BUSHIDO BOTS

The Development of a Polished Online Multiplayer Experience in 2D

A Major Qualifying Project Report

submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

Submitted by

Wyatt Gray and Shane Daley

with assistance from

Jeffrey Thomas

and advised by

Professor Dean O’Donnell

Abstract

Bushido Bots is an IMGD MQP in which a team of three students concepted, developed, tested, and refined a two-dimensional, 12-player online game in the Unity3D engine. Our conceptual focus was developing a game that’s polish and entertainment value approaches that of professionally developed products. Each of the game’s art assets and animations was carefully hand-drawn by our artists. *Bushido Bots* blends familiar mechanics from popular games to deliver a unique yet easily approachable experience.

Acknowledgements

We would like to thank Dean O'Donnell, our project advisor, who helped us through the stressful and fragile game development process by offering techniques, input, and encouragement through seven months of development.

Keith Zizza for his valuable advice in the development of our robot voices.

Michael Grossfeld for recommending the ex2D software to expedite (and make possible) the development of our game using the Unity3D engine.

All of our playtesters, for their tremendously valuable feedback and encouragement.

And, of course, a tremendous thank you to Jeffrey Thomas, who graciously agreed to put himself through enormous hardship over three terms of Independent Study credit in order to program our technically ambitious game completely on his own. *Bushido Bots* simply would not exist without him.

Table of Contents

List of Figures

Part 1: Introduction

Overview

Bushido Bots is a 2D multiplayer action game, in which cartoon robots battle each other with medieval Japanese weaponry in an attempt to route the enemy forces and seize the area. Each player controls one robot, earning gold for their victories and investing it in eight weapons and armors which can be mixed and matched to produce 16 unique armor weapon pairings, each providing a unique gameplay style for players to experiment with.

Development

The game was developed over seven months, from October 2012 to April 2013, with two months of concepting and preparation preceding development. Wyatt Gray assumed the role of producer, designer, UI artist, environment artist, and sound designer. Shane Daley acted as character artist and designer. Having already completed an MQP, Jeffrey Thomas joined the team for Independent Study credit to act as our lead programmer, while also frequently contributing towards important design decisions.

Design

From the beginning, our team placed a tremendous emphasis on the design aspect of our game. We felt that although many MQP's were artistically or technologically impressive, they did not place enough emphasis on creating engaging gameplay, nor did they leave sufficient time to polish and fine-tune their final product. Our team's core focus was developing a polished game simple enough for a casual gamer to enjoy, yet deep enough for a gaming enthusiast to enjoy as well. With this in mind, we designed the game bottom up, choosing our art style, mechanics, and theming as to best create a game that would draw in players of all skill levels and hold their interest. We wanted *Bushido Bots* to resemble professionally developed games, and as such, strived to include as many features of a market-ready game as possible, such as a fully functional menu system, a stat-tracking scoreboard, a how-to-play section, an intuitive interface, and support for both keyboards and Xbox 360 controllers.

Art

The art of *Bushido Bots* is colorful and heavily stylized. This style is intended to capture the humorous, strange personality of our game. We made the unusual choice to develop a 2D game rather than a 3D game. All art assets were created using Adobe Flash and Adobe Photoshop, then transferred into Unity for its powerful networking capabilities. Our team set out to create assets with quality on par with that of successful indie titles such as Super Meat Boy, Skulls of the Shogun, and Castle Crashers.

Technology

Bushido Bots features stable, online multiplayer for up to 12 players, and is unusual in its use of 2D Flash art in the Unity3D engine. Networked gameplay is a tremendous challenge in and of itself-- creating a seamless experience for a fast-paced game enhances this challenge tremendously, and creating this experience for twelve simultaneous players increases that challenge even more. The Unity3D engine contains several features which greatly facilitate programming networked games. In order to overcome the challenge of creating 2D animations and assets, we used *ex2D*, a purchasable program used to assemble 2D animations within the

Unity engine. Thus, we were able to create a game with both advanced networked play and 2D animations.

Part 2: Gameplay

When initially designing *Bushido Bots*, we decided to incorporate some of our favorite mechanics from a variety of popular games. The intention was to create a game which would feel familiar enough to quickly pick up and play, yet unfamiliar enough to provide a unique experience worthy of the player's attention. Most gamers will recognize all of the mechanics used within *Bushido Bots* from games such as Castle Crashers, Team Fortress 2, and Counterstrike, all mixed together to create something new.

Gameplay Summary

In *Bushido Bots*, two teams of up to six robots battle over a Japanese fortress. Each player controls one of these robots, and is charged with mixing and matching armor and weapons for his robot warrior, and using him to engage and defeat enemy players' robots in order to claim the three 'control beacons' located across the map. Defeating enemies and controlling beacons will drain the enemy team's score to zero, winning the match for your team.

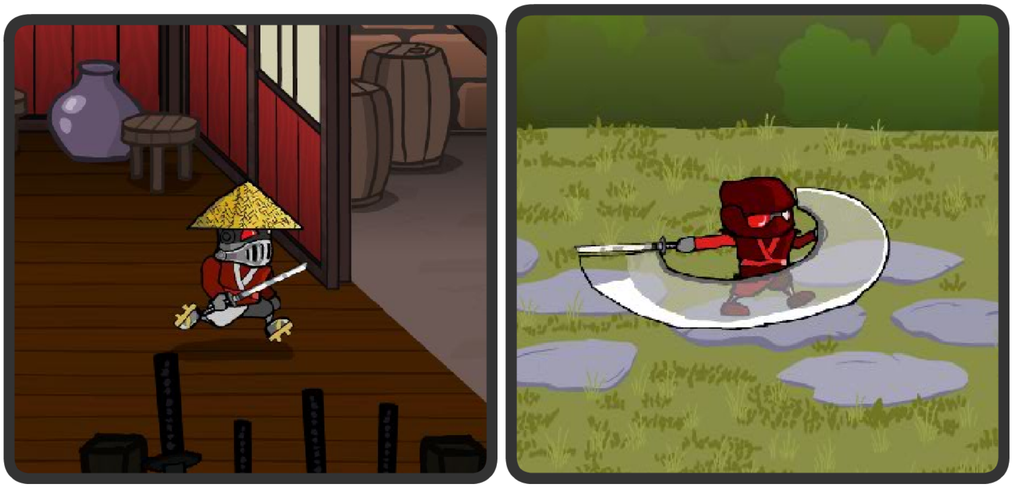
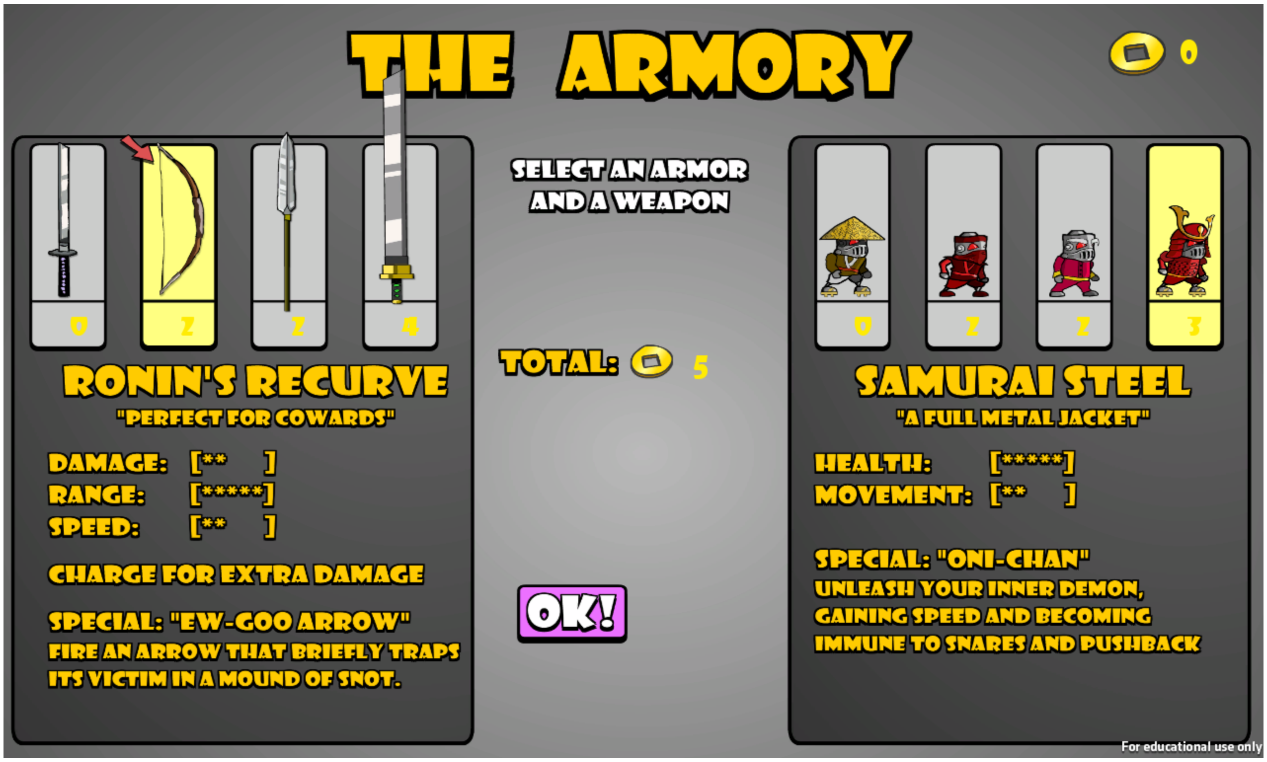


Fig 2.1: Left to right: A player's robot runs into combat. A robot performs a two-swing combo.

The Armory

Before the match starts, and each time the player's robot has been defeated, the player is brought to the 'Armory', which is essentially a shopping menu where players may spend their gold to purchase weapons and armors for their robot. Players start the game with 5 gold and may earn more by defeating enemies.



RONIN'S RECURVE

"PERFECT FOR COWARDS"

DAMAGE: [**]
RANGE: [*****]
SPEED: [**]

CHARGE FOR EXTRA DAMAGE

SPECIAL: "EW-GOO ARROW"
FIRE AN ARROW THAT BRIEFLY TRAPS ITS VICTIM IN A MOUND OF SNOT.

TOTAL: 5

SAMURAI STEEL

"A FULL METAL JACKET"

HEALTH: [*****]
MOVEMENT: [**]

SPECIAL: "ONI-CHAN"
UNLEASH YOUR INNER DEMON, GAINING SPEED AND BECOMING IMMUNE TO SNARES AND PUSHBACK

Fig 2.2: The armory screen, where players choose their weapon and armor before entering the match.

The Armory displays each of the game's weapons and armors. In the top right corner is the player's current supply of gold. The player must choose one weapon and one armor to assemble their robot. The cost of each item is displayed beneath it, with the total cost of the selected weapon and armor displayed in the center of the screen. Each weapon and armor comes with a detailed (and witty) description to help players make informed choices.

When a player has made an affordable weapon and armor selection, they may confirm their choice and respawn into the game in their game’s spawn room. Descriptions of each of the game’s eight weapons and armors can be found in *Appendix A: Weapons* and *Appendix B: Armors*.

Combat

Combat in *Bushido Bots* is designed to be fast-paced, replayable, and deep. Combat consists of attacking, maneuvering, blocking, and utilizing your defensive and offensive special abilities. These five factors come together to create a balanced and rich combat system that is easy to learn but difficult to master (see Sample Tactics on page for elaboration). Players are rewarded for playing carefully and using each aspect of combat when appropriate. They are also rewarded for working together, as their character’s strengths will allow them to compensate for each other's weaknesses.

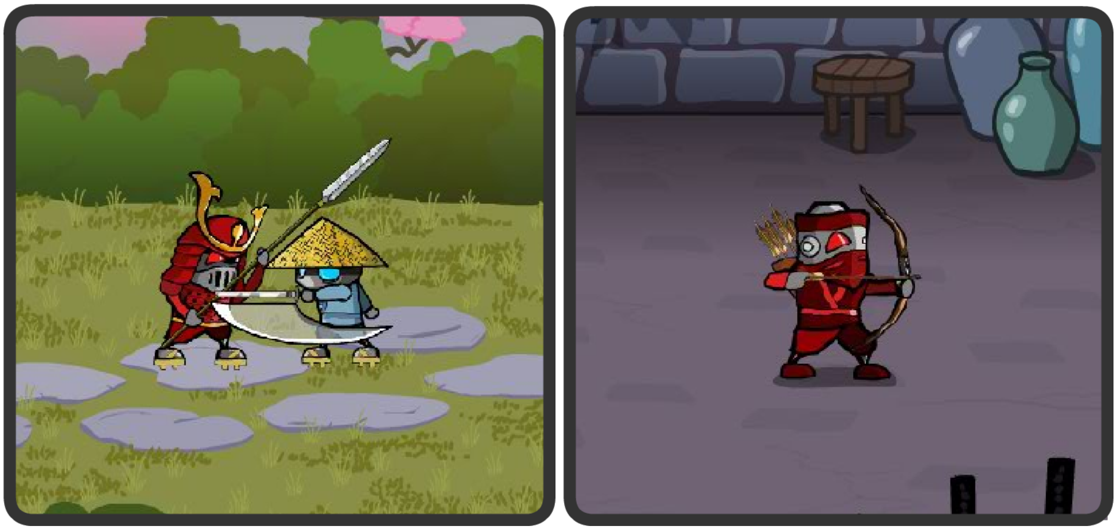


Fig 2.3: Left to right: Blocking with a spear throws enemies off-balance. Charging up the bow for a stronger shot.

Teamwork and coordination are heavily rewarded in *Bushido Bots*. The game is designed to encourage players to create a balanced team in order to overcome enemy tactics. If a team is overspecialized, they may find themselves lacking a critically needed role. For example, a team without a sensei will find themselves unable to sustain their lives, as their team lacks a method of recovering health. A team without archers or ninjas will be forced to charge enemy bowmen through a hail of painful arrows. A team without enough swordsmen will find themselves vulnerable to ninjas ambushing their archers and healers. A team without enough samurai and spearmen will have trouble protecting their beacons. Each weapon and armor provides its own strengths and weaknesses, and thus coordination and teamwork often determine the victors.

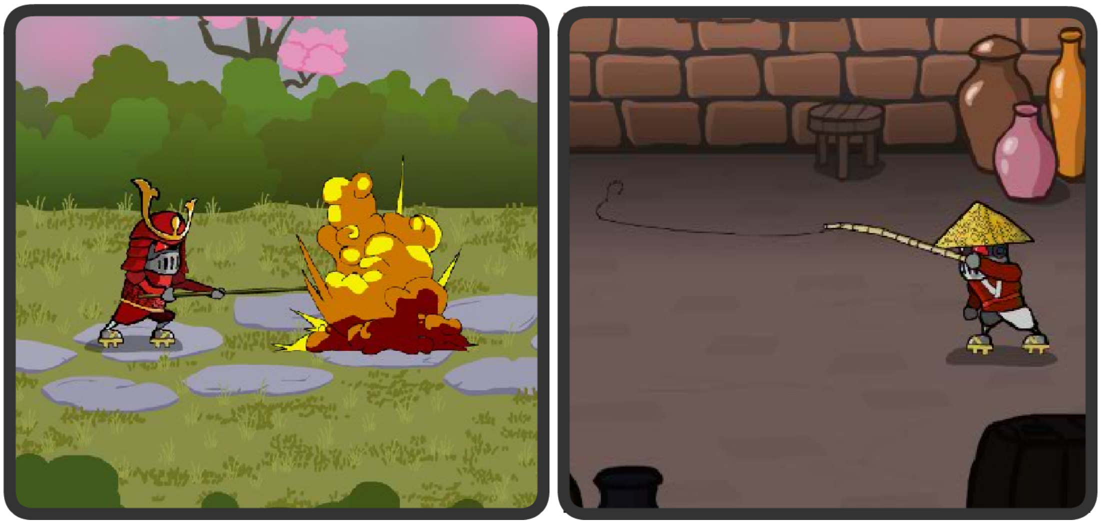
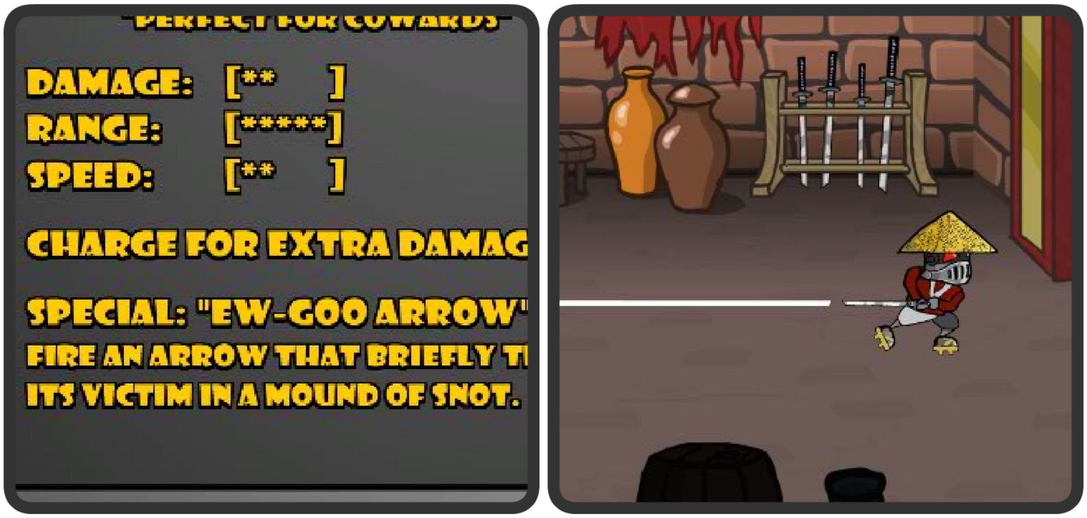


Fig 2.4: A samurai uses the spear to finish off an enemy. A peasant uses his 'goldfish' ability to steal some gold.

Furthermore, many of the game's special abilities are of low value to the player but high value to the team. Spears have the ability to push away dangerous enemies or pull vulnerable ones, putting the user's team in an advantageous position. Bows can fire a goo-arrow which traps its target, making him short work for allied swordsmen. The sensei's special ability heals himself, but more importantly, heals all nearby teammates as well, allowing him to sustain his allies through numerous engagements. We wanted players to have the opportunity to provide value to their teams, and so we implemented these supportive functions into the game.



"PERFECT FOR COWARDS"

DAMAGE: [**]
RANGE: [*****]
SPEED: [**]

CHARGE FOR EXTRA DAMAG

SPECIAL: "EW-GOO ARROW"
FIRE AN ARROW THAT BRIEFLY T
ITS VICTIM IN A MOUND OF SNOT.

Fig 2.5: Closeup on a weapon's special ability description. A katana wielder uses 'blade rush' to charge forward.

Winning the Game

Bushido Bots' sole game mode can be briefly described as "Three-point King of the Hill". To win a match of *Bushido Bots*, a team must reduce the enemy score to zero. Each team starts with 200 points. Teams cannot gain points, but instead reduce the enemy's points by defeating its players and controlling more capture points at a time. The greater the difference in capture points owned, the faster the other team's points decrease. There are three capture points: one close to the red team's spawn point, one close to the blue team's spawn point, and one directly in the middle. The placement was intended to make the game's most decisive battles take place in the middle, and to make it difficult for a team to capture all three control points. Each point takes fifteen seconds to capture, divided by the number of allied players standing on the point. If players from both teams are standing on the point, it cannot be influenced.

