randomization is an example of a tier of content meant for much later in development, as it is not imperative to the function of our game, and thus not immediately prioritized. It was labeled a stretch goal meant to be polish content and add engagement to the project if we identify we have the work hours left to implement it. We encourage the project manager to implement this “layered” design approach to iteration.

Finally, the project manager needs to recognize from the onset of the project that “life will happen”. Expect the team will be unable to meet all the deadlines to fully meet their original vision. Individual team members, including the project manager themself, will most likely encounter personal struggles and other obligations as college students that take time away from the project and delay progress. One of the advantages of the iteration process and good workflow management is they allow the team to develop one step at a time and prioritize the most critical systems and features. The team will have multiple opportunities throughout the project to ensure

these critical systems and features are well-implemented and clear to the player before moving on to additional features.

13.1.3. Deciding on Art and Gameplay Direction

An important consideration to think about is about how scope and rescoping will affect the art direction and gameplay mechanics. While unique art styles are very interesting and can take a game to the next level on visuals alone, developers must consider how much extra work implementing that art style will take. The skill levels across the team will determine how easy or difficult it will be, as well as how long it may take. We had to make plenty of concessions about art direction without a dedicated artist. Our sound designer had to take on the role of a 3D modeler and learn several new concepts and pieces of software at the same time, which not only led to the 3D models being output slower than if someone with more skill had found their way onto the project, but it also ate into time that could have been used for sound design and left the project with a much smaller and less developed soundscape than desired. Developers must ensure they either have the bandwidth to take on so many roles, or they must ensure that they have the right developers for the job. Misallocating your developers and sending them to learn new disciplines during active development will consume a lot of time that may have been better spent elsewhere.

Additionally, tech implementation is a difficult and time-consuming process, and developers must ensure they either have the skills, bandwidth, or manpower to handle learning new engines, new techniques, and growing pains as their production cycle begins. If gameplay mechanics are proving troublesome, increased manpower and mental bandwidth must be allocated lest enough time be spent bashing one's head against a wall that they feel they can’t move on until that mechanic is fixed and implemented.

References

Beyond!. (2024, January 30). Why are game devs obsessed with this hairstyle?. IGN. https://www.ign.com/videos/why-are-game-devs-obsessed-with-this-hairstyle-beyond-clips

Dowd, T. (2024, February 28). The ‘Killmonger cut’ is everywhere in games, here’s why the industry needs to fix this. IGN. https://www.ign.com/articles/killmonger-cut-everywhere-games-spider-man-tekken-8

Freesound. (n.d.). *Frequently asked questions*. Freesound.org. https://freesound.org/help/faq/

Hutchins, J. (2019, March 14). Juice WRLD and the rise of emo rap. *Arts at Michigan*. https://artsatmichigan.umich.edu/ink/2019/03/14/juice-wrld-and-the-rise-of-emo-rap/

Jiang, J. (2023, April 25). ‘You’re going to lose anyway’: Asian American athletes face stereotypes across sports. *Pepperdine Graphic*. https://pepperdine-graphic.com/youre-going-to-lose-anyway-asian-american-athletes-face-stereotypes-across-sports/

Lopez, G. (2016, April 17). A popular video game now randomizes your race and gender — and many white men are furious. *Vox*. https://www.vox.com/2016/4/17/11442730/rust-experimental-race-gender-random

Omori [Video game]. (2020). OMOCAT, LLC.

Rust [Video game]. (2018). Facepunch Studios; Double Eleven.

Super paper mario [Video game]. (2007). Intelligent Systems; Nintendo.

Supergiant Games. (n.d.). *Hades*. Steam. https://store.steampowered.com/app/1145360/Hades/

Taniguchi, G. (Director), & Okouchi, I. (Writer). (2006). *Code geass: Lelouch of the rebellion* [TV series]. Sunrise; Crunchyroll.

The wild at heart [Video game]. (2021). Moonlight Kids; Humble Games.

TV Tropes. (n.d.). *Asian and Nerdy*. tvtropes.org. https://tvtropes.org/pmwiki/pmwiki.php/Main/AsianAndNerdy

TV Tropes. (n.d.). *Bollywood Nerd*. tvtropes.org. https://tvtropes.org/pmwiki/pmwiki.php/Main/BollywoodNerd

Walker, J. (2007, September 19). Super paper mario: Creasing us up. *Eurogamer*. https://www.eurogamer.net/super-paper-mario-review

Waters, M. (Director). (2004). *Mean girls* [Film]. Broadway Video; Paramount Pictures.