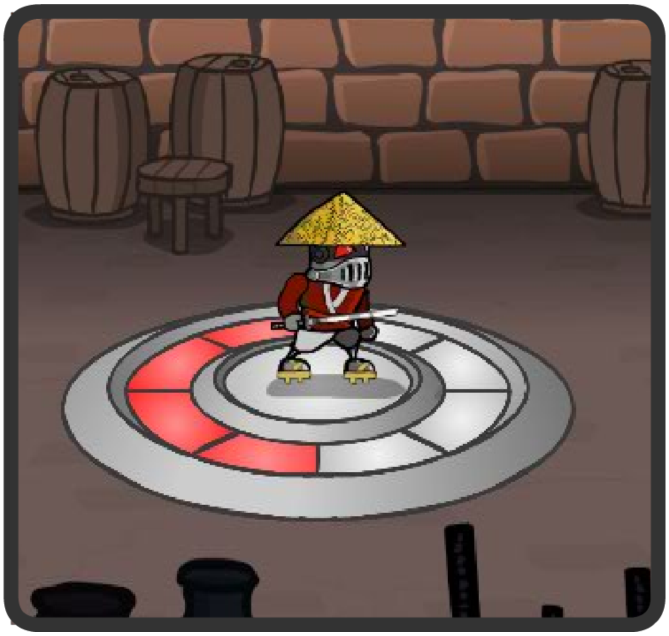


Fig 2.6: A red player stands on the beacon, capturing for his team. Its panels change color to show his progress.

Originally, we considered imposing a time limit on the game so that whoever had the most points when the time was up would win. Initially, we believed this would keep games from lasting too long. However, we soon realized this was unnecessary since the beacons inevitably drain the scores to zero. By adjusting the rate at which beacon control drains the enemy score, we were able to put an approximate time limit on the game without having to implement an actual timer. On average, a match of *Bushido Bots* lasts approximately eight minutes, which is based on the 150-point initial scores, which is up from our original score of 100, which led to five-minute matches. In discussion with our testers, they commented that 150-point matches felt long enough to have several battles but not so long that losing was painfully drawn out. As such, we kept the score at 150.

We also carefully considered whether to have the the score count up from zero or count down to zero. Ultimately, we decided that the score ticking down towards zero was more exciting, as it enhanced the idea that the game is ‘ticking down towards conclusion’.

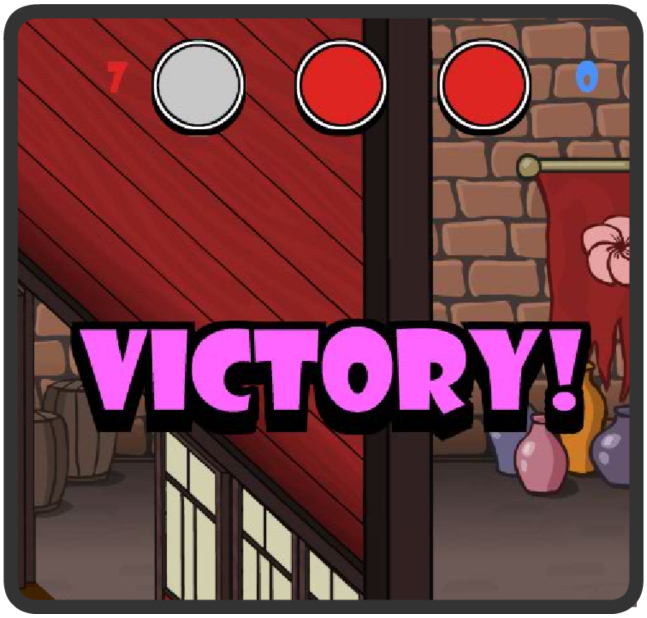


Fig 2.7: Winning the match is accompanied with a victory message, and a short tune.

The game’s “three-point king of the hill” mode was not the only game mode we had in mind. We strongly considered developing a game mode called ‘Siege’ where an attacking team was tasked with breaking into the defending team’s fortress, destroying power generators to lower barriers, and eventually destroying the defending team’s idol. As much as we liked the idea, we decided that implementing and balancing such a mode would be outside the scope of development.

The Level

Bushido Bots features a symmetrical level that contains two bases--one for each team-- and a field in the middle. The bases consist of two rooms, a clean and neat spawn room and a brick and mortar armory. Each room is filled with weapons, vases, and barrels, which are purely cosmetic. The threshold between each area serves as a choke point. Since the three central areas of the map contain a capture point, having to cross through a choke point to enter the spawn room provides an advantage to the defender waiting on the other side.

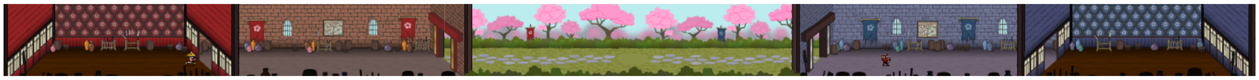


Fig 2.8: The entire level. From left to right: red spawn room, red barracks, field, blue barracks, blue spawn room.

Since they do not contain capture points and are not intended for combat, they are compact. The armories and field are the large to give enough space for combat to take place unimpeded, since the majority of confrontations are intended to take place in these areas. Additionally, the length of the map acts as a rubber-banding feature as the dominant team must travel farther to reach the action after dying, since have pushed farther away from their spawn room. This reduces their momentum allowing the losing team to more easily turn their luck around.

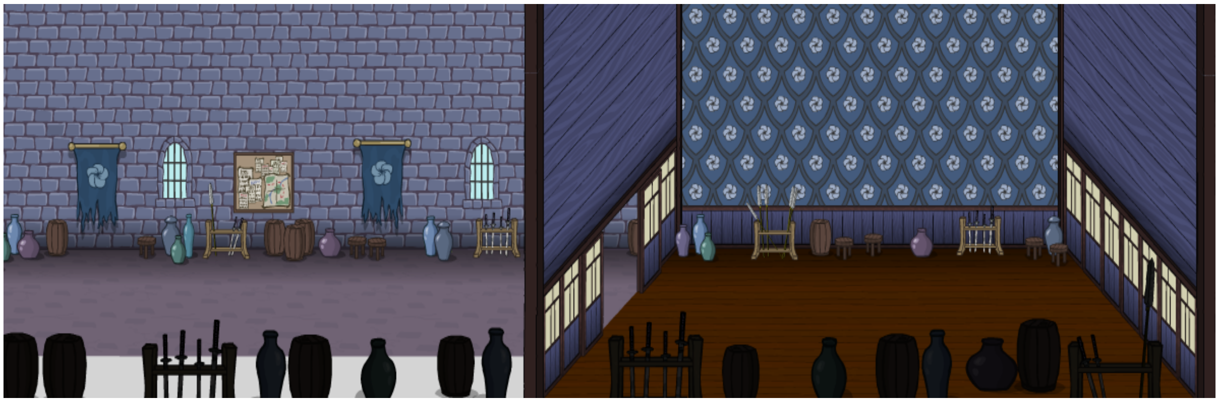


Fig 2.9: Closeup of the blue barracks and spawn room. Props are placed at top and bottom to avoid obstruction.

During pre-production, the team still had not decided on a setting and layout for the game's level. Several ideas were proposed: The ruinous streets of a post-apocalyptic Tokyo, a quiet fishing village in the forest, the grounds of a huge shrine, and many others. Ultimately, Wyatt decided to develop the ‘feudal fortresses’ theme, as he felt that it best reinforced the idea that the robots thought they were soldiers in feudal Japan. He also liked that it would afford him the opportunity to create distinct and colorful environments. Although the team initially planned to make more levels if they had the time, it was later decided that focusing all efforts on creating one professional quality environment was a smarter use of time.

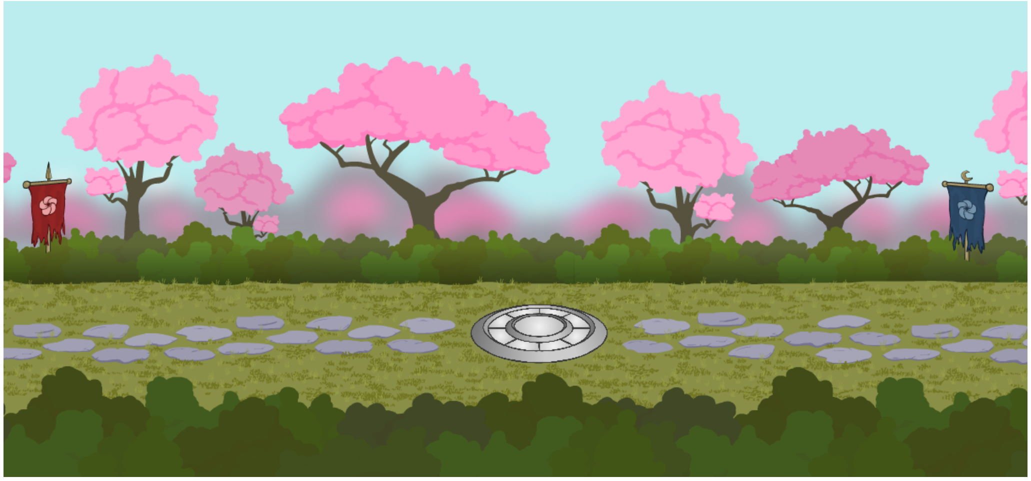


Fig 2.10: The map’s central beacon in the field acts as a tie-breaker, forcing both teams to leave their bases.

Design Statement

Our core design statement for *Bushido Bots* was “Is it fun?”. Jesse Schell, a professor of game design at Carnegie Mellon, succinctly defines fun as “pleasure with surprises” in his book, *The Art of Game Design*[1]. With this in mind, we set out to create a game which was both pleasurable and would contain enough surprises to hold our players’ interest for a long time: as soon as the surprise runs out, the game stops being fun, and simply becomes pleasurable.

In general, there are two types of surprises: content-based surprises are those that are ‘baked into the game’, like an enemy jumping out of a bush. Alternatively, there are emergent surprises,

[1] Schell, Jesse. *The Art of Game Design: A Book of Lenses.* Burlington: Elsevier, 2008.

which is surprise that emerges from the unpredictability of the game. Randomness and multiplayer are two common methods of creating emergent surprises.

As such, we chose to make the game multiplayer. Here are a few consequences of this choice:

- The surprise and challenge of a competitive multiplayer game is player-driven, rather than content driven. This means that instead of designing a lot of *content* (maps, weapons, enemies, stories) for the player to experience, the content is generated by the players as they behave uniquely, generating a varied experience infinitely beyond the capacity of a computer. No amount of programming has even come close to the complexity of the human mind, and as such, our players are our most powerful resource for generating a continually enjoyable experience.

- Each member of the team is a fan of competitive multiplayer gameplay, and as such, featuring this element in our game further increased our excitement and resolve to develop the game.

- Since the content of a competitive multiplayer game can be scaled down, the quality of content must be higher, since the players will constantly engage this same content when playing the game. For example, in a game with a lot of content, the player may avoid or quickly pass through an unenjoyable portion of content and continue enjoying the game's other content. Since our game has limited content, it was important that this content be highly appealing to all players and for a longer period of time.

Replayability

In order to create a replayable game, we sought to create *emergent gameplay:* “Complex situations in games that emerge from the interaction of relatively simple game mechanics.”[2] In order to achieve emergent gameplay, our designers sought to keep gameplay basic enough to learn in a couple minutes, but offer enough entertainment value to hold players’ interest for multiple playthroughs.

In many competitive games, players must memorize numerous weapons, armors, items, abilities, maps, locations, strategies, and terms. This creates a steep learning curve which punishes players as they attempt to familiarize themselves with the game. The core of our emergent gameplay is a small number of basic weapons and armors that can be mixed and matched to create sixteen unique and viable combinations.

Furthermore, the mere inclusion of competitive multiplayer adds tremendous emergent gameplay to the game. Every human player will play the game slightly differently every single time. As such, playing against other humans will literally *always* provide a unique experience. For this reason, ancient games such as Go and Chess are still commonly played today, sometimes thousands of times over the course of one’s life (in the case of professionals).

Each player may choose from four weapons and four armors. This means that there are sixteen possible weapon/armor combinations for the player. The differences among the weapons are all rather basic and obvious (katana is fast and weak, spear is long and slow), allowing players to immediately understand their purpose. These simple differences, however, have dramatic

[2] Schell, 141.

effects on one’s playstyle.

Furthermore, since the players must combine a weapon with an armor, each of these playstyles is further augmented. Again, the differences among the armors are obvious (peasant is expendable, samurai is tough, ninja is fast and sneaky, sensei can heal), and these simple differences have dramatic effects on the player’s options.

With each weapon and armor playing quite differently from the rest, these sixteen combinations each provide a unique experience for the player. Getting bored using a peasant with a spear? Try playing as a ninja with that spear instead and you’ll need to seriously adjust your strategy. Or swap that spear out for a bow. Again, a completely different approach is needed.

One issue that often emerges in competitive games is the discovery of a dominant strategy. However, we feel that we have alleviated the possibility of a dominant strategy by building a light rock-paper-scissors system into the game to punish over-specialization. For example, ninjas are particularly good against archers. Samurai, however, beat ninjas handily. Meanwhile, archers can outrun Samurai. In this sense, the only dominant strategy is to build a balanced team to avoid being picked apart by your chosen class’s weakness.

Sample Tactics

Bushido Bots is designed to support emergent gameplay, and as such, we are sure that our players will discover numerous effective strategies towards defeating their opponents. Outlined below, however, are a few basic strategies to help players understand the kinds of tactics which might guide a player’s actions in combat.

Katana vs. Bow

New players quickly discover a small yet important facet of combat: well-timed melee attacks cut down incoming arrows. With this in mind, the katana is best suited for slicing up arrows and closing in on enemy archers. Unfortunately, a short weapon range means it’ll take the longest to close the gap. This space can be travelled through the katana’s “Blade Rush” ability, which can jump forward and dodge arrows. Once the gap is closed, the katana wielder will make short work of an enemy archer, if his armor is light enough that he can keep up.

A bow user must decide whether he thinks he can land his arrows or not-- if not, he should run away. The bow user should also consider using his goo-arrow to trap the incoming katana wielder so that he may escape, or to await team support.

Spear vs. Dai-Katana

The spear user will spend this fight attempting to keep just out of the other player’s range, as the spear is slightly faster and has slightly longer range than the dai-katana. The spear user will be forced to score more hits, but this is facilitated by the spear’s pushback passive ability. If the dai-katana wielder begins to retreat, a spear user can pull him in for the finishing blow.

A dai-katana user must use his weapon’s extreme damage to surprise and quickly overwhelm opponents. He need not worry about blocking or being blocked, as the dai-katana is too big to defend himself with and attacks go straight through the opponents’ parries. To avoid the spear’s pushback the dai-katana user can use the weapon’s special ability to jump over attacks and slam his opponent for massive damage.

Samurai Armor - Tough but slow

A player that chooses to don the Samurai Steel excels at surviving. Even with a weaker weapon, a samurai has the clear advantage in a straight-up fight with a less-armored opponent. Unfortunately, the armor's slow move speed means that opponents can more safely attack from a range and harass with hit-and-run tactics. A samurai will be hard-pressed to catch a fleeing ninja, but a ninja will be hard pressed to defeat a samurai by himself. However, the samurai can use his armor's special ability to briefly run faster, become immune to the archer's goo arrow, ignore the push and pull effects of the spear, and deny even the most precisely-timed parries.

Ninja Armor - Fast but fragile

No one is safe from the stealthy ninja. The ninja armor allows the fastest movement of the armors available to the player, but it offers the least protection. A ninja will almost never have difficulty catching an opponent, but has to be wary of being overpowered once he begins his attack. Fortunately, the ninja can become completely invisible to enemy players to gain a useful tactical advantage. A few arrows can spell the defeat of an overly-brazen ninja, but an enemy bowman is at the mercy of an up-close-and-personal sneak attack from an unseen wearer of the ninja outfit. Ninjas excel at getting the jump on opponents, dealing moderate damage, and escaping to surprise opponents later.

Opportunities for Teamwork

Bushido Bots is designed to reward teamwork and cohesiveness. By using abilities from a diverse set of weapons and armors, players may compensate for each others weaknesses. A bow wielder can deal significant damage and force an engagement, but without a frontline of melee defenders, he is incredibly vulnerable. Enemies can only block in one direction, so a clever team will flank their opponents to catch them off-guard - a perfect job for a ninja. Meanwhile, the samurai's large health pool and special ability make him perfect for dashing in first and taking all of the blows. A sensei is vital for his valuable healing ability. Although an enemy might easily defeat two of your allies if they are separated, they will almost assuredly win if they stick together. These kinds of mechanics and situations all encourage team-based gameplay, which we feel provides a more memorable and enjoyable experience, and rewards communication (via Skype, TeamSpeak, or proximity) and higher-level play.

Setting and Story

It is the Year 2200. Humankind has inevitably driven itself to extinction through the escalation biological warfare. In their wake, only robots remained to rebuild society. However, due to a tragic flaw in their programming, these robots have chosen to use the traditional arms and armor of their creators.

Tensions are high. The mighty Robo-Shogun has declared war on the Emperor, and Robo-Japan has descended into chaos. Now, the Royal RedBots and Blazing BlueBots clash in the once-serene fields of Japan, seeking to bring great honor upon their masters.

Developing the Setting and Story

The team agreed early on that story and setting would take a backseat to gameplay in *Bushido Bots*, for a few reasons:

First, we wanted the player to get from booting the game up to playing as quickly as possible. That meant not including any story in the game. If we'd had the time, we would have liked to include an opening cinematic outlining the story and setting to provide players with a clearer context for the gameplay.

Since our game was designed to be about emergent, rather than premade story, plot held little relevance to our game. Instead, we decided to focus on world-building. The peculiar 'robots in the future fighting like its the past' setting was intended to amuse our players and capture their imagination. What would the Robo-Shogun look like? Why did he declare war on the Robo-Emperor? By leaving the plot light, we encourage the players to fill in the gaps using their own imagination.

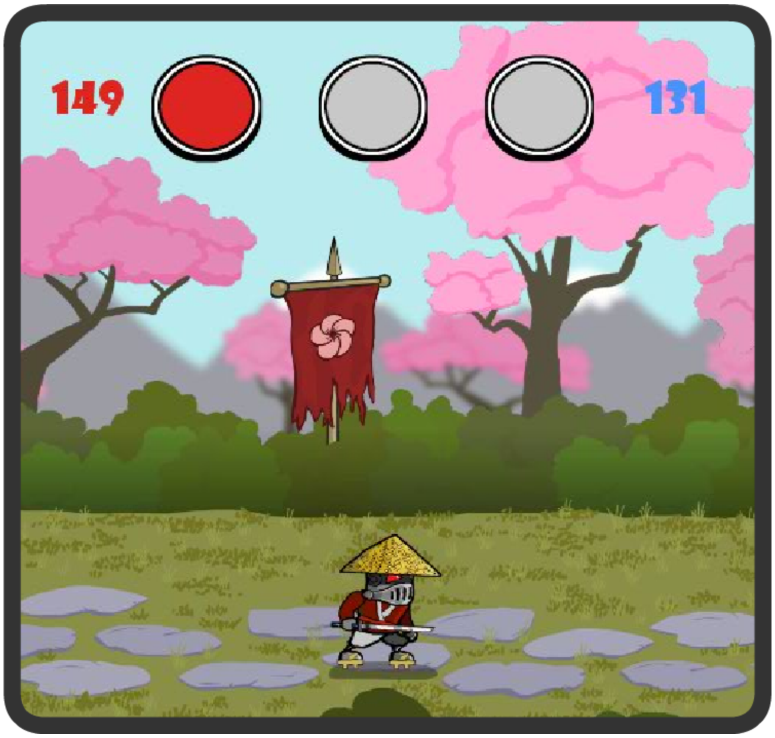


Fig 2.11: The field. Japanese Cherry Blossoms reinforce setting. Red banner indicates that the red base is nearby.

Ultimately, we chose a Japanese setting because we felt it would provide a compelling selection of familiar weapons and armors for players to use (ninja, samurai, katana, bow, etc.), as well as complement the quirky and campy nature of our game.

The robot theme was introduced after we switched the game from 8-bit to high-res in pre-production. The team agreed that we didn't want the game to be bloody, and quickly brainstormed that we could use robots instead of humans, as they could spray oil and sparks when struck, and simply blow up when 'killed' instead of getting cut down and leaving a corpse on the ground. Everyone agreed this was an elegant and interesting solution, and would make our game stand out a bit more thematically to boot, and so it was settled.

By putting a heavy emphasis on the game's theming, we were able to create a more enjoyable experience for the player. Everything in the game is designed to either be cartoony, robotic, or feudal Japanese (and all three of them whenever possible). This makes the game feel cohesive rather than patched-together, resulting in a more engaging and immersive world.

Controls

We decided to make the game playable on either keyboard or wired Xbox controller (plugged into a USB port before starting up the game). There are a few reasons we decided to allow both:

The keyboard route was necessary for testing the game conveniently on PC, as well as allowing those without wired Xbox controllers (a vast majority of our players) to play the game.

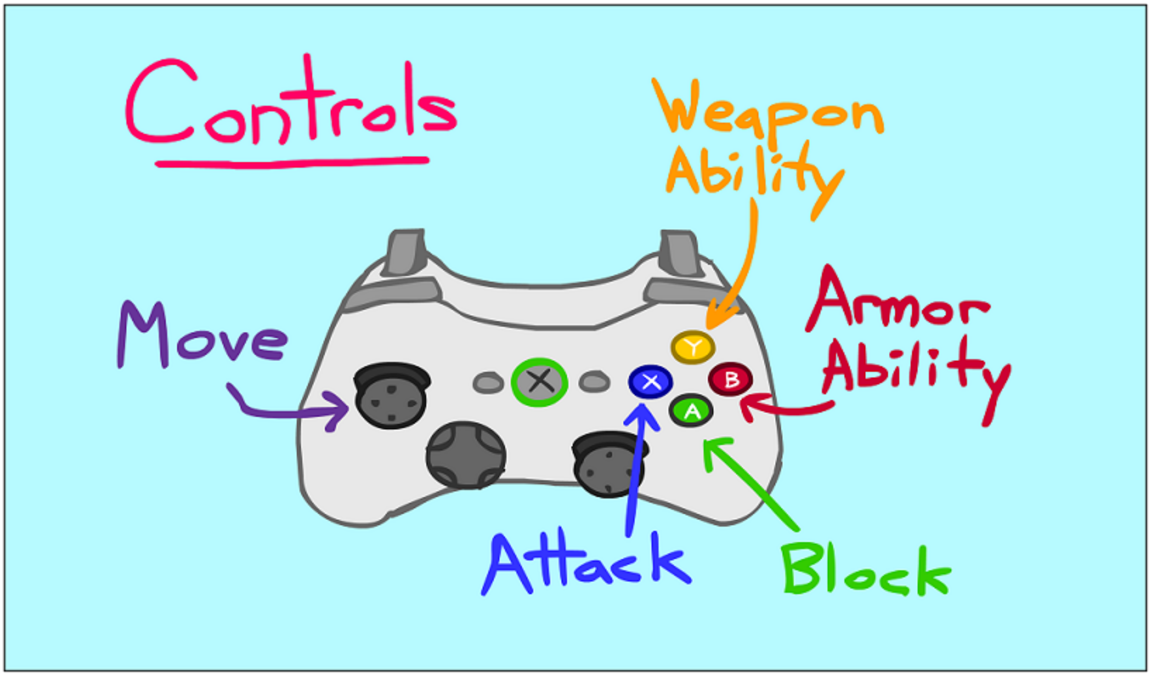


Fig 2.12: A diagram of the button layout for the Xbox 360 controller. Used as a loading screen.

Unfortunately, the gameplay and genre strongly lend themselves to being played on a controller and NOT a keyboard, so we went ahead and left the option open to play on an Xbox controller if the player desires. It's much more comfortable than the keyboard, but frequently just not an option.

Initially, we planned to develop the game on PC, and then release it on Xbox Live Arcade (XBLA). Our reasoning was that the game had a certain 'indie console game' art style and overall feel. More importantly, the gameplay strongly lent itself to being played using a console controller. Unfortunately, the idea of porting the game onto XBLA proved to be outside the scope of our development time, as the logistics of porting the game to XBLA would have added numerous hours of additional tech work for Jeff-- hours which simply were not available.

Part 3: Artistic Vision

Building an Artistic Vision

Originally, we envisioned the game as having ‘8-bit graphics’ (as shown below). Back then, the game was tentatively named *Bushido Blade Rush*. We thought that low-res graphics would reinforce the retro feel we wanted the game to have, and would free up a lot of our time from art to focus on gameplay iteration and adding more content.

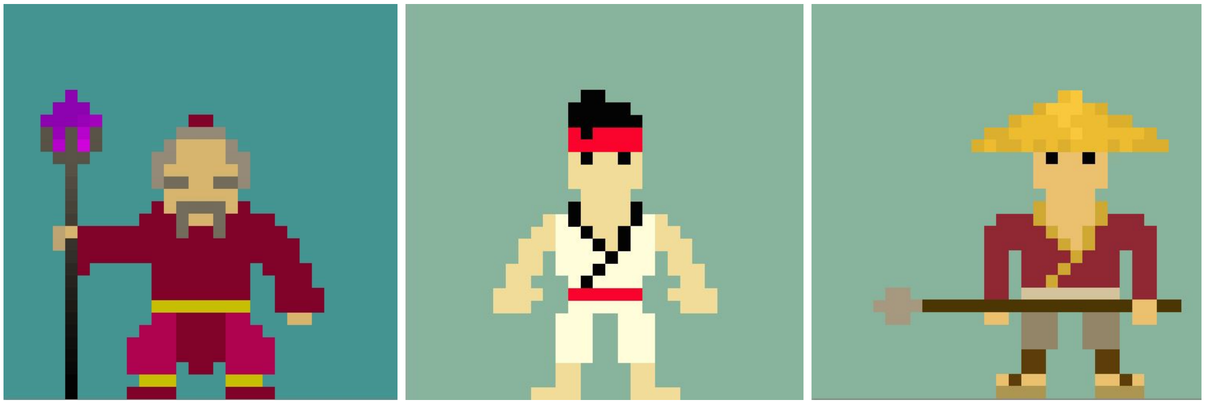


Figure 3.1: Pixel art created pre-production. From left to right: The sorceror, the brawler, and the peasant.

However, after spending about a week developing 8-bit art assets in pre-production, we realized that the art style did not lend itself at all to the gameplay we had envisioned:
Making animations that would convey the information that needed to be conveyed seemed impossible-- the characters needed to play fluid animations to allow other players to be able to react and ‘block’ incoming attacks. 8-bit graphics do not lend themselves to animation well.

It was difficult to capture the various armors and weapons using 8-bit graphics. Furthermore, displaying these weapons and armors and their animations clearly in all four directions (toward the camera, away from the camera, left-facing, right-facing) did not seem like something we could effectively do.

And so we scrapped the 8-bit art style and decided we would draw high-resolution 2D art assets ‘cartoon-style’ using digital art programs. Initially, Shane concepted the game art in Adobe Photoshop, his 2D art program of choice. However, it became apparent that Photoshop did not lend itself to 2D animation. As such, the artists decided to develop all art assets for in-game use using Adobe Flash. Although this meant that Shane had to spend numerous hours learning the new software, it ultimately saved him many hours of redrawing and rebuilding the countless animation frames he created for the game. It also worked out well for Wyatt, who already had significant experience using Flash, which is his preferred 2D art program.

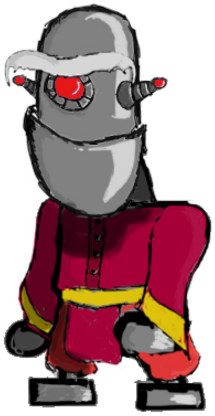


Figure 3.2: Concept art drawn in Photoshop for the sensei outfit. This head did not make it into the final product.

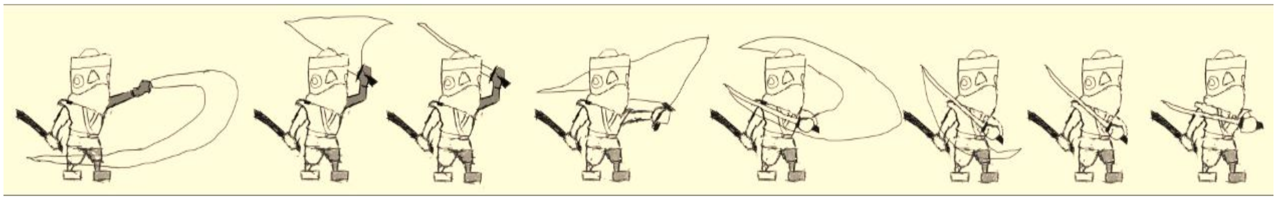


Figure 3.3: The katana's attack animation being prototyped in Photoshop. Later, prototyping was done directly in Flash.

Artistically, the game was heavily influenced by 'Newgrounds-style' Flash art, which could be described as any art that appears to have been made in Adobe Flash with use of the Flash Paint Brush, a unique art tool which provides drawings with a distinct look (as shown below).

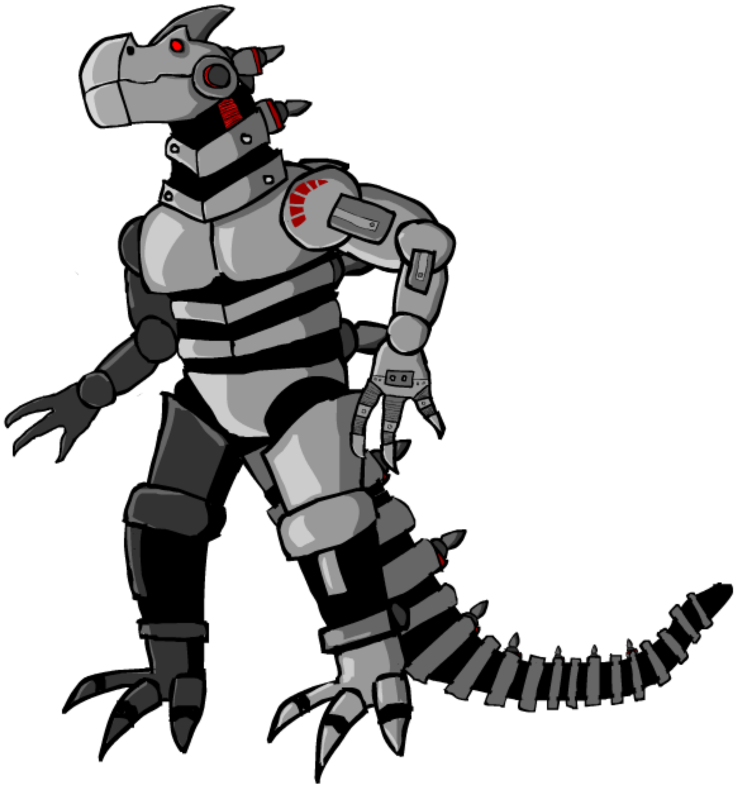


Figure 3.4: An art asset created quickly in Flash, demonstrating Flash's propensity for a unique and quality style.

The team decided early on that cohesive art was essential to the visual flavor of *Bushido Bots*. The Flash paint brush made this notably easier, as it unified all of the art assets even as two different artists worked and grew in skill. For example, we believe that early finalized art created by Shane looks like it belongs with art created by Wyatt that made it into the game much later into production.

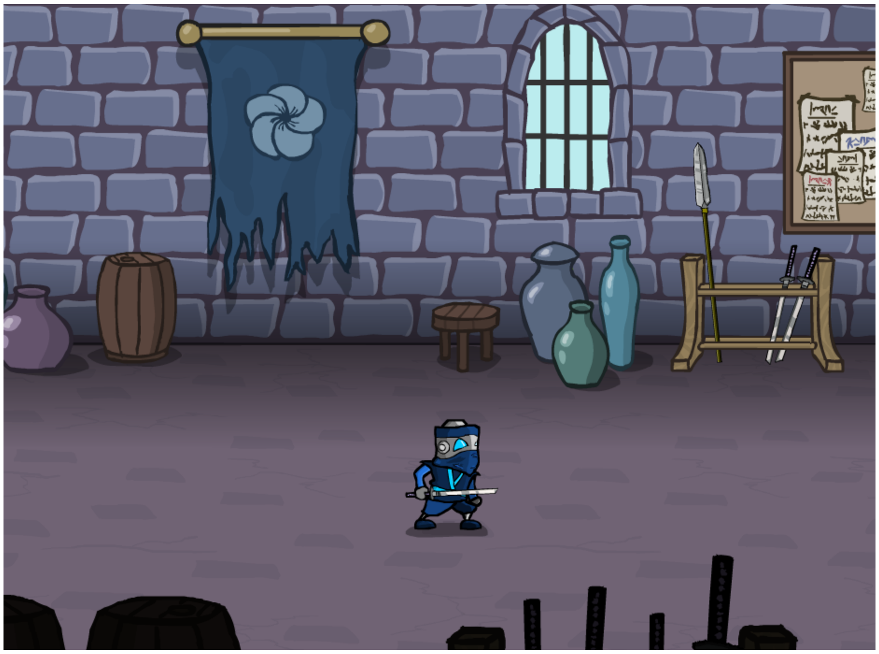


Figure 3.5: A katana-wielding ninja in the blue barracks. The environment was not finished until late development.

Some of the art team's favorite Flash artists and biggest inspirations include Dan Paladin, the lead artist for Behemoth Games (*Castle Crashers, Battleblock Theater*), as well as Ed McMillen, lead artist for Team Meat (*Super Meat Boy, The Binding of Isaac*). Although we were not aware of it until we were well into development, our game is also visually similar to *Skulls of the Shogun*, by 17-BIT.

Art Pipeline

The process of figuring out the pipeline for our game was one of the main technical challenges of development. While uploading environment art and UI elements proved to be as simple as we had imagined, the process of playing 2D animations in Unity turned out to be one of the largest obstacles we would face. Over the course of development, our art pipeline vastly improved, significantly cutting down the time it took to create, upload, and organize animations.

In the early stages of development, we believed it would be relatively simple to transfer animations from Flash to Unity. We had the misconception that there would be a simple and convenient way to take a finished Flash animation, upload it into Unity, and then have the animation play as a movie clip on a 2D plane to simulate a character.

Unfortunately, this proved to be incorrect. After a couple weeks of research and searching for a simple solution, the team came up short. However, while discussing the issue in one of the campus computer labs, a friend and fellow developer named Michael Grossfeld suggested that the team investigate a tool called *ex2D*, which was designed to assemble 2D sprite sheets into playable animations in Unity.

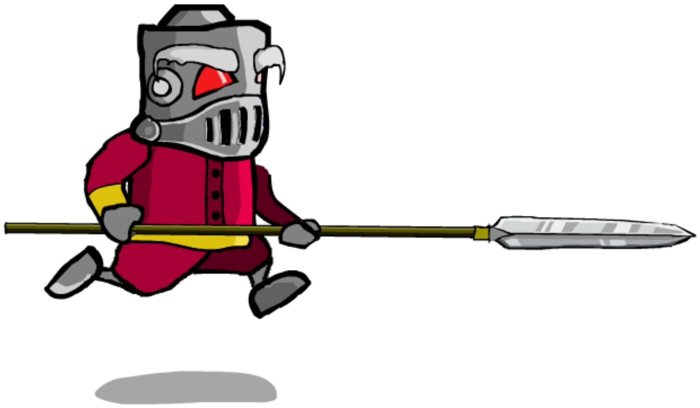


Figure 3.6: A still of a finalized animation (red sensei running with the spear).

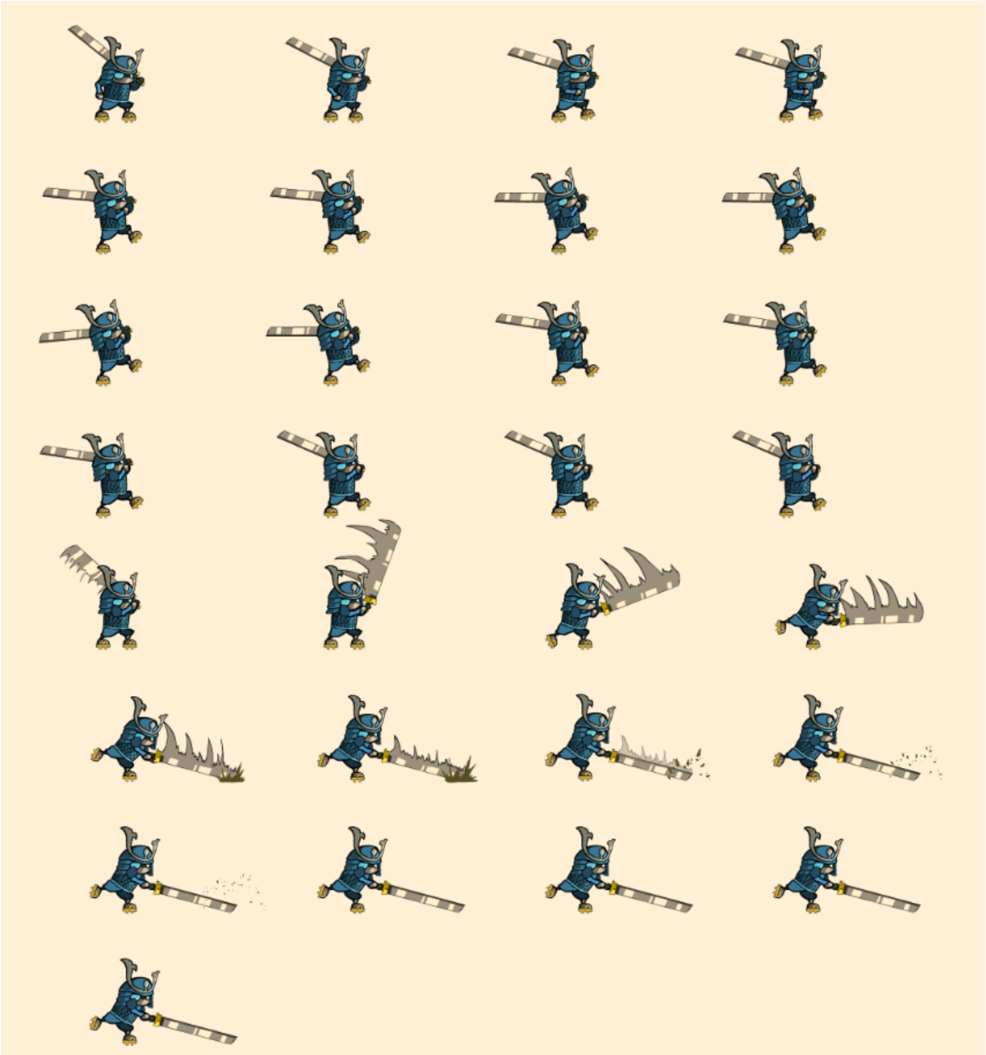


Figure 3.7: A sprite sheet representing the progression of an animation.

Jeff investigated the tool and determined that it would solve the pipeline issue. Wyatt purchased the tool and the team began the process of adapting to the tool. Initially, the pipeline in conjunction with ex2D was incredibly disorganized and unclear.