Wickens, K. (2021, June 9). I amassed an army of spritelings in the wild at heart and might actually cry. *PC Gamer*. https://www.pcgamer.com/the-wild-at-heart-puzzle-adventure/

Yume nikki [Video game]. (2004). Kikiyama; Playism.

Appendix

Appendix A: Excerpt from Level 2 Design Document

- **Second level:**
    - Stereotypical school tropes
    - Narrative:
        - Protagonist never really interacted with peers in school, so all the students are stereotypical and in one-dimensional groups because of the Protagonist's superficial understanding of their peers
        - Believes bully stole their cringe sketchbook with their "cringe" anime art
            - Believes their bully hid their sketchbook and they have to figure out where it was hidden
        - The sketchbook is in a specific locker but the dream world has changed the location of said locker
            - Must interact with other students, including the Protagonist's own bully, to find said locker
        - Find the sketchbook in a Goth kid's locker
            - By the end: remember it was actually a Goth who stole it
        - Confront the Goth, but in a nicer way
            - Protagonist originally lit into the Goth kid in real life
    - Characters:
        - Protagonist (dream)
        - Bully (dream)
        - Goth kid (dream)
        - Orange cat fursona (dream) [Guide NPC]
        - Nerd (dream)
        - Prep (dream)
        - Student peers (dream) [minor/background]
            - Sitting on top of lockers looking down at us (different demographics based on wing)
    - Objects:
        - Sketchbook [Goal]
        - Paper scraps (pick up)
        - Cipher note (pick up)
        - Whiteboard with Caesar cipher hint
        - Locker(s)
        - Trash can
        - Plastic toy grabber
    - Objective/Puzzle:
        - Find who stole the sketchbook
        - Find embarrassing sketchbook that was stolen:
            - Collect cipher and paper scraps:
                - Find the wing it is in

            - Find specific locker that it is in
- Rooms:
    - School lobby
        - Homeroom
    - Different wings with different sets of lockers
        - Different demographics based on wing, different color palette for lockers/room?
            - Jocks wing
            - Goth wing
            - Nerd wing
            - Prep wing
- Puzzle design:
    - Intro
        - Everyone is in School Lobby
        - Jock bullies Protagonist and says he's a hall monitor, so don't let him find them or he'll stuff them in the homeroom supply closet
        - Everyone says he's an asshole and leaves
        - Fursona picks you up, notices that you don't have your sketchbook and says to go find it. Maybe someone stole it? Best get on that–says it'll be in the Nerd wing for any help.
    - Fursona
        - Guide NPC - Basically a hint machine in the Nerd wing
    - Cipher:
        - Lockers have shifted, not in locations we thought they were.
            - Solve cipher to find where right locker is
                - Find which wing it is in
                - Find which locker is the right one in that wing
            - Solve another puzzle to unlock locker
        - Player gets the name of the wing but it is ciphered
            - Have to solve cipher to find which wing it is
                - Fursona drops the piece of paper with all the letters after talking with player (School Lobby)
                - There's a Caesar cipher infographic on the homeroom chalkboard (Homeroom)
        - Solve cipher directly in inventory: Cipher scrap displayed, four letter slots that scroll through alphabet to solve cipher *(see below)*

Appendix B: Excerpt from Art Direction Design Document

- **Refer to Miro board** https://miro.com/welcomeonboard/YXJtZUZ6bmg0dVh1RTZOOFpabmdnaXNhSWFZR0dURmJESENuNFljblc4ZUZXN0pEd1pMYnFZU3Zqc1dLZ2M5N3wzNDU4NzY0NTMzOTEyNzAyMjkzfDI=?share_link_id=314581007727
- No dedicated artist, simplified art style
- 3d modeled backgrounds and environment objects in isometric view
    - Dream world background: trippy dream videos
        - Dream world: isometric room won't take up entire screen, have space for background art
        - Objects in background floating/flying around
    - Vertex colors for rooms and 3d objects
    - Static overlay and sound design for inciting incidents/climaxes
    - Any interactable objects should be 2D
        - Key items should be 2D to stand out from 3D
            - Matches the paper-hand drawn style of the Level
        - Blocks to push around can still be 3D in the space but look drawn in 2D
- 2d sprites in dream world- styled to be like doodles on paper, pieces of paper being ripped out
    - In the protagonist's room are different dream journals from the different stages of life, so the NPCs that appear in the dream have evolving art styles (still making sure to keep it simple).
        - First level is shitty crayon shapes like a kid would make
            - First level: drawing of kid versus adult:
                - Just a ball and limbs, look the same but different sizes
        - Second level is shitty weeb doodles
        - Animal Crossing/Miis-basic shapes to make faces
            - Party theme is "Wild West"
        - Fourth level is boring corporate artstyle (Corporate Memphis) without faces
    - **Walking animation: three frames of the doodle on the paper walking, but full model just slides across level**
        - Stack paper on top of each other as animation, like you're stacking the different doodles together
    - Protagonist would keep the same basic artstyle throughout the game for player clarity.
- 2d sprite in real world- not designed to look like doodles on paper but still 2d for our sake
    - Basic "box and stick" figure art with hair and clothing
    - **Puppet rig**
- **Think of randomized Protagonist and minor NPCs as paper dolls with dress up pieces**
    - **"Scribbles changing colors and few drawings of hair and clothing"**
- Real world backgrounds and environment objects are completely in black and white and boring as shit