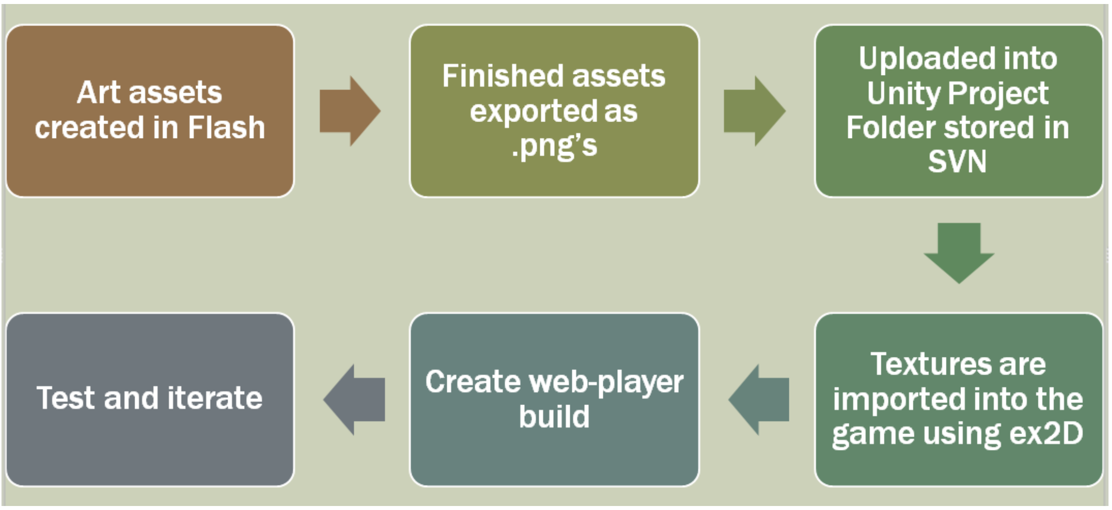
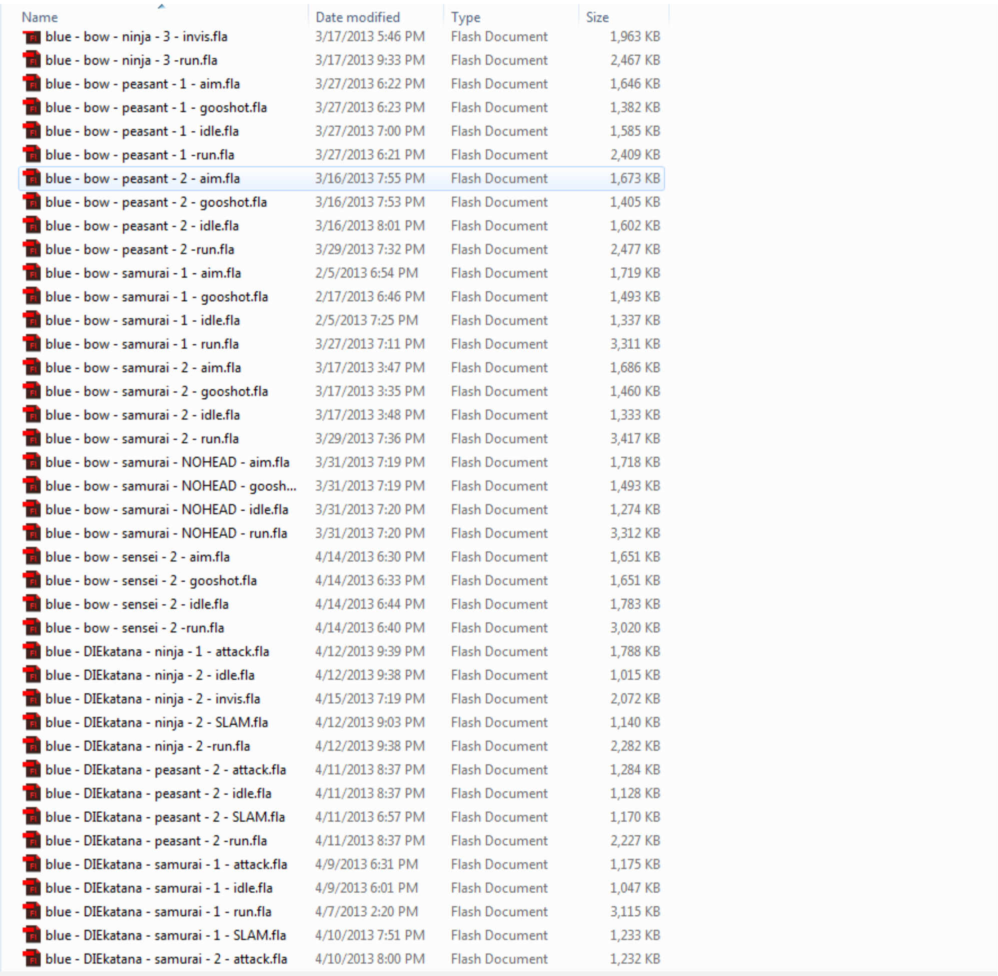

Art assets created in Flash
Finished assets exported as .png's
Uploaded into Unity Project Folder stored in SVN
Textures are imported into the game using ex2D
Create web-player build
Test and iterate

Figure 3.8: A flowchart of our pipeline.

There were a number of hurdles to be cleared to figure out the most efficient way to get animated art assets in game. Early on, even after it was discovered that Flash animations could be exported as a series of .png's (one separate picture per frame in each animation) there were problems with the dots per inch (dpi) of the images. A higher dpi meant more memory usage and different dpi's meant that assets came into Unity at different sizes. There was no efficient way to scale images after they were exported from Flash, so it was determined that every animation had to be exported at the same dpi and that the dpi had to be low, without greatly affecting the visual quality of the images.

With this limitation in place already in place, positioning of the characters within Flash turned out to be an issue that cost many hours to mitigate, even far into production. We determined that every Flash document had to share the same canvas size and every character in each separate document had to be positioned in the same exact spot. We later discovered that the canvas size did not actually have to be the same size, but we had already made the mistake of settling on a 700x400 size for animations that were not nearly that large. Unfortunately, this exacerbated later memory problems.

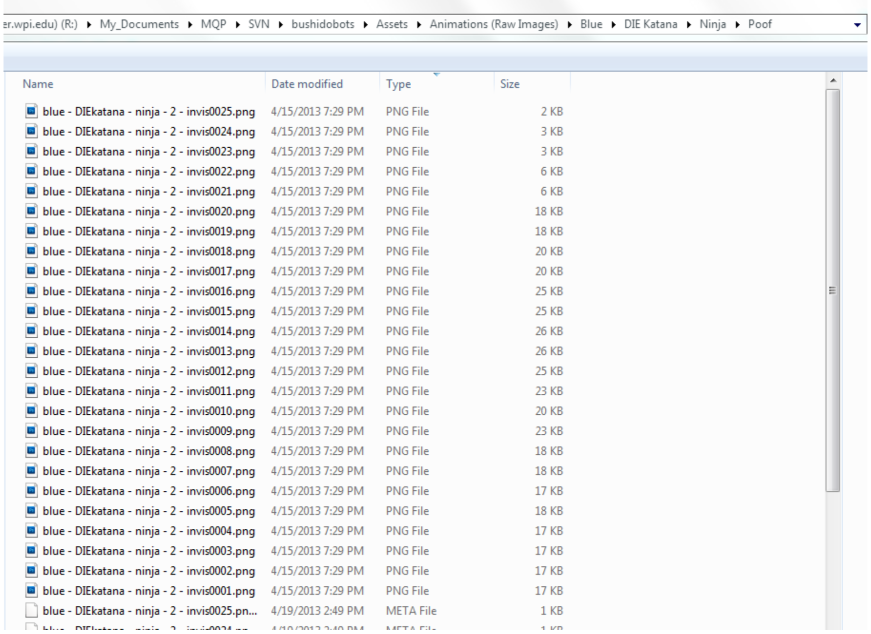
Our animations had to be impeccably organized, as there are hundreds of distinct but related animations (over 400) and thus thousands of separate frames (over 10,000) to be saved as pictures we needed an easily understandable naming convention. Since each animation could be described as a combination of color, weapon, outfit, head type, and action we settled on ‘color - weapon - outfit - head type - action.’ An example of this naming convention in Flash file form is “blue - diekatana - ninja - 2 - invis.fla.”



Name	Date modified	Type	Size
blue - bow - ninja - 3 - invis.fla	3/17/2013 5:46 PM	Flash Document	1,963 KB
blue - bow - ninja - 3 -run.fla	3/17/2013 9:33 PM	Flash Document	2,467 KB
blue - bow - peasant - 1 - aim.fla	3/27/2013 6:22 PM	Flash Document	1,646 KB
blue - bow - peasant - 1 - gooshot.fla	3/27/2013 6:23 PM	Flash Document	1,382 KB
blue - bow - peasant - 1 - idle.fla	3/27/2013 7:00 PM	Flash Document	1,585 KB
blue - bow - peasant - 1 -run.fla	3/27/2013 6:21 PM	Flash Document	2,409 KB
blue - bow - peasant - 2 - aim.fla	3/16/2013 7:55 PM	Flash Document	1,673 KB
blue - bow - peasant - 2 - gooshot.fla	3/16/2013 7:53 PM	Flash Document	1,405 KB
blue - bow - peasant - 2 - idle.fla	3/16/2013 8:01 PM	Flash Document	1,602 KB
blue - bow - peasant - 2 -run.fla	3/29/2013 7:32 PM	Flash Document	2,477 KB
blue - bow - samurai - 1 - aim.fla	2/5/2013 6:54 PM	Flash Document	1,719 KB
blue - bow - samurai - 1 - gooshot.fla	2/17/2013 6:46 PM	Flash Document	1,493 KB
blue - bow - samurai - 1 - idle.fla	2/5/2013 7:25 PM	Flash Document	1,337 KB
blue - bow - samurai - 1 - run.fla	3/27/2013 7:11 PM	Flash Document	3,311 KB
blue - bow - samurai - 2 - aim.fla	3/17/2013 3:47 PM	Flash Document	1,686 KB
blue - bow - samurai - 2 - gooshot.fla	3/17/2013 3:35 PM	Flash Document	1,460 KB
blue - bow - samurai - 2 - idle.fla	3/17/2013 3:48 PM	Flash Document	1,333 KB
blue - bow - samurai - 2 - run.fla	3/29/2013 7:36 PM	Flash Document	3,417 KB
blue - bow - samurai - NOHEAD - aim.fla	3/31/2013 7:19 PM	Flash Document	1,718 KB
blue - bow - samurai - NOHEAD - goosh...	3/31/2013 7:19 PM	Flash Document	1,493 KB
blue - bow - samurai - NOHEAD - idle.fla	3/31/2013 7:20 PM	Flash Document	1,274 KB
blue - bow - samurai - NOHEAD - run.fla	3/31/2013 7:20 PM	Flash Document	3,312 KB
blue - bow - sensei - 2 - aim.fla	4/14/2013 6:30 PM	Flash Document	1,651 KB
blue - bow - sensei - 2 - gooshot.fla	4/14/2013 6:33 PM	Flash Document	1,651 KB
blue - bow - sensei - 2 - idle.fla	4/14/2013 6:44 PM	Flash Document	1,783 KB
blue - bow - sensei - 2 -run.fla	4/14/2013 6:40 PM	Flash Document	3,020 KB
blue - DIEkatana - ninja - 1 - attack.fla	4/12/2013 9:39 PM	Flash Document	1,788 KB
blue - DIEkatana - ninja - 2 - idle.fla	4/12/2013 9:38 PM	Flash Document	1,015 KB
blue - DIEkatana - ninja - 2 - invis.fla	4/15/2013 7:19 PM	Flash Document	2,072 KB
blue - DIEkatana - ninja - 2 - SLAM.fla	4/12/2013 9:03 PM	Flash Document	1,140 KB
blue - DIEkatana - ninja - 2 -run.fla	4/12/2013 9:38 PM	Flash Document	2,282 KB
blue - DIEkatana - peasant - 2 - attack.fla	4/11/2013 8:37 PM	Flash Document	1,284 KB
blue - DIEkatana - peasant - 2 - idle.fla	4/11/2013 8:37 PM	Flash Document	1,128 KB
blue - DIEkatana - peasant - 2 - SLAM.fla	4/11/2013 6:57 PM	Flash Document	1,170 KB
blue - DIEkatana - peasant - 2 -run.fla	4/11/2013 8:37 PM	Flash Document	2,227 KB
blue - DIEkatana - samurai - 1 - attack.fla	4/9/2013 6:31 PM	Flash Document	1,175 KB
blue - DIEkatana - samurai - 1 - idle.fla	4/9/2013 6:01 PM	Flash Document	1,047 KB
blue - DIEkatana - samurai - 1 - run.fla	4/7/2013 2:20 PM	Flash Document	3,115 KB
blue - DIEkatana - samurai - 1 - SLAM.fla	4/10/2013 7:51 PM	Flash Document	1,233 KB
blue - DIEkatana - samurai - 2 - attack.fla	4/10/2013 8:00 PM	Flash Document	1,232 KB

Figure 3.9: Hundreds of .fla files were used to store the game's numerous animations.

Similarly, when the animations were in the form of a series of .png's and stored for use by Jeff, the folder directory paths had to be easy to follow. We ended up organizing our folders in the same way we named our files.

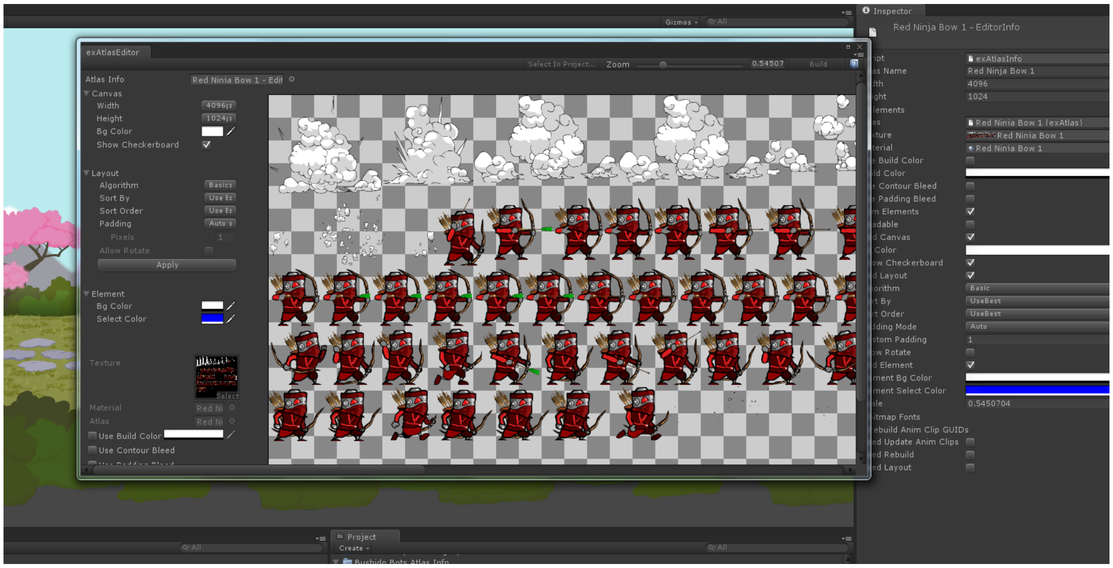


er.wpi.edu) (R:) ▸ My_Documents ▸ MQP ▸ SVN ▸ bushidobots ▸ Assets ▸ Animations (Raw Images) ▸ Blue ▸ DIE Katana ▸ Ninja ▸ Poof

Name	Date modified	Type	Size
blue - DIEkatana - ninja - 2 - invis0025.png	4/15/2013 7:29 PM	PNG File	2 KB
blue - DIEkatana - ninja - 2 - invis0024.png	4/15/2013 7:29 PM	PNG File	3 KB
blue - DIEkatana - ninja - 2 - invis0023.png	4/15/2013 7:29 PM	PNG File	3 KB
blue - DIEkatana - ninja - 2 - invis0022.png	4/15/2013 7:29 PM	PNG File	6 KB
blue - DIEkatana - ninja - 2 - invis0021.png	4/15/2013 7:29 PM	PNG File	6 KB
blue - DIEkatana - ninja - 2 - invis0020.png	4/15/2013 7:29 PM	PNG File	18 KB
blue - DIEkatana - ninja - 2 - invis0019.png	4/15/2013 7:29 PM	PNG File	18 KB
blue - DIEkatana - ninja - 2 - invis0018.png	4/15/2013 7:29 PM	PNG File	20 KB
blue - DIEkatana - ninja - 2 - invis0017.png	4/15/2013 7:29 PM	PNG File	20 KB
blue - DIEkatana - ninja - 2 - invis0016.png	4/15/2013 7:29 PM	PNG File	25 KB
blue - DIEkatana - ninja - 2 - invis0015.png	4/15/2013 7:29 PM	PNG File	25 KB
blue - DIEkatana - ninja - 2 - invis0014.png	4/15/2013 7:29 PM	PNG File	26 KB
blue - DIEkatana - ninja - 2 - invis0013.png	4/15/2013 7:29 PM	PNG File	26 KB
blue - DIEkatana - ninja - 2 - invis0012.png	4/15/2013 7:29 PM	PNG File	25 KB
blue - DIEkatana - ninja - 2 - invis0011.png	4/15/2013 7:29 PM	PNG File	23 KB
blue - DIEkatana - ninja - 2 - invis0010.png	4/15/2013 7:29 PM	PNG File	20 KB
blue - DIEkatana - ninja - 2 - invis0009.png	4/15/2013 7:29 PM	PNG File	23 KB
blue - DIEkatana - ninja - 2 - invis0008.png	4/15/2013 7:29 PM	PNG File	18 KB
blue - DIEkatana - ninja - 2 - invis0007.png	4/15/2013 7:29 PM	PNG File	18 KB
blue - DIEkatana - ninja - 2 - invis0006.png	4/15/2013 7:29 PM	PNG File	17 KB
blue - DIEkatana - ninja - 2 - invis0005.png	4/15/2013 7:29 PM	PNG File	18 KB
blue - DIEkatana - ninja - 2 - invis0004.png	4/15/2013 7:29 PM	PNG File	17 KB
blue - DIEkatana - ninja - 2 - invis0003.png	4/15/2013 7:29 PM	PNG File	17 KB
blue - DIEkatana - ninja - 2 - invis0002.png	4/15/2013 7:29 PM	PNG File	17 KB
blue - DIEkatana - ninja - 2 - invis0001.png	4/15/2013 7:29 PM	PNG File	17 KB
blue - DIEkatana - ninja - 2 - invis0025.pn...	4/19/2013 2:49 PM	META File	1 KB

Figure 3.10: A single animation's folder. Notice the length of the directory path at the top of the window.

From this point, Jeff would import the images into Unity, and start putting them in ex2D. He would first create an Atlas, which was essentially a spritesheet, by dragging a folder full of animation frames onto the pane shown below:



Atlas Info: Red Ninia Bow 1 - Edit

▼ Canvas
Width: 4096
Height: 1024
Bg Color
Show Checkerboard ☑

▼ Layout
Algorithm: Basic
Sort By: Use B
Sort Order: Use B
Padding: Auto
Pixels: 1
Allow Rotate ☐
Apply

▼ Element
Bg Color
Select Color

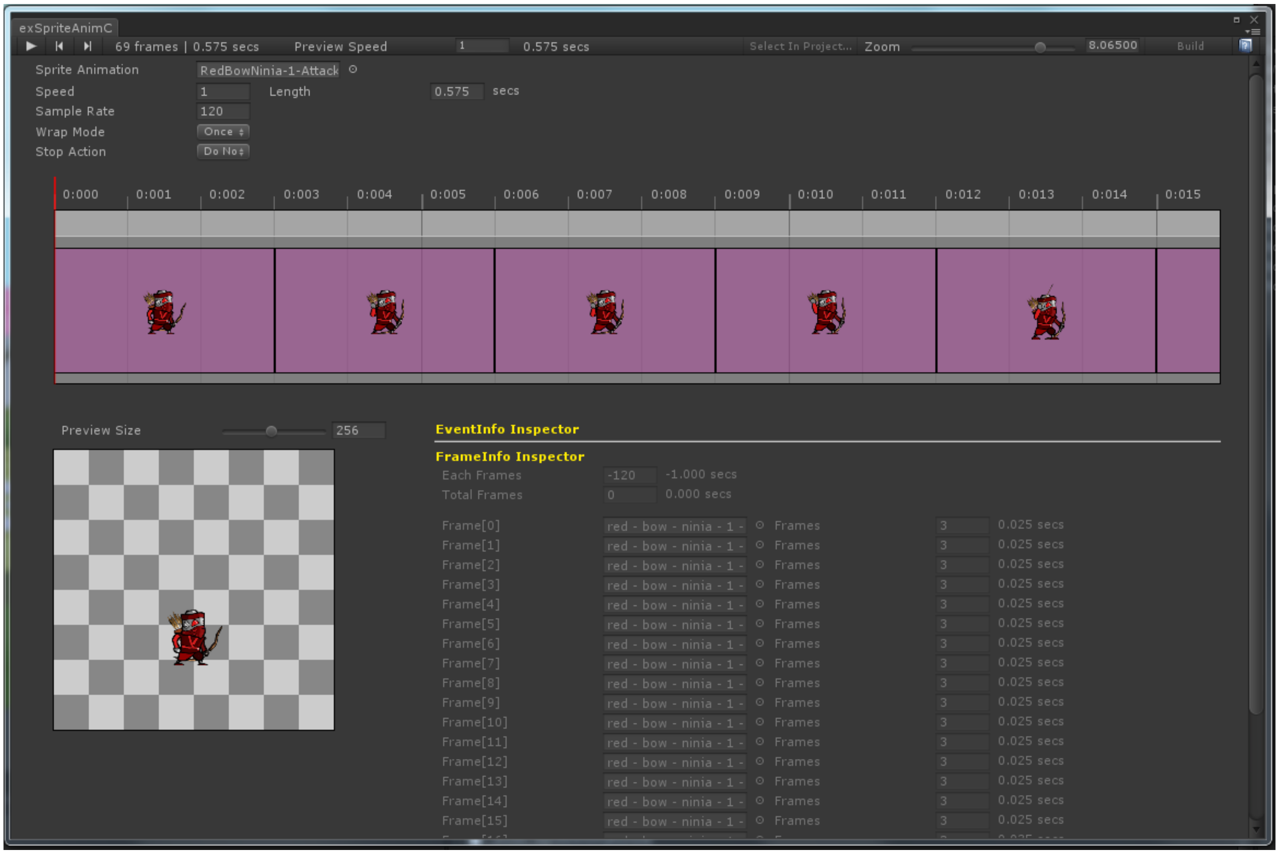
Texture
Material: Red Ni
Atlas: Red Ni
☐ Use Build Color
☐ Use Contour Bleed

Red Ninja Bow 1 - EditorInfo

ipt: exAtlasInfo
as Name: Red Ninja Bow 1
dth: 4096
ight: 1024
lements
as: Red Ninia Bow 1 (exAtlas)
xture: Red Ninia Bow 1
terial: Red Ninia Bow 1
e Build Color ☐
ild Color
e Contour Bleed ☐
e Padding Bleed ☐
m Elements ☑
adable ☐
d Canvas ☑
Color
ow Checkerboard ☑
d Layout ☑
orithm: Basic
t By: UseBest
t Order: UseBest
dding Mode: Auto
stom Padding: 1
ow Rotate ☐
d Element ☑
ment Bg Color
ment Select Color
ale: 0.5450704
itmap Fonts
ebuild Anim Clip GUIDs
ed Update Anim Clips ☐
ed Rebuild ☐
ed Layout ☐

Figure 3.11: A screenshot of a spritesheet being assembled in ex2D.

Ex2D would automatically cut out any transparent space in the frames, and position the remaining images to most efficiently use the space given on the spritesheet. Once the Atlas was built, a SpriteAnimation could be made. The frames were dragged into the SpriteAnimation editor, and from there they could be arranged in the proper order, the frame rate could be set for each individual frame, and the animation could be built. Once built, the animation could be added to a GameObject, and played in the game.



69 frames | 0.575 secs

Preview Speed 1 0.575 secs

Zoom 8.06500

Sprite Animation: RedBowNinia-1-Attack

Speed: 1 Length: 0.575 secs

Sample Rate: 120

Wrap Mode: Once

Stop Action: Do No

Preview Size: 256

EventInfo Inspector

FrameInfo Inspector

Each Frames: -120 -1.000 secs

Total Frames: 0 0.000 secs

Frame[0]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[1]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[2]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[3]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[4]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[5]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[6]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[7]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[8]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[9]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[10]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[11]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[12]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[13]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[14]	red - bow - ninia - 1 -	Frames	3	0.025 secs
Frame[15]	red - bow - ninia - 1 -	Frames	3	0.025 secs

Fig 3.12: A screenshot of an animation being assembled in ex2D.

Thankfully, importing non-animated assets proved to be just as simple and straightforward as we envisioned in pre-production. Wyatt would simply upload his finished art assets into the team's SVN as .png's. From here, Jeff would apply them to flat planes and position them in-game. This included all of the game's environment art, HUD elements, and menus.

The Case for 2D Art

Our team made the unusual decision to develop the game in 2D rather than 3D, as most MQP's do. There were numerous reasons which pulled us towards a 2D art style:

- We believed being 2D would help us stand out from other games
- We were eager to improve our 2D art skills
- We personally enjoy creating 2D assets more than 3D assets
- We are fans of 2D games and wish there were more of them
- We felt that a stylized 2D art would best capture the personality of our game

Advantages and Disadvantages of Adobe Flash

All in-game art was created using Adobe Flash -- a program focused on creating 2D animations, but also frequently used to develop games. We chose to develop in Adobe Flash for a variety of reasons. Primarily, we felt it lent itself better to creating a cartoony style than Photoshop, due its Flash Brush, a unique tool which works like a digital paint brush but vectorizes itself, so that it never loses detail when scaled up or down. Additionally, Flash comes with a convenient suite of 2D animation capabilities which made animating our characters vastly easier than it would have been otherwise.

Developing in Flash also introduced several obstacles as well. Fully utilizing Flash can be a long and confusing process. Initially, asset creation was slow, inefficient, and riddled with time-consuming mistakes. Fortunately, our artists continued to find new shortcuts throughout development which expedited the process significantly. Shane estimates that creating the katana's finalized swing animation in November took him around eight hours. In early April, he was able to create the dai-katana’s swing animation in under two hours.

Part 4: Technology

In choosing Unity as a game engine, we both considered the features it would provide as well as the team's familiarity with the engine. For the purposes of this game, Unity has many features that saved a great deal of development time. Its collision detection system and physics engine were both very useful in determining hitboxes for robots, firing arrows at constant velocities, providing bounds for the level, keeping track of players on capture points, and even for determining when players came close enough to trigger dialogues lines.

Additionally, Unity came with support for many different forms of input, with minimal work required on our part. Commands could be assigned to multiple different buttons across different interfaces simultaneously, such as keyboard, mouse, and Xbox controller. For the purposes of programming the game, what the user is using to input commands is ultimately immaterial. With the team's existing experience in C# - one of the scripting languages supported by Unity - the engine, and editor, Unity was the clear choice for this game.

In addition to the above, Unity provides a powerful suite of networking tools to aid in the creation of multiplayer games, which were used to great effect in the development of *Bushido Bots*. The primary tool Unity uses is a component called a 'Network View'. This is attached to objects in the game - such as players, arrows, and script containers - and allows many different functionalities. First, functions in scripts can be called on all players across the network through a Remote Procedure Call, or RPC. In this way, the server can relay commands from remote players to all other players. When one player commands their avatar to move forward, the server is notified via RPC, and then makes its own RPC to tell the rest of the clients to move that player's avatar.

Additionally, Network Views allow for automatic state synchronization from server to client. Simple values, such as position, rotation, and velocity, can be automatically synchronized across all clients. In addition, scripts can choose to automatically send data on the server side, and clients can automatically receive and process this data. In this way, the server can maintain an authoritative version of the game, and clients can keep accurate simulations of that game state. This form of a server-client relationship is called an "authoritative server", and it was the server architecture used in Bushdio Bots.

The one functionality that Unity did not provide was an easy method of handling sprites. The Unity engine can handle 2D textures and planes, but switching between textures is unwieldy and difficult to script. To overcome this obstacle, we purchased a third-party Unity extension called ex2D. What this tool allowed us to do was take a series of images, and craft them into single sprites or a working animation that could then be referenced from within a script. From there, we could tell a game object to play a specific animation, check on the status of the current animation, scale the sprites, change the orientation of the sprites, and add offsets to move the sprite without moving the underlying object. In the editor, we had such fine control over the sprite animations that we could set the duration for individual frames within a single animation.

However, all this power came at a cost. Building animations, while certainly more convenient than using basic Unity, was slow. Animations had to be built frame-by-frame, and only after all the frames had been imported into an Atlas first (see Pipeline in Section 3). Additionally, ex2D is not an industry-strength tool. It has massive RAM overhead, especially when editing Atlases and animations. The Unity editor would often crash if too many animations were accessed too quickly. These RAM issues carried over to the working build of the game, massively increasing the memory usage of the game.

In spite of these issues, the team decided to continue using ex2D. Other sprite plugins would have made it just as, if not more difficult to import sprites. We were able to work around the memory issues by carefully choosing a smaller subset of art assets to put into the game, and by designing our game to use as few sprites and animations as possible. Overall, the benefits of ex2D outweighed the cost and it was a large help to our team.

Part 5: Testing and Iteration

In *The Art of Game Design*, Jesse Schell suggests that the most valuable trait of a game designer is *listening*[3]. This is because ultimately, the designer must set aside his preconceptions and carefully and unassumingly listen to himself, his game, his client, his audience, and his team.

In order to listen to our audience, testing and iteration were integral to the development of *Bushido Bots*. From conception, *Bushido Bots* was designed to appeal to our friends and fellow gamers. From conception, we have also constantly engaged with friends, strangers, and fellow developers in order to create a top-notch experience for our players.

Our team conducted approximately twelve internal playtests, four small playtests (2-6 players) and two major playtests (8+ players).

Testing the stability of our networking was a particularly interesting obstacle for our game since our team was comprised of only three people. We were required to run public playtests in order to find networking issues, which often resulted in rather buggy and unfun matches for our testers.

Internal playtests (perhaps more accurately described as bug hunts) were integral for ironing out as many technical issues as possible for public playtests, but provided little value in terms of identifying the game's strength and weaknesses. In order to engage our audience and gather

[3] Schell, 4.

feedback on focused issues or at transitional periods during development, we ran small playtests with non-team members. Sometimes this would be Wyatt playing the game 1v1 with a friend and simply discussing the game, asking questions, and sharing ideas. Other times, this would mean 3v3 matches where the team would split itself between the two teams in order to get a feel for the current gameplay, as well as to gauge players' reactions to new changes and game balance.

The two major playtests were accompanied by feedback forms. These forms were distributed to the players before beginning the session. Players were encouraged to report bugs and provide feedback, ideas, and personal reactions. See *Appendix C: Playtest Feedback*, for the feedback we received from our major playtests.

The First Major Playtest

During the first major playtest, we had made the mistake of not privately playtesting the build beforehand. This led to several issues, and served as a very valuable lesson toward the playtesting process.

Essentially, the team had agreed that it would be smart to run a large playtest soon to test the stability of the multiplayer under the stress of 12 simultaneous players. Wyatt sent out an email to several of his friends, asking them to respond if they would be interested/available on a given night to gather and travel to one of the campus' computer labs, where everyone would log in and play together. He got several responses and everything appeared to be in order. This is where the team began to make some mistakes.

What we should have done:

Shane and Wyatt should have spent the two days before the big playtest identifying bugs and listing them for Jeff, and Jeff should have been on bug duty, working full speed to fix major issues so that we would not hit snags during the playtest.

What we actually did:

We continued to add more features and content to the game. Our logic was “the more stuff we get in, the more feedback we’ll have, and the more accurate it will be to the final product.” What we didn’t realize was that the game was highly unstable due to numerous bugs. Additionally, since we prepared the playtest build only a few hours before the playtest, we hadn’t taken the time to see if the new build even *functioned* on the lab computers.

So Wyatt gathered his playtesters, who had volunteered their free time to assist in the iteration process, and brought them to the Fuller Zoo Lab (a computer lab on campus). But when everyone tried to run the game, it only returned an error. Jeff frantically dug around for the problem and discovered that the new build’s RAM requirement exceeded the hard limits Internet Explorer allows for a web-player game. In the past, we’d never had this issue, so we never saw it coming, but this just reinforces the following lesson:

Pick a build, test it where you’ll be playing it, fix as many bugs as you can, and don’t add any new features to that build!

Anyway, we were determined to run the playtest, so we resolved to all return to our rooms, at which point Wyatt would send a set of instructions to all playtesters on how to run the game and connect to a match. And so everyone successfully joined the match and...

The game was very, very buggy. We got a lot of feedback about bugs, but ultimately it was a missed opportunity to get gameplay feedback, which is much more valuable than bug reports, since our team can provide that feedback on our own time.

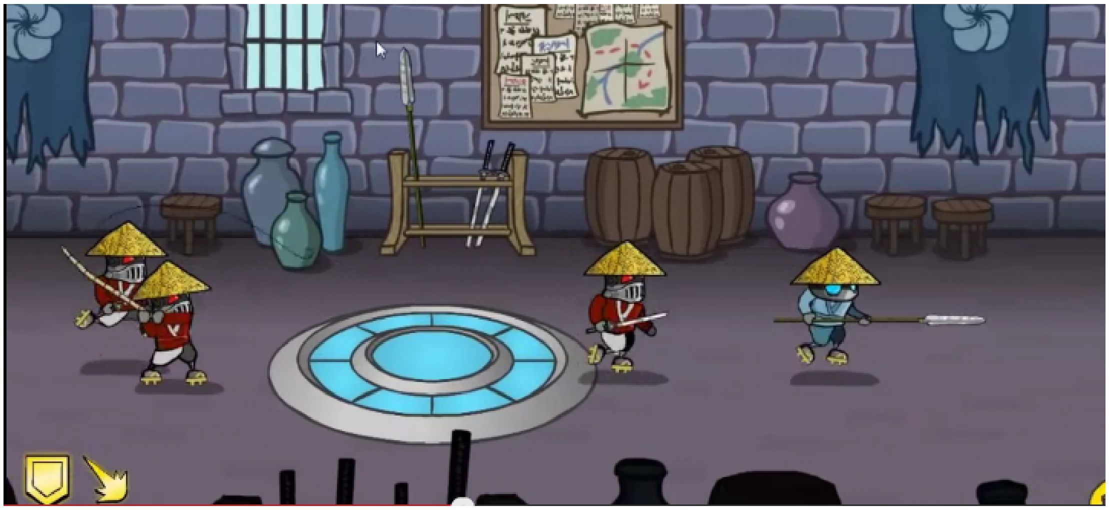


Figure 5.1: A screenshot from our public playtest.

Part of the pain of this lesson was that we had wasted the time of our playtesters. Between traveling to the lab, setting up the game, it not working, returning home, emailing instructions, and finally connecting, only to have the game have glaring technical issues, we didn't deliver what our playtesters expected, and that was disappointing for us. Overall, we didn't learn much about the game, but we learned a *lot* about how to run a playtest.

Thankfully, future playtests saw significantly less technical issues. These playtests were smaller and focused on game balance and identifying frustrating elements of gameplay. Feedback from these playtests can be found in Appendix C.

The Second Major Playtest

On April 23, our team conducted our second major playtest, collecting 12 players in a match simultaneously. Because of the lessons learned from our first major playtest, our team was prepared to run the second. We worked for four hours running a rigorous bug test to eliminate as many potentially playtest-ruining issues as possible. At the predetermined hour, our playtesters downloaded, installed, and entered into the game without issue, and the playtest carried out as planned.

While several of our players reported very few latency issues, others reported severe lag spikes, especially once all twelve players had joined the server. Overall, feedback was on the positive side, with all players confirming that they ‘found themselves laughing or smiling during gameplay’. Feedback for this playtest can be found in Appendix C.

Part 6: Post Mortem

What Went Wrong

Pipeline Problems

Our pipeline was simply not efficient. Every animation that was created had to be saved as a new file, even if it underwent a simple color change to be used on the other team. For example, a character's running animation could not be saved and modified outside of Flash-- this meant that the same character running with a katana and a bow had to be saved as two separate files and qualified as two unique assets. Every asset that was created had to be manually exported and brought into Unity, sometimes with additional modifications. This created more work for Wyatt and Shane, and a lot more work for Jeff. Although an elegant solution most likely exists, the easiest way to avoid these issues would have been to simply make a Flash game, but Jeff insists that coding and networking in Actionscript would have been disastrous. Furthermore, Adobe has stated that they are phasing out Flash, and as such, learning Actionscript is of limited use towards future employment[4].

Third Party Tools

Flash and Unity served their purposes well for the most part, but even though it was a paid product, ex2D was not of entirely professional quality. It greatly facilitated our pipeline and we would not have opted not to use it in any circumstance, but ex2D turned out to be poorly optimized and a memory hog and ultimately prevented us from getting as many different unique robot heads as we wanted into the game. It was also somewhat feature sparse. In the future,

[4] Rose, Mike. "Unity Drops Flash Support, says Adobe is not firmly commited." http://www.gamasutra.com/view/news/191112/Unity_drops_Flash_support_says_Adobe_is_not_firmly_committed.php.

we would most likely search for a better tool, even if it were more expensive.

Playtesting Woes

We had success finding bugs and the like early on when the team engaged in playtests. Unfortunately, there are only three of us, and our game is intended to be played by a minimum of 8-10 players at a time. We ended up running large scale public playtests much later into development than we should have, and ended up finding many new bugs. This was useful, but as a result we did not receive as much feedback on our design choices as we would have liked and felt like we wasted our playtesters' time. Thankfully, we found the game to be more balanced than we had initially anticipated, and finished in a state we are happy with balance-wise.

Demoing the Game

Since the game is multiplayer, demoing it without an internet connection is essentially impossible without having Jeff build a tech demo of the game where a single player can enter the game and face opponents without any AI. Ultimately, we consistently struggled to show the game to people (imagine playing *Team Fortress 2* by yourself - it's not really a game at that point). This made it hard to demonstrate the game's functionality. A possible solution might have been to create several gameplay trailers early on in development, or to program basic AI for players to engage.

What Went Well

Constant Communication

Our small team size made communication much easier. We met regularly throughout each week and had perfect knowledge of our roles and what was required from us at any given time. Team members were extremely useful for feedback as well as design discussion. Shane and Jeff frequently discussed the most efficient way to handle things like problematic animations to reduce the workload on both the art and tech sides of things. Shane learned how to use Flash more rapidly due to Wyatt's previous experience with the program, and by the end of development both artists were able to help each other work through any problems they faced during asset generation.

Scoping and Scalability

Bushido Bots was extremely well-scoped. The team discussed and classified which features could reasonably be implemented and which were wishlist items. The art was created to be as modular as possible, to allow a variety of different assets to be implemented relatively quickly. The team agreed that making a polished product was more important than adding numerous assets and features. Almost everything that the team wanted made it into the game, with most of what didn't being due to memory issues.

Stylized Art

Adobe Flash and peer feedback allowed our artists to create quality art that was stylized but still cohesive. A large variety of art was generated quickly, and Wyatt's in particular was iterated upon numerous times, with valuable aesthetic and technical feedback. The game looks and feels how we intended it to, and it has received praise from numerous players and onlookers.