Appendix C: Protofest Feedback Survey Questions

Protofest Feedback - Raveling Dreams MQP

Thank you for stopping by our Protofest table! Please fill out this form in full honesty.

1. **How simple were the level's puzzles to understand, overall?**

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very complicated

2. **How simple was the cipher puzzle to understand?**

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very complicated

3. **How simple were the paper scraps puzzles to understand?**

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very complicated

4. **Additional feedback on puzzle simplicity?**

5. **How difficult to solve were the puzzles, overall?**

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very hard

6. **How difficult to solve was the cipher puzzle?**

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very hard

7. **How difficult to solve were the paper scraps puzzles?**

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very hard

8. **Additional feedback on puzzle difficulty?**

9. **General feedback? Anything you'd like to say?**

Appendix D: Alphafest Feedback Survey Questions

Alphafest Feedback - Raveling Dreams MQP

Thank you for stopping by our Alphafest table! Please fill out this form in full honesty.

* Indicates required question

1. **How simple to understand was the movement?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very simple

2. **Additional comments on movement?**

3. **How simple to understand was the layout of the map?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very simple

4. **Additional comments on the map?**

5. **How well do you recognize the environment as a middle school?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very well

6. **Additional comments on the environment?**

7. **What gender do you think the main character is, based on visuals alone?**

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Neither
- ◯ Unsure

8. **How did the art style make you feel?**

Mark only one oval.

- ◯ Angry
- ◯ Confused
- ◯ Happy
- ◯ Nostalgic
- ◯ Sad
- ◯ Other: _____

9. **Additional feedback on style?**

_____

_____

_____

_____

_____

10. **How easy to follow the narrative?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very easy

11. **Additional comments on the narrative?**

12. **How clear were the level objectives?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very complicated

13. **Additional feedback on level objectives?**

14. **How simple was the cipher puzzle to understand?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very simple

15. **How easy to solve was the cipher puzzle?** *

Mark only one oval.

	1	2	3	4	5	6	
Very	◯	◯	◯	◯	◯	◯	Very easy

16. **Additional feedback on cipher puzzle?**

17. **Did you run into any bugs? Please explain.**

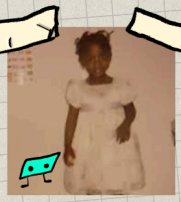
18. **General feedback? Anything you'd like to say?**

Appendix E: Project Presentation Day Slides

The Team

Ethan Chau
Project Manager & Writer

Aidan von Conta
3D Artist & Audio Lead



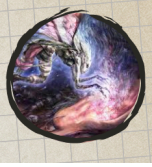
Luca Wol
Programmer & Designer

Ellie Kim
2D Artist

Abi Rauch
Concept Artist

Rose Bohrer
CS/IMGT

Ashe Neth
Programmer



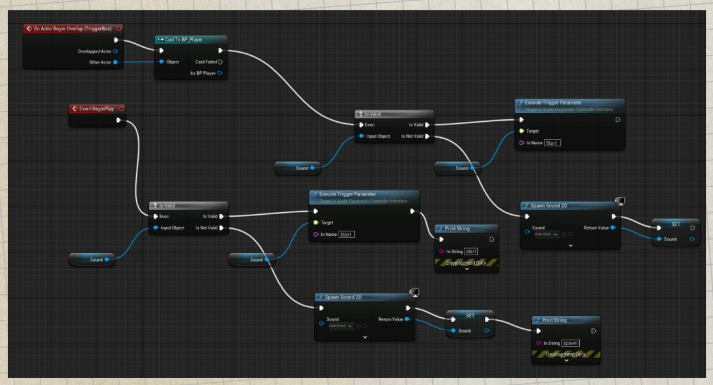
TheDyingSun
Art Consultant

Farley Chery
IMGD

Sound Design

Ambiance

Created a system for background sounds with continuous playback, allowing the background noise to stop and start from its current point on the wavelength.