Ambitious Tech

With only one programmer, twelve-player networking and incorporating hundreds of animations was a lot of work, but it turned out very well. Unity and ex2D were utilized as effectively as possible and the technical groundwork for our game was created early into development. The large variety of animations and special abilities are in the game and function correctly, the HUD elements do their job effectively, and twelve-player networking works extremely well.

Smart Design

The goal of our project was to create a game which players would genuinely enjoy enough to play multiple times. Meticulous planning and forethought went into the design and fine-tuning of the game, with all members of the team weighing in on design decisions both large and small. Based on player feedback, we feel that we accomplished our design goal.

What Surprised Us

People loved the art

Although Shane and Wyatt consider design their strongest development skill, our team was surprised to receive “I really like the art” as the most common positive feedback for the game. We believe that hand-drawn 2D art assets and a strong dedication to theming provided a visual style that pleased players more than we had anticipated.

Creating 2D art is challenging and time-consuming

Our artists initially assumed that creating 2D art would allow them to churn out assets very quickly. What we discovered, however, was that 2D art required as much, if not more effort, to generate as 3D art, due to the challenge of capturing all assets in the correct angle, color, and artistic style. For example, drawing the floors and walls for the spawn room took several days, while developing it in 3D would have taken under an hour.

A small team has its advantages

We expected that having a team of only three people would make creating a quality game difficult, and we were right. With 25% less development power, our team had to double-down and design wisely in order to make a game on par with other teams. However, we discovered the subtle advantages of small team size and maximized those advantages as much as possible.

First, communication was easier. Getting everyone into one place to discuss important development decisions was rarely a problem. Second, team roles were very clear-- no one was ever confused as to what was expected from them. This eliminated any possibility of social loafing, which cannot be emphasized enough. Third, and perhaps most importantly, our small team size made us very close friends. We found ourselves obligated by friendship to do a great job not just for ourselves, but for each other.

Part 7: Conclusion

VISION

Our core design statement for *Bushido Bots* was “Is it fun?” With this in mind, our team is incredibly proud of our final product. In our later playtests, we received generally positive feedback from our playtesters, many of which suggested that the game felt like something they could buy and play with their friends in their free time.

TECH

Technologically, the team succeeded. The game’s online multiplayer, while challenging to implement, was a huge accomplishment and greatly enhanced both the fun and the marketability of the game.

ART

Shane and Wyatt worked carefully to create a visually cohesive art style for the game, which permeates smoothly between the character art, HUD, menus, and environment. We feel that the final product approaches the professional quality seen in top-selling indie games such as *Castle Crashers* and *Super Meat Boy*.

DESIGN

The team took on a great challenge by attempting to create a competitive multiplayer beat-em-up, a genre which is largely unexplored. Through careful planning, intensive discussion, and iteration based on player feedback, the design team was able to synthesize a unique gameplay experience that rewards teamwork and skillful gameplay.

END PRODUCT

Bushido Bots is the culmination of seven months of intensive development as part of a standard academic workload. Our team has delivered a feature-complete game that resembles highly successful indie games such as *Castle Crashers* and *Super Meat Boy*. The game carefully combines familiar game mechanics into a new formula which is intended to hold our players' interest for numerous playthroughs. We are proud to have designed a game that was able to be developed in a shorter-than-average timespan by a smaller-than-average team, and hope that our project will inspire future Interactive Media and Game Development majors to create fun, exciting, and unique experiences that we, in turn, will have the opportunity to enjoy.

Works Cited

“Armor Samurai,” Unexpected Customs (http://2.bp.blogspot.com/_m8SHL7mBiRA/TExqO26oBwI/AAAAAAAAAHU/oMYpG8DHt-8/s1600/IMG_3699.jpg).

Castle Crashers weapons, (http://images2.wikia.nocookie.net/__cb20090104174330/castlecrashers/images/d/d7/Mastersheet.jpg).

"Ex2D." *ExDev Studio Ex2D*. N.p., n.d. Web. 22 Apr. 2013. (http://www.ex-dev.com/ex2d/)

Explosion effect sprite sheet, (http://i181.photobucket.com/albums/x259/Ricoman2000/Neogeo%20RMXP%20game/explosion2.jpg).

“First Position,” Journal of Combative Sport: Yumi: The Japanese Long Bow (http://ejmas.com/jcs/yumi/Firstposition.jpg).

Katana photograph, Kai Noromias Log (http://kainoromiastales.webs.com/photos/Weapons/katana.jpg).

LighterNoteProd, "Adobe Flash Bone Basics- Character Animation Part 1," (http://www.youtube.com/watch?v=pdzJKH0Stek).

Marutake, “Rusty Brown Byo-Toji Suit of Samurai Armor & Helmet,” Samurai Store International (http://www.samurai-store.com/armor/images/L032/L032armor_samuraistore.jpg).

Mortal Kombat 3 blood sprite sheet, (http://spritedatabase.net/files/snes/475/Sprite/MK3_Blood.gif).

*Ninja Master Costum*e. Halloween Costumes (http://images.halloweencostumes.com/ninja-master-costume-zoom.jp)

Paladin, Dan, “Marsh Land Construction,” (http://vimeo.com/823265#at=0).

puremrz, “Sakura Tree Zomg” (2009), digital painting (http://fc03.deviantart.net/fs48/f/2009/225/1/9/Sakura_Tree_Zomg_by_puremrz.png).

Sakura tree photograph, (http://www.9ori.com/store/media/images/5c5992f421.jpg).

Schell, Jesse. *The Art of Game Design: A Book of Lenses*. Burlington: Elsevier, 2008.

Seven Japanese Arrows. Antiques Reporter (http://img.antiquesreporter.com.au/090723WEAU/475.jpg)

Shoji Double Sided Sliding Door Kit. J-Life (http://www.jlifeinternational.com/shoji/shoji4honey.jpg).

Small-Butera, Alex, "Flash Animation Tutorial #3, Basic Animation," (http://www.youtube.com/watch?v=iIrJVUn0QT)

Smoke effect sprite sheet, (http://img152.imageshack.us/img152/1448/smoke1qv8.png).

"Source Multiplayer Networking." *Valve Developer Community*. Valve Corporation, n.d. Web. 22 Apr. 2013. (https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking)

Yari photograph, Japan Culture Center

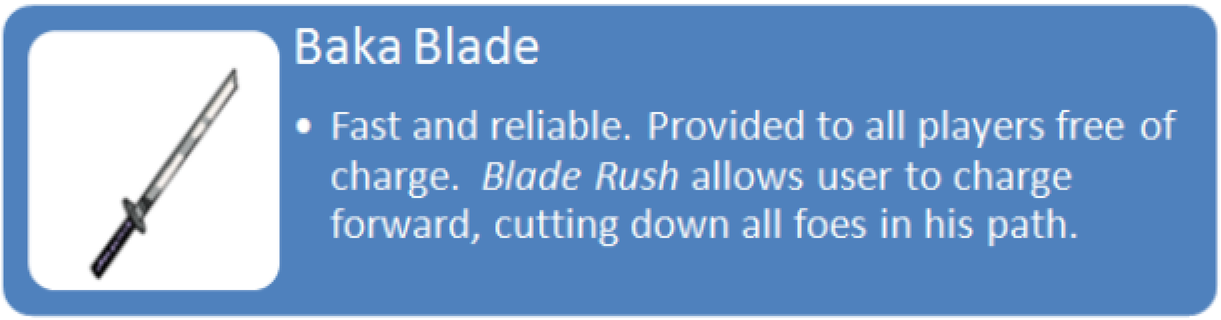
(http://1.bp.blogspot.com/-QGtFDlF6ml0/TdhLL1Vbu4I/AAAAAAAAAw0/80WVAvluGhM/s400/Yari-japanese+sword+2.jpg).

Appendix A: Weapons

The Katana ("Baka Blade")

The Baka Blade is the only free weapon that the player can choose. It attacks the fastest of the weapons and can be swung once or twice as a "combo attack." It was purposefully designed to be effective in a variety of situations, but dominant in none of them. The Baka Blade's special ability is the "Blade Rush", which allows players to teleport forward a short distance while dodging attacks and dealing damage.

The katana is free and thus comes with the least utility. It is, however, deadly in close range, and a sword wielder should be taken as seriously as any other. Blade Rush provides invincibility frames, which make it particularly useful in bypassing a spear's pushback, charging through incoming arrows, or delivering damage to an enemy and taking none in return, making this weapon good for skirmishing dangerous opponents.



Baka Blade

- Fast and reliable. Provided to all players free of charge. *Blade Rush* allows user to charge forward, cutting down all foes in his path.

The Bow ("Ronin's Recurve")

The Ronin's Recurve is a mid-cost projectile weapon that allows players to fire arrows at distant opponents. With each arrow, the player chooses how long they would like to charge their shot. With longer charge times, the arrows travel faster and do more damage. This presents a constant choice to the wielder of the Ronin's Recurve-- fire rapidly or commit to more precise, damaging attacks. It's special ability is "Ew-Goo Shot", which fires a slime arrow that deals no damage but temporarily traps an opponent in a ball of slime.

The bow is an important tool in forcing enemies to engage, since the wielder can deal damage without taking any until enemies approach. It is also a soft counter to the samurai, who can be outrun and peppered with arrows until he is ready to be defeated. Bow wielders find themselves particularly vulnerable in close range, which the ninja can reach easily using invisibility.

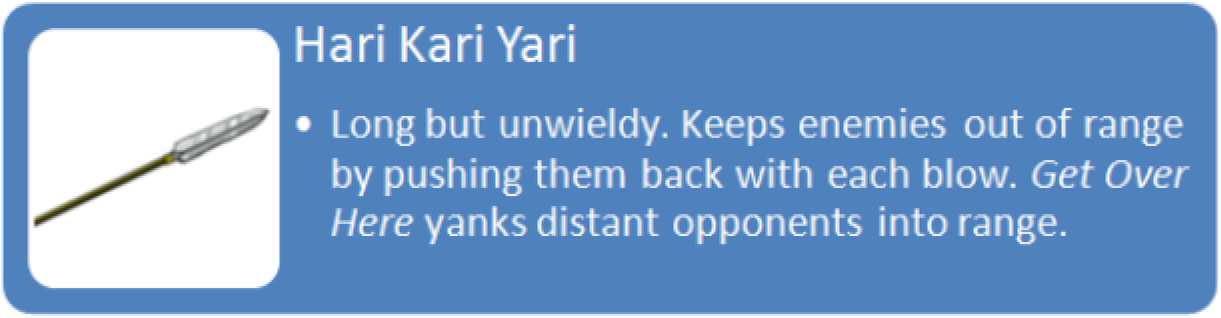
Ronin's Recurve

- Harasses enemies from a distance. Good for support, but vulnerable in close quarters. *Ew-Goo Shot* traps its victim in a thick layer of slime.

The Spear ("Hari Kari Yari")

The Hari Kari Yari is a mid-cost weapon and possesses the longest range of the game's melee weapons. It does more damage than the Baka Blade and also stuns and pushes opponents back, but it is slower and has a narrower hitbox. Its special ability is "Get Over Here", which gives players the ability to throw the Yari forward and yank it back with a chain attached to the shaft, stunning and pulling opponents closer.

The spear is a soft counter to the katana, since, if used properly, its knockback will continually push incoming sword-wielders away. Its special ability is particularly useful for pulling fleeing adversaries back into combat, or yanking archers into range.



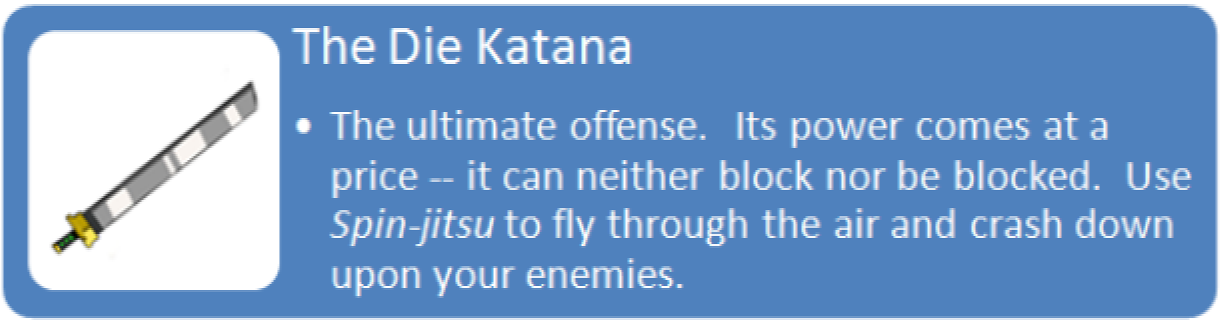
Hari Kari Yari

- Long but unwieldy. Keeps enemies out of range by pushing them back with each blow. *Get Over Here* yanks distant opponents into range.

The Dai-Katana (“Die Katana”)

The Die Katana is the biggest and most expensive weapon in the game. It has more range and significantly more power than the Baka Blade but is slower. It’s size prevents it from being able to parry or be parried, making it an excellent weapon for offensive players. The Die Katana’s special ability is “Spin-jitsu”, which causes the user to somersault into the air and slam the ground below, dodging attacks on the way.

This weapon is particularly expensive, and thus is rarer than other items. It is not, however, strictly a late-game item as all players start the match with enough gold to immediately purchase it. This may be seen as putting all your eggs in one basket, however, since it will cost the player most of their starting gold.



The Die Katana

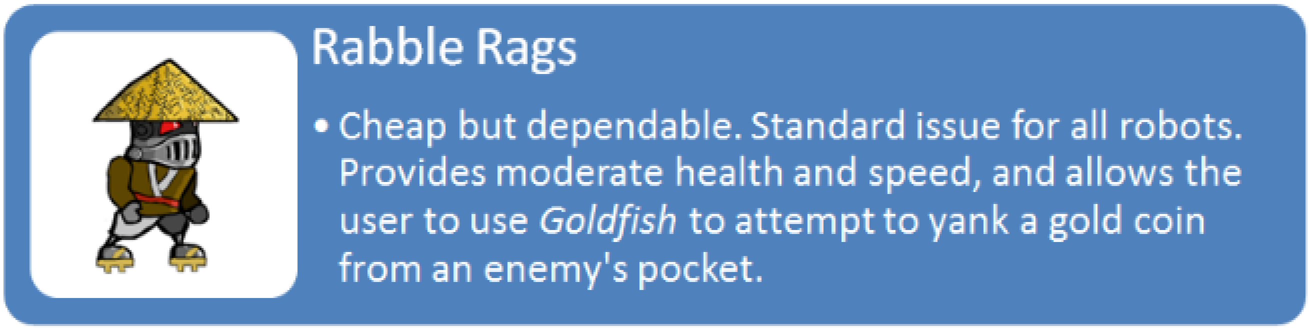
- The ultimate offense. Its power comes at a price -- it can neither block nor be blocked. Use *Spin-jitsu* to fly through the air and crash down upon your enemies.

Appendix B: Armors

The Peasant Outfit (“Rabble Rags”)

The Rabble Rags are the only free outfit in the game. They offer slightly below average health and average speed. Players wearing the Rabble Rags can use the ability “Gold Fish” which allows them to cast a bamboo fishing rod to attempt to steal gold from enemies’ pockets.

An all-around weak armor, the rabble rags are the second slowest armor, in front of the samurai but behind sensei and ninja. Its special ability is designed to help poor players gather enough gold to buy something better, and to punish enemies who are raking in too much gold. The Rabble Rags were intentionally designed to be weak in order to encourage players to switch to other armors. We felt this would create more visual diversity among players, who often choose to spend their gold on weapons rather than armor, thus resulting in a horde of peasants.



Rabble Rags

- Cheap but dependable. Standard issue for all robots. Provides moderate health and speed, and allows the user to use *Goldfish* to attempt to yank a gold coin from an enemy's pocket.

The Ninja Outfit (“Ninja’s Nighties”)

The Ninja’s Nighties exchange a low maximum health for the highest mobility in the game. This outfit grants the “Go Poof” ability, allowing the wearer to become completely invisible to enemies until they attack or use an ability.

Having the lowest health means that the ninja is unsuitable for prolonged combat. Instead, the ninja should rely on his superior speed and ability to disappear in order to succeed. Superior speed makes the ninja perfect for skirmishing with weapons such as the bow and katana, while the ability to vanish makes him good for launching deadly ambushes using the spear or dai-katana. As such, the ninja is well-suited to all four weapons, but requires different tactics with each. A ninja is much more durable with a sensei to heal him, as a small health pool means ninjas can only take a few hits. The ability to stealth makes ninjas particularly well-suited to ambushing enemy archers and senseis who hide in the back and are often left vulnerable.

Ninja's Nighties

- Lightweight yet flimsy. Its wearers can outrun all others, but have the smallest health pool to compensate. Activate *Go Poof* to drop a smoke bomb and disappear.

The Samurai Armor ("Samurai Steel")

By far the most protective outfit, the Samurai Steel rewards its purchasers with a huge health pool at the cost of reduced mobility. Skilled players will activate the "Oni-Chan" ability before charging into battle, gaining increased movement speed and invulnerability to the effects of blocking, pushing and pulling, and stuns for seven second.

Having a tremendous health pool makes samurai ideal for charging in head-first, or staunchly defending control points against enemy attacks. Having low move speed means samurai are particularly vulnerable to being skirmished by not only archers, but the blade rush attack of katana-wielders. It will take quite a while, however, to bring down a samurai, whose health nearly doubles that of the ninja. In order to overcome his crippling movement deficiency, the samurai's special ability gives him seven seconds of immunity to pushback, stuns, and pulls, as well as giving him a nice speed boost to chase down enemies or flee. He should be careful when activating this ability though, as the cooldown is particularly long.

Samurai Steel

- Tough and slow. This heavy suit trades mobility for high health. Unleashing *Oni-Chan* makes the wearer faster and immune to movement-impairing effects for a short time.

The Sensei Outfit ("Sensei's Silks")

The Sensei's Silks provide a notable health increase andslightly above-average mobility. A sensei holds a support role on their team, utilizing the "Zen Healing" ability to heal himself and nearby allies. This allows his team to maintain their momentum, even after numerous engagements or prolonged combat.

Ultimately, the sensei is vital towards a team's prolonged success. Without a sensei, a succeeding team soon finds their health worn away and are overcome by the non-stop waves of respawning foes. With a sensei, this damage can be mitigated through judicious use of Zen Healing. Now, the team is able is withstand numerous attacks without being defeated, which not only means they are able to preserve their presence on the map, it also means that those players will spend less gold repurchasing their expensive equipment.

Sensei's Silks

- Sturdy and supportive. The sensei has a sizeable health pool and average speed. His *Zen Healing* ability restores health to nearby allies.

Appendix C: Playtest Survey and Feedback

MAJOR PLAYTEST 1:

1. Did any art assets stand out as not matching or not looking as good as others?

> *"The red team peasant was definitely brown. I thought it was another team."*
>
> *"No, they looked good"*
>
> *"Power up icons could be tuned up"*
>
> *"No, the art is good!"*
>
> *"Everything looks pretty great!"*

2. Did you feel that matches or spawn times were too long or short?

> *"The points capped quickly with one man"*
>
> *"Seemed ok"*
>
> *"No"*
>
> *"Spawn times are fine"*
>
> *"The one match where we killed each other was a bit on the short side I thought"*

3. Did you feel that the losing team was able to make a comeback?

> *"Couldn't tell"*
>
> *"No"*
>
> *"Not enough playable matches to tell"*

4. Did any weapons or armors feel too good or bad for their price?

> *"The bow shot too slowly and had too small of a hitbox to be useful."*

“Samurai armor is really good”

“I couldn’t upgrade”

“Only tried sword and a little of bow, so dunno
.”
“No”

5. Which weapons and armors did you like the most? What did you like about them?”

“Spear for the throw”

“Samurai + Spear”

“Samurai special attack is awesome”

“Ninja’s disappear is badass”

“Didn’t try enough of them. But the animations for the spear are awesome and I’m looking forward to that.”

6. What was the most frustrating part of the game (aside from nasty bugs)?

“Connecting, hit detection”
“Besides the bugs nothing”

“THE NASTY BUGS! But also button placement was a tad awkward”

7. If you have any other comments, suggestions, or cool ideas, leave them here!

“Let me play a later version, it’ll be awesome!”

“We lost?!”

“So your spawn variable overrides your character’s position when the xbox analog hits center. It seems like a networking bug of some sort.”

“GODZILLA. The art is great and I’m sure the programming will be fine once the kicks are worked out.”

MAJOR PLAYTEST 2:

1. **Did any art assets stand out as not matching or not looking as good as others?**

“Not particularly.“

“No I did not notice any art assets as not matching or not looking as good as others.”

“No, everything looked appropriate and blended well.”

“The flaming samurai helmet didn’t look quite as polished as everything else (which looked quite good).”

2. **At any point, did you experience lag so severe that it made combat feel unplayable? If so, how often? Every engagement? About every third engagement? Only a few times?**

“Not unplayable, but there was a fair amount of lag that appeared every other engagement or so. I'd be moving and still get hit by an enemy who wasn't near me and when I let go of a directional input I'd telepot back to where I started.”

“Yes, about every other engagement. It got a bit crippling in close quarters.”

“It happened about five times, but what would happen is I would be running but taking damage, and it turned out I was really somewhere else once I took my finger off of the arrow keys.”

“Nothing made it really "unplayable" to me. I mean there were times (about every third) that me or someone else was porting around and such, it was annoying, but didn't really stop me from playing.”

3. **Did you feel that matches or spawn times were too long or short?**

“They could be slightly longer in order to punish people who might just run in on a suicide run for quick hits.”

“The matches and spawn times were fine.”

“I thought that they were fine, they did not seem too long or short.”

“Spawn times were fine.”

4. **Did you feel that teamwork was very important towards success?**

“Somewhat. Combat was definitely easer with a team mate helping you but anyone with enough skill could take 2-3 enemies at once depending on their class and weapon.”

“Teamwork didn't always seem too important, but we were all learning what worked and was effective on an individual player level. Teamwork comes later.”

“Yes I believe teamwork would be necessary to succeed in most circumstances.”

“There were time where 3 blue guys would chase a red guy around in a circle for a while because they couldn't hit him, but besides that, teamwork helps a lot.”

5. **Did any weapons or armors feel too powerful or not powerful enough?**

“Samurai and Ninja seemed way better than the Sensei and Villager. The ninja may need a nerf to how long he can stay invisible because at the moment it's an infinite amount of time, so it's fairly easy to equip the Die Katana and kill an enemy with a Spin-jitsu an enemy who doesn't know you're there. I didn't particularly notice the healing from the Sensei Zen Healing and the villager's gold fish seems to not work properly as I don't think anyone managed to successfully steal any coins. Sword could perhaps use a slightly faster start up time in order to make up for its lack of range. Nobody wanted to use it once they got more coins. I never used the bow so I can't comment on it. The Die Katana seems really good at the moment, most people were using it in the game and it makes a Samurai an absolute tank. Hari Kari Yari might need a slightly larger hitbox since it seemed difficult to get hits off. “

“The Die-katana (the big one) seemed a bit overpowered, but would have been easier to dodge with less lag. The sensei didn't do anything.”

“I felt the spear was a little weak, and I also thought the third armor that would heal was pretty weak.”

“The Die-katana and the guy with a lot of health was sort of OP if the other team didn't do the same thing.”

6. Which weapons and armors did you enjoy the most?

“I mainly used the Ninja Die Katana combo.”

“ I personally enjoy ranged support, so the bow was my favorite.”

“ I enjoyed the samurai armor and the large sword. Using the special attack for the large sword

was awesome when landed properly.”

“I used mostly what I mentioned above.”

7. **Did you read the weapon and armor descriptions in the armory?**

“Not at first, but only because I knew what they did from previous playtests.”

“Yes”

“Yes I did because I wanted to make the right decision on what was best to use.”

“Not really, no.”

8. **Did you notice yourself smiling or laughing at any point while playing?**

“Absolutely”

“Yes, some of the dialogue was funny, and getting a kill in a game like that can be very satisfying when it was a hard fought fight.”

“I was mostly smiling and laughing because I was playing with people I know, but I know I would have that much regardless of who I was playing with.”

“Yeah I was laughing pretty hard, its fun to play with your friends and to kill/troll them. I'm not sure I would have the same experience with random people.”

9. **Did you feel that you had to travel a lot or too far to find action/combat?**

“Not really. Once their was more people in the server it didn't seem that difficult to find an enemy”

“ I really only got this feeling when I would get near the middle and it would inexplicably teleport me back.”

“Because the enemy was rolling over us it was pretty easy to find the enemy in our spawn or the first capture point.”

“Nah it was just what you would expect for a game like this.”

10. **Did the game objective, mechanics, or controls ever seem confusing?**

“The basic objective of the game seemed to be conveyed well, although I didn't realize I could

block until I read the Die Katana description."

" Nothing seemed confusing."

"Not at all, I thought it was pretty easy."

"I didn't know how to use my special powers at first since I just jumped into the game. Idk if it was mentioned anywhere, but I didn't notice. But regardless, I figured it out pretty quickly."

11. **What was your favorite part of the game? Be honest: least favorite part?**

"Favorite: Interesting class and weapon mechanic. I like the idea of starting off as an army of villagers and slowly expanding from there. Least: Either a couple more class/weapon choices or make villager and Baka Blade useful. "

"I enjoyed when the fights had a larger number of people on both sides. I wasn't fond of the lag, and I didn't feel like capturing the points were giving us any tangible benefits besides the victory ticker going down."

"My favorite part was the combat in the game. That kind of brings me to my least favorite part, which was some of the bugs in the game. I was struggling to engage enemies because of the lag where I would swing once they would disappear, and then would reappear randomly somewhere else. If this was fixed I would find the game much more enjoyable."

"Running around the enemy base and having them all try to kill me. Starting a new round with the shitty default class/weapon. Also the lag kinda sucked."

12. If you have any comments, suggestions, or cool ideas, leave them here!

"There were a couple glitches in the game: The scoreboard doesn't show the names of people who joined before you. There was something weird going on with the display of the middle point for me. It would be white when it said we had captured it on the top of the HUD. Some of the attacks seem a little off, like it would look like the enemy would be hit but they wouldn't take damage. I don't know if that's because of lag or the hitboxes themselves. I encountered this problem especially with the Baka Blade's special and the Die Katana's regular attack. Villager's special should be a little quicker and have a larger "hitbox" Like I said earlier nobody used it successfully to steal coins. Coins seemed kind of useless once you got the 5 or so you needed to get to your class of choice so once you got 2 kills you were pretty much all set, but that may just be because of the altered demo version we got in order to try out all of the weapons easily. I'm still curious to see where the robo-dragon does in all of this. Maybe alternate maps with differences such as traps? It would be entertaining to use the Spear's special ability to pull an enemy into a spike pit or jump over a ledge with a Spin-jitsu."

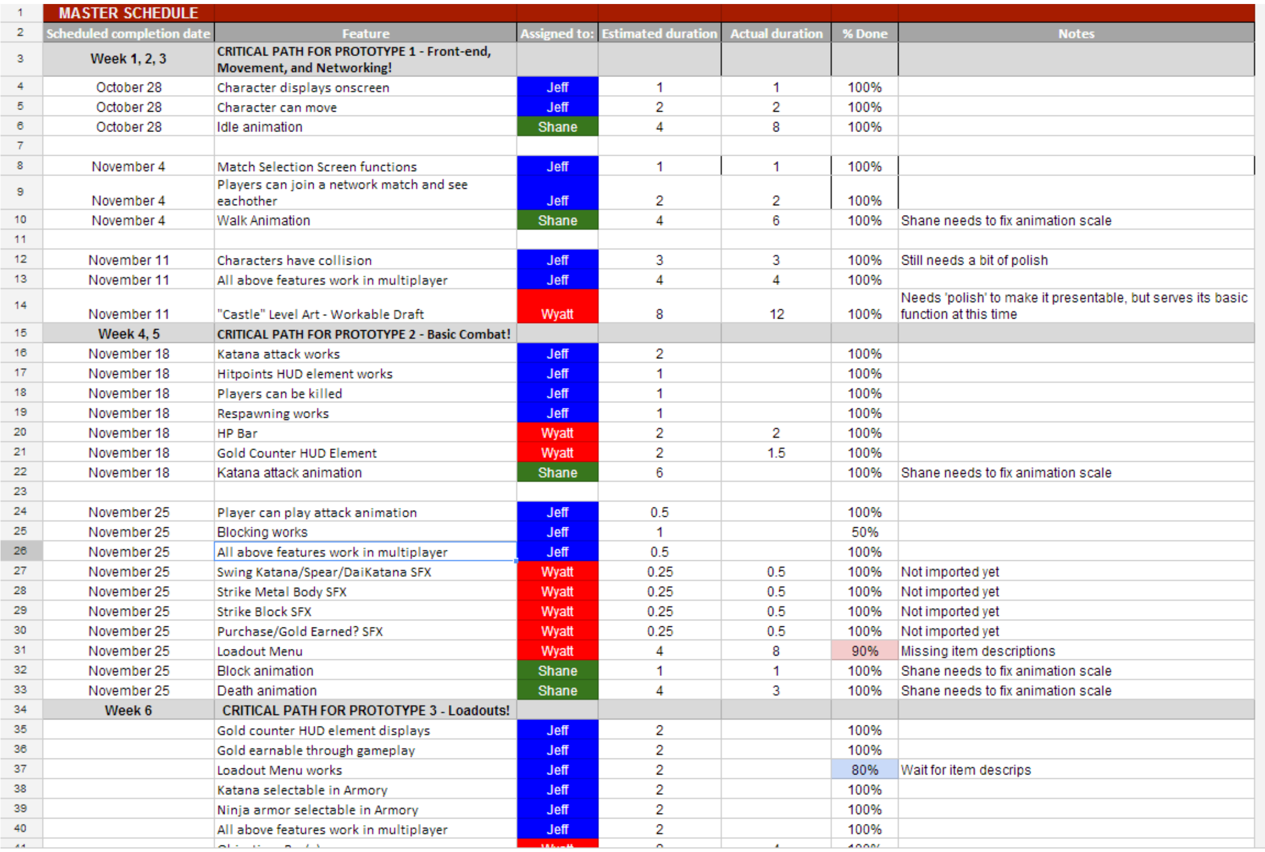
“All around a fun game!”

“I think that if the large sword had a range when it landed (like affected the enemies next to the landing), it would make it more useful. I also never found the fishing pole to actually work, but if it could be improved that would be great.I'm not entirely sure what the purpose of the gold was.”

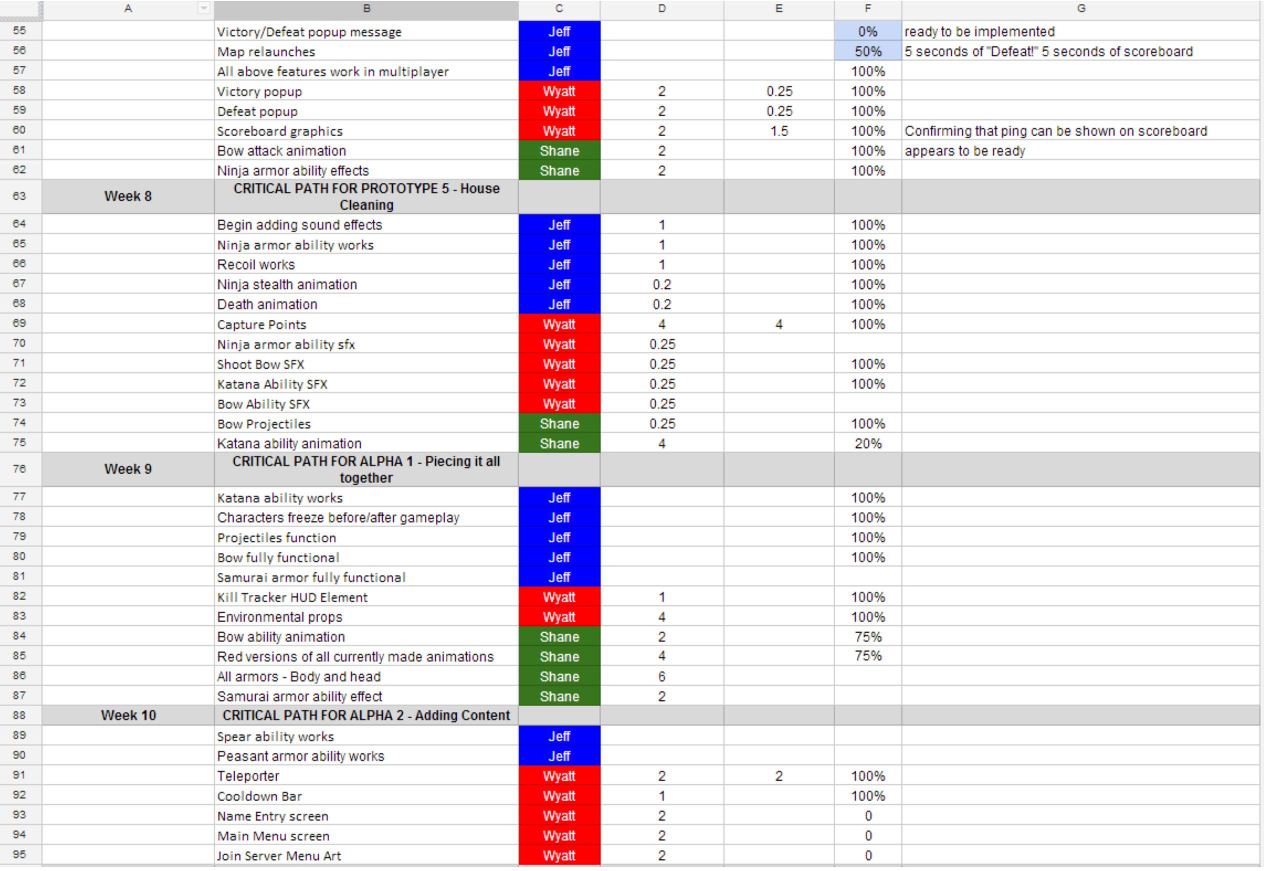
“I could always buy the best armor/weapon. I'm not sure if I just always had enough gold, or this was just a glitch.”

Appendix D: Early Development Feature Schedule

Pictures of our original feature schedule. We replaced it with a newer, simpler one more reminiscent to agile product backlogs, later in development.



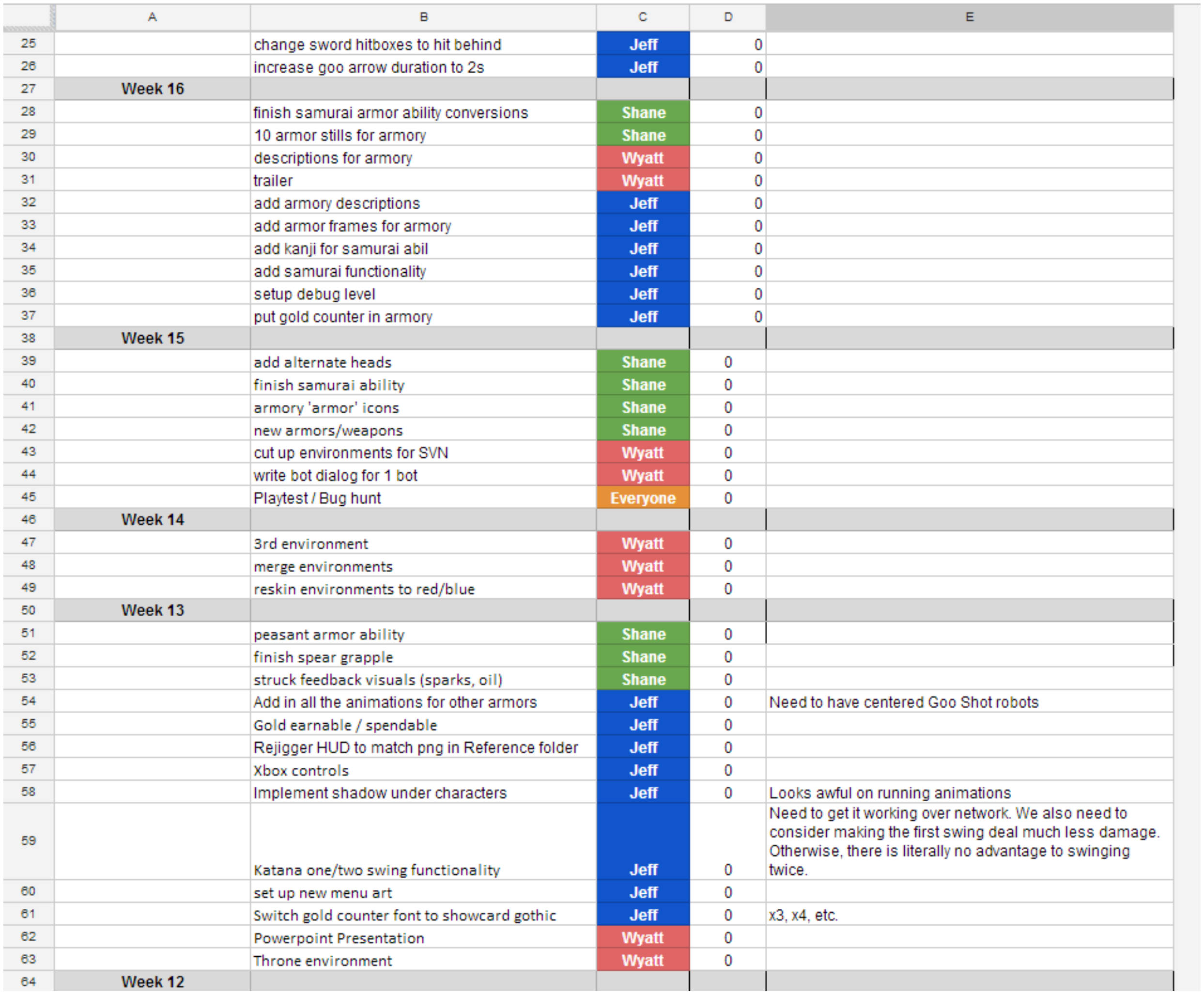
1	MASTER SCHEDULE						
2	Scheduled completion date	Feature	Assigned to:	Estimated duration	Actual duration	% Done	Notes
3	**Week 1, 2, 3**	**CRITICAL PATH FOR PROTOTYPE 1 - Front-end, Movement, and Networking!**					
4	October 28	Character displays onscreen	Jeff	1	1	100%	
5	October 28	Character can move	Jeff	2	2	100%	
6	October 28	Idle animation	Shane	4	8	100%	
7							
8	November 4	Match Selection Screen functions	Jeff	1	1	100%	
9	November 4	Players can join a network match and see eachother	Jeff	2	2	100%	
10	November 4	Walk Animation	Shane	4	6	100%	Shane needs to fix animation scale
11							
12	November 11	Characters have collision	Jeff	3	3	100%	Still needs a bit of polish
13	November 11	All above features work in multiplayer	Jeff	4	4	100%	
14	November 11	"Castle" Level Art - Workable Draft	Wyatt	8	12	100%	Needs 'polish' to make it presentable, but serves its basic function at this time
15	**Week 4, 5**	**CRITICAL PATH FOR PROTOTYPE 2 - Basic Combat!**					
16	November 18	Katana attack works	Jeff	2		100%	
17	November 18	Hitpoints HUD element works	Jeff	1		100%	
18	November 18	Players can be killed	Jeff	1		100%	
19	November 18	Respawning works	Jeff	1		100%	
20	November 18	HP Bar	Wyatt	2	2	100%	
21	November 18	Gold Counter HUD Element	Wyatt	2	1.5	100%	
22	November 18	Katana attack animation	Shane	6		100%	Shane needs to fix animation scale
23							
24	November 25	Player can play attack animation	Jeff	0.5		100%	
25	November 25	Blocking works	Jeff	1		50%	
26	November 25	All above features work in multiplayer	Jeff	0.5		100%	
27	November 25	Swing Katana/Spear/DaiKatana SFX	Wyatt	0.25	0.5	100%	Not imported yet
28	November 25	Strike Metal Body SFX	Wyatt	0.25	0.5	100%	Not imported yet
29	November 25	Strike Block SFX	Wyatt	0.25	0.5	100%	Not imported yet
30	November 25	Purchase/Gold Earned? SFX	Wyatt	0.25	0.5	100%	Not imported yet
31	November 25	Loadout Menu	Wyatt	4	8	90%	Missing item descriptions
32	November 25	Block animation	Shane	1	1	100%	Shane needs to fix animation scale
33	November 25	Death animation	Shane	4	3	100%	Shane needs to fix animation scale
34	**Week 6**	**CRITICAL PATH FOR PROTOTYPE 3 - Loadouts!**					
35		Gold counter HUD element displays	Jeff	2		100%	
36		Gold earnable through gameplay	Jeff	2		100%	
37		Loadout Menu works	Jeff	2		80%	Wait for item descrips
38		Katana selectable in Armory	Jeff	2		100%	
39		Ninja armor selectable in Armory	Jeff	2		100%	
40		All above features work in multiplayer	Jeff	2		100%	