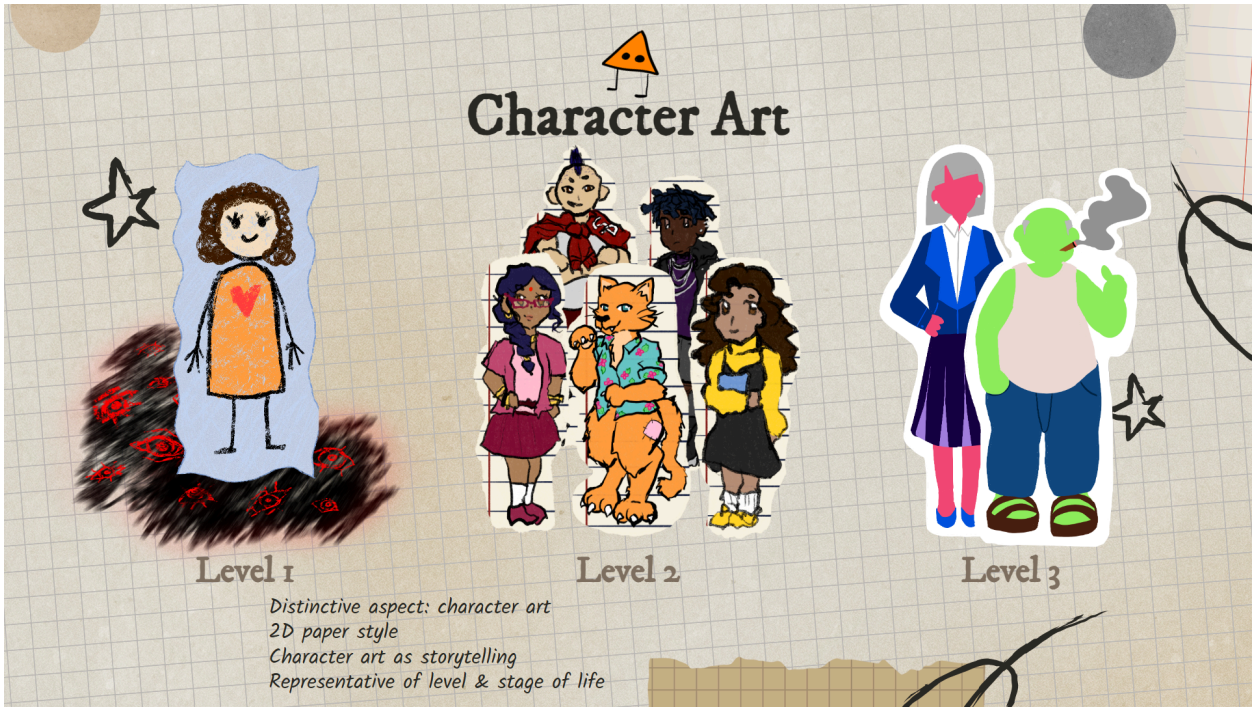


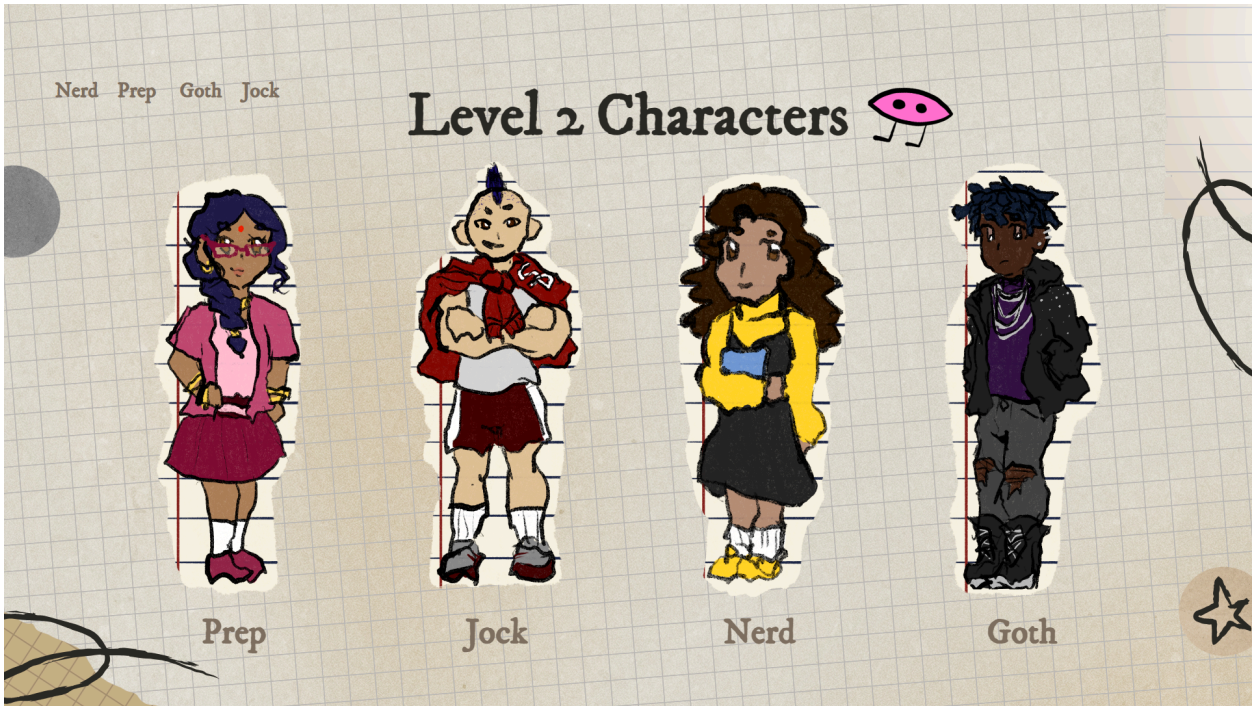
I was in bit of a crunch from spending so much time working on modeling, so I focused on implementing just what was necessary

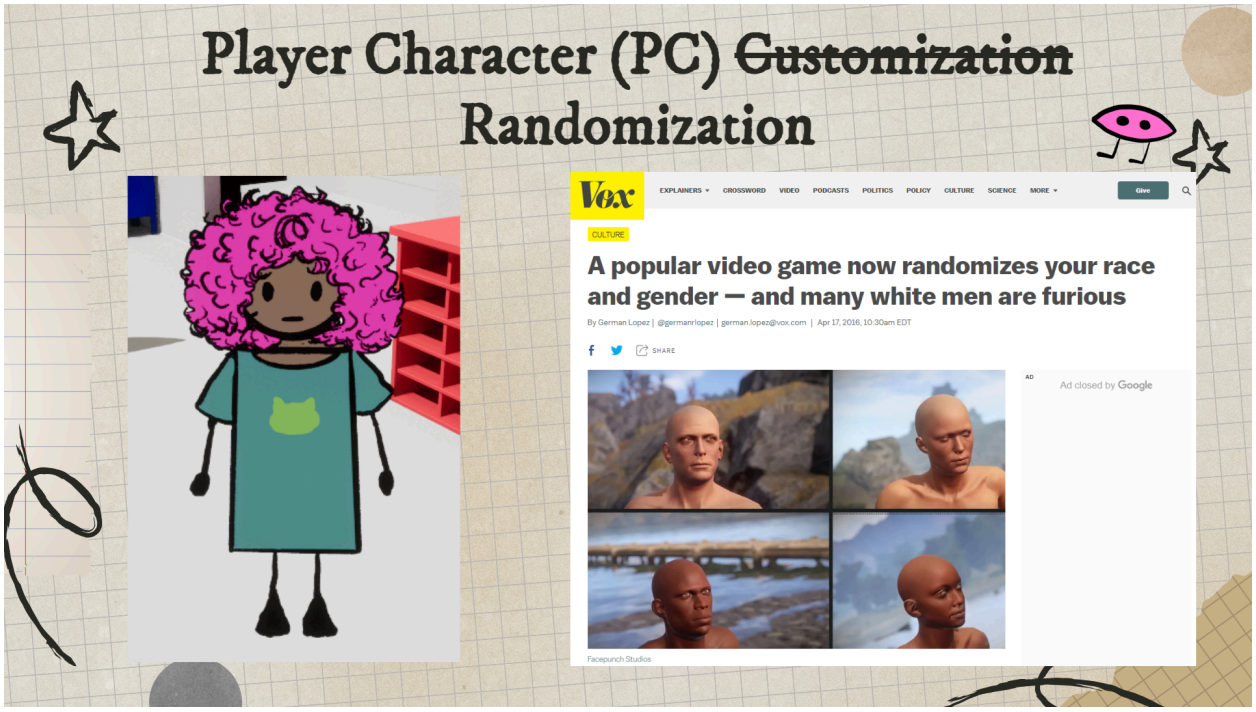
Recording

2/4 background sounds taken and mastered from freesound.org (under a CC-BY or CC0 license), everything else was recorded and mastered by me.


70+ models!
Level Design
No asset store!
Level 1
Spooky in some areas
Homey in others
Level 2
Bright and sterile, familiar but not in a comfortable way
Level 3
Messy, confusing, difficult


Character Art
Level 1
Level 2
Level 3
Distinctive aspect: character art
2D paper style
Character art as storytelling
Representative of level & stage of life


Nerd Prep Goth Jock
Level 2 Characters
Prep
Jock
Nerd
Goth


Player Character (PC) ~~Customization~~
Randomization
Vox
EXPLAINERS
CROSSWORD
VIDEO
PODCASTS
POLITICS
POLICY
CULTURE
SCIENCE
MORE
Give
CULTURE
A popular video game now randomizes your race and gender — and many white men are furious
By German Lopez | @germanrlopez | german.lopez@vox.com | Apr 17, 2016, 10:30am EDT
SHARE
AD
Ad closed by Google
Facepunch Studios

PC Randomization - Hair as Expression



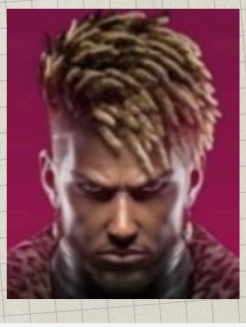




Hair is important to our identities & expression


PC Randomization - Hair as Expression




IGN

Upgrade to PLUS

Prince of Persia: The Lost Crown • Jan 30, 2024

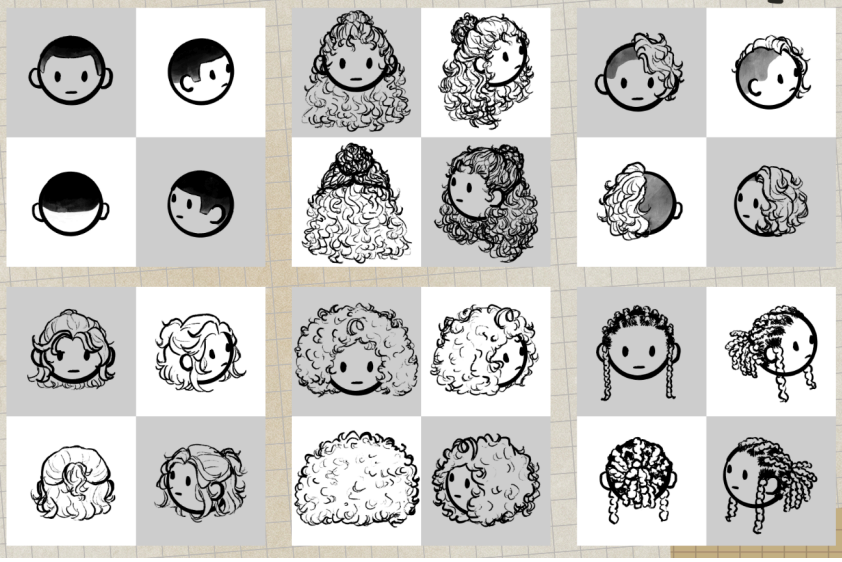
Why Are Game Devs Obsessed With THIS Hairstyle? - Beyond Clips

[redacted]

30 January, 2024

I live in Baltimore. Over 60% African American. Not once have I seen anyone rocking the Killmonger.

PC Randomization - Hair as Expression



16 possible combinations of body type and hairstyle!

Identity Through Dialogue

Mom

Hm? A-neoi, I thought I told you to go to bed.

Next

Jock

I just think Orion High School Host Club is better!

Next

Mom

Hm? Mijo, I thought I told you to go to bed.

Next

Jock

What! My mom ferments the kimchi herself!

Next

Growth As Developers

Crucial experience as a producer on a long-term project and as a developer in Unreal Engine 5.

Dove headfirst into modeling and level design as a sound designer while learning UE5.

Leadership in a programming setting and simultaneous design work.

- IMGD 2024 -

Thank You! We're Doing It!

But this is just the beginning, yeah?