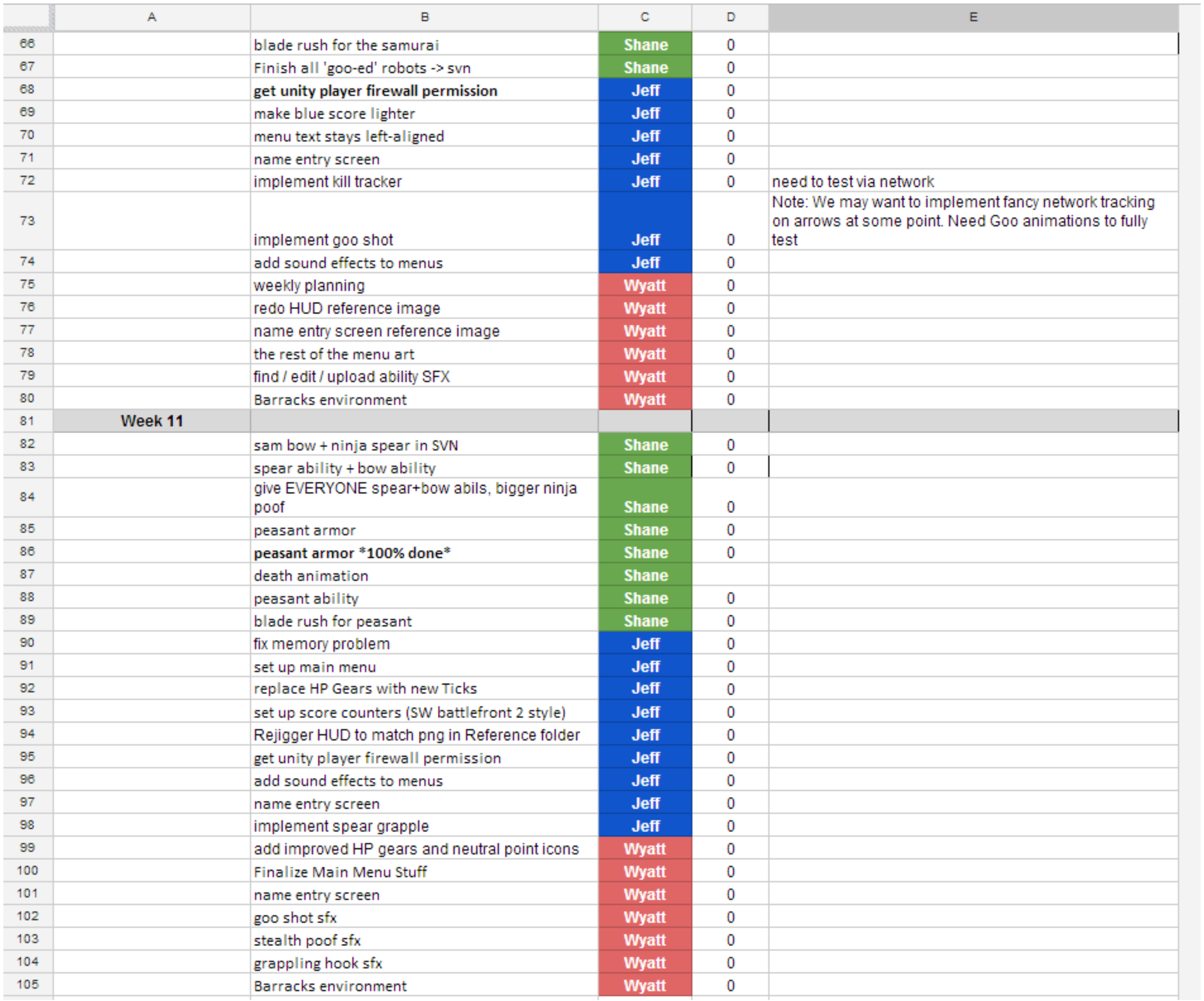
	A	B	C	D	E	F	G
55		Victory/Defeat popup message	Jeff			0%	ready to be implemented
56		Map relaunches	Jeff			50%	5 seconds of "Defeat!" 5 seconds of scoreboard
57		All above features work in multiplayer	Jeff			100%	
58		Victory popup	Wyatt	2	0.25	100%	
59		Defeat popup	Wyatt	2	0.25	100%	
60		Scoreboard graphics	Wyatt	2	1.5	100%	Confirming that ping can be shown on scoreboard
61		Bow attack animation	Shane	2		100%	appears to be ready
62		Ninja armor ability effects	Shane	2		100%	
63	**Week 8**	**CRITICAL PATH FOR PROTOTYPE 5 - House Cleaning**					
64		Begin adding sound effects	Jeff	1		100%	
65		Ninja armor ability works	Jeff	1		100%	
66		Recoil works	Jeff	1		100%	
67		Ninja stealth animation	Jeff	0.2		100%	
68		Death animation	Jeff	0.2		100%	
69		Capture Points	Wyatt	4	4	100%	
70		Ninja armor ability sfx	Wyatt	0.25			
71		Shoot Bow SFX	Wyatt	0.25		100%	
72		Katana Ability SFX	Wyatt	0.25		100%	
73		Bow Ability SFX	Wyatt	0.25			
74		Bow Projectiles	Shane	0.25		100%	
75		Katana ability animation	Shane	4		20%	
76	**Week 9**	**CRITICAL PATH FOR ALPHA 1 - Piecing it all together**					
77		Katana ability works	Jeff			100%	
78		Characters freeze before/after gameplay	Jeff			100%	
79		Projectiles function	Jeff			100%	
80		Bow fully functional	Jeff			100%	
81		Samurai armor fully functional	Jeff				
82		Kill Tracker HUD Element	Wyatt	1		100%	
83		Environmental props	Wyatt	4		100%	
84		Bow ability animation	Shane	2		75%	
85		Red versions of all currently made animations	Shane	4		75%	
86		All armors - Body and head	Shane	6			
87		Samurai armor ability effect	Shane	2			
88	**Week 10**	**CRITICAL PATH FOR ALPHA 2 - Adding Content**					
89		Spear ability works	Jeff				
90		Peasant armor ability works	Jeff				
91		Teleporter	Wyatt	2	2	100%	
92		Cooldown Bar	Wyatt	1		100%	
93		Name Entry screen	Wyatt	2		0	
94		Main Menu screen	Wyatt	2		0	
95		Join Server Menu Art	Wyatt	2		0	

Appendix E: Late Development Feature Schedule

Pictures of our newer feature schedule, with much of the information fields stripped away to encourage team members to actually fill out their info. Also made reading the document much easier.



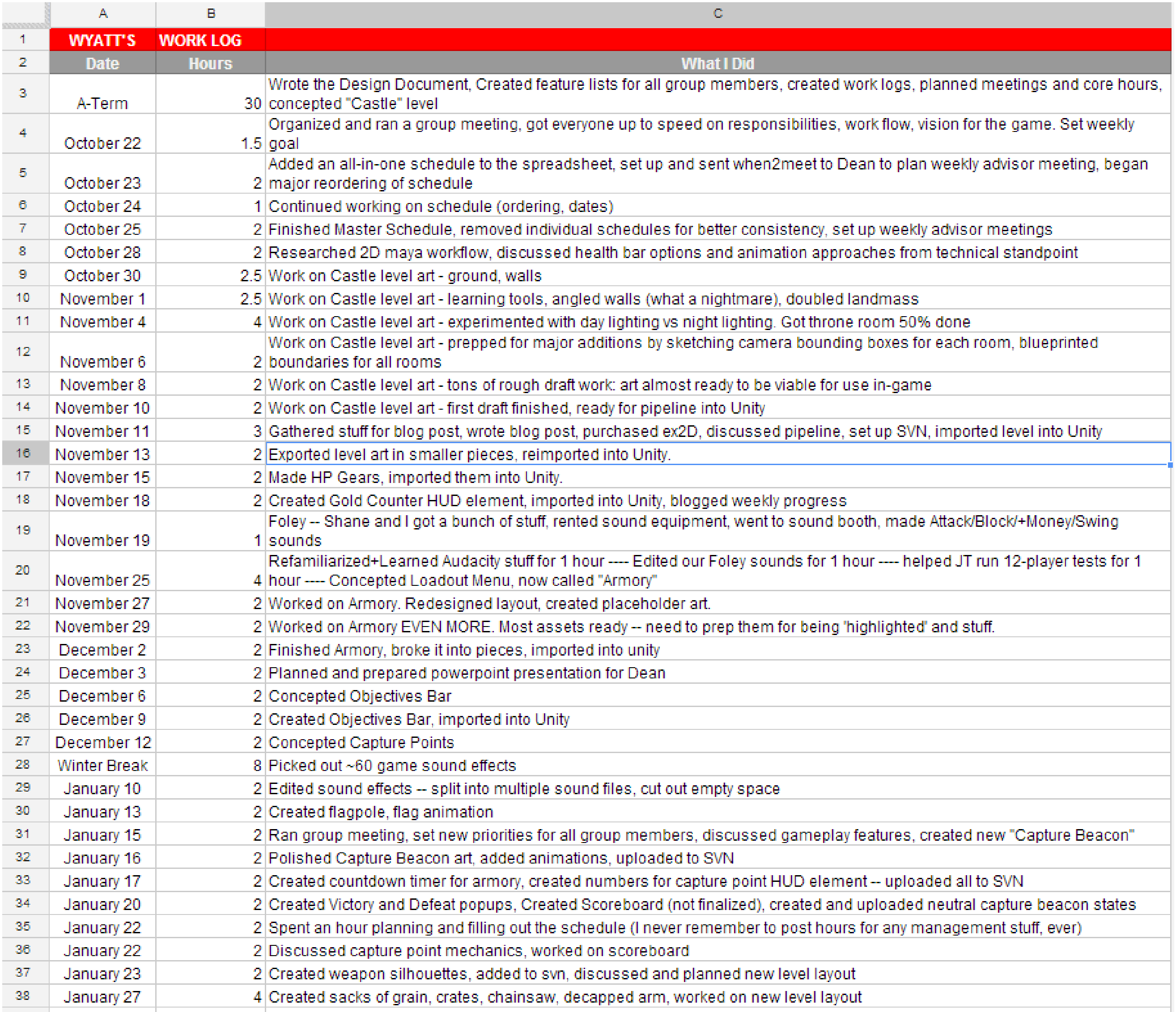
	A	B	C	D	E
25		change sword hitboxes to hit behind	Jeff	0	
26		increase goo arrow duration to 2s	Jeff	0	
27	Week 16				
28		finish samurai armor ability conversions	Shane	0	
29		10 armor stills for armory	Shane	0	
30		descriptions for armory	Wyatt	0	
31		trailer	Wyatt	0	
32		add armory descriptions	Jeff	0	
33		add armor frames for armory	Jeff	0	
34		add kanji for samurai abil	Jeff	0	
35		add samurai functionality	Jeff	0	
36		setup debug level	Jeff	0	
37		put gold counter in armory	Jeff	0	
38	Week 15				
39		add alternate heads	Shane	0	
40		finish samurai ability	Shane	0	
41		armory 'armor' icons	Shane	0	
42		new armors/weapons	Shane	0	
43		cut up environments for SVN	Wyatt	0	
44		write bot dialog for 1 bot	Wyatt	0	
45		Playtest / Bug hunt	Everyone	0	
46	Week 14				
47		3rd environment	Wyatt	0	
48		merge environments	Wyatt	0	
49		reskin environments to red/blue	Wyatt	0	
50	Week 13				
51		peasant armor ability	Shane	0	
52		finish spear grapple	Shane	0	
53		struck feedback visuals (sparks, oil)	Shane	0	
54		Add in all the animations for other armors	Jeff	0	Need to have centered Goo Shot robots
55		Gold earnable / spendable	Jeff	0	
56		Rejigger HUD to match png in Reference folder	Jeff	0	
57		Xbox controls	Jeff	0	
58		Implement shadow under characters	Jeff	0	Looks awful on running animations
59		Katana one/two swing functionality	Jeff	0	Need to get it working over network. We also need to consider making the first swing deal much less damage. Otherwise, there is literally no advantage to swinging twice.
60		set up new menu art	Jeff	0	
61		Switch gold counter font to showcard gothic	Jeff	0	x3, x4, etc.
62		Powerpoint Presentation	Wyatt	0	
63		Throne environment	Wyatt	0	
64	Week 12				



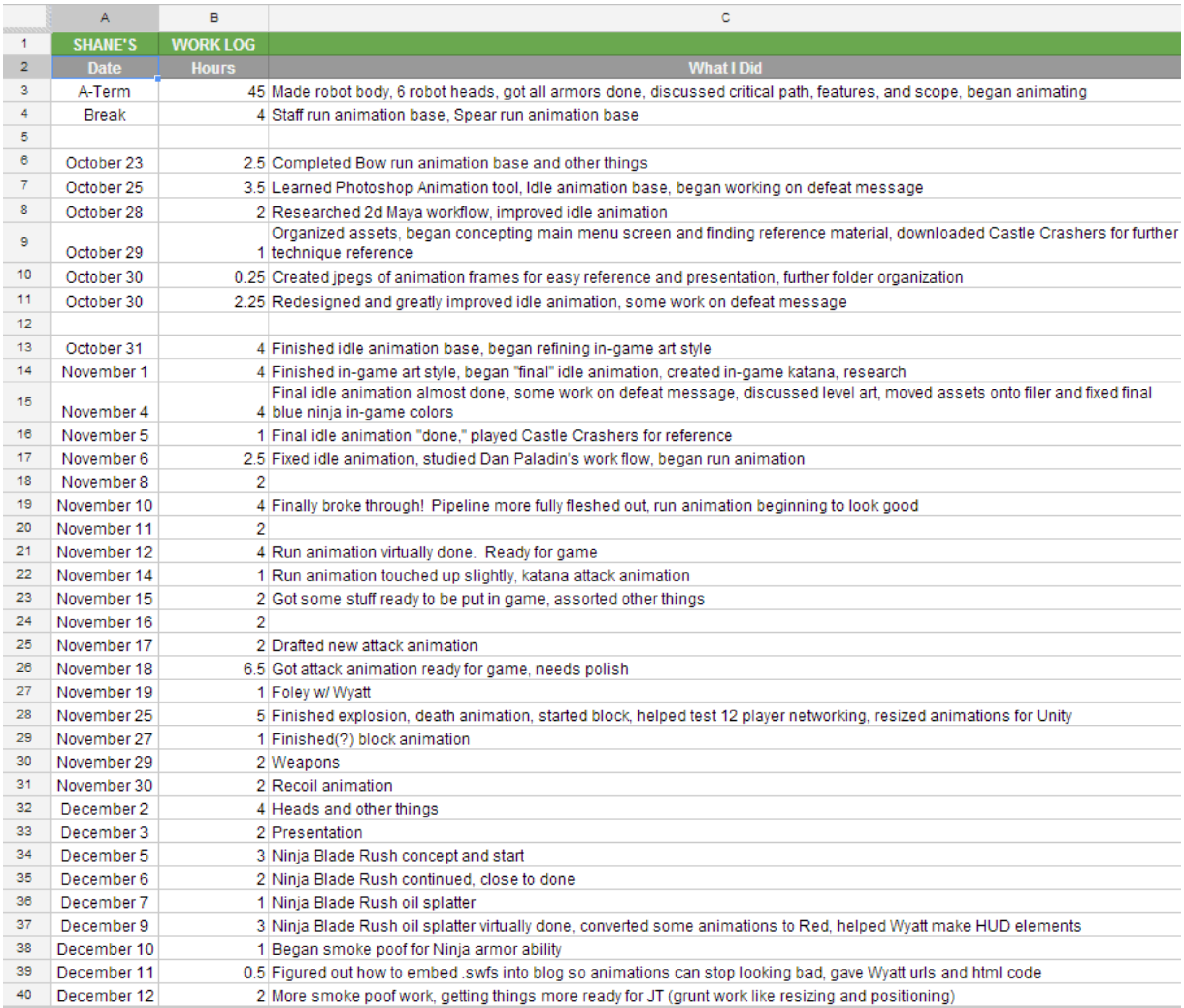
	A	B	C	D	E
66		blade rush for the samurai	Shane	0	
67		Finish all 'goo-ed' robots -> svn	Shane	0	
68		**get unity player firewall permission**	Jeff	0	
69		make blue score lighter	Jeff	0	
70		menu text stays left-aligned	Jeff	0	
71		name entry screen	Jeff	0	
72		implement kill tracker	Jeff	0	need to test via network
73		implement goo shot	Jeff	0	Note: We may want to implement fancy network tracking on arrows at some point. Need Goo animations to fully test
74		add sound effects to menus	Jeff	0	
75		weekly planning	Wyatt	0	
76		redo HUD reference image	Wyatt	0	
77		name entry screen reference image	Wyatt	0	
78		the rest of the menu art	Wyatt	0	
79		find / edit / upload ability SFX	Wyatt	0	
80		Barracks environment	Wyatt	0	
81	**Week 11**				
82		sam bow + ninja spear in SVN	Shane	0	
83		spear ability + bow ability	Shane	0	
84		give EVERYONE spear+bow abils, bigger ninja poof	Shane	0	
85		peasant armor	Shane	0	
86		**peasant armor *100% done***	Shane	0	
87		death animation	Shane		
88		peasant ability	Shane	0	
89		blade rush for peasant	Shane	0	
90		fix memory problem	Jeff	0	
91		set up main menu	Jeff	0	
92		replace HP Gears with new Ticks	Jeff	0	
93		set up score counters (SW battlefront 2 style)	Jeff	0	
94		Rejigger HUD to match png in Reference folder	Jeff	0	
95		get unity player firewall permission	Jeff	0	
96		add sound effects to menus	Jeff	0	
97		name entry screen	Jeff	0	
98		implement spear grapple	Jeff	0	
99		add improved HP gears and neutral point icons	Wyatt	0	
100		Finalize Main Menu Stuff	Wyatt	0	
101		name entry screen	Wyatt	0	
102		goo shot sfx	Wyatt	0	
103		stealth poof sfx	Wyatt	0	
104		grappling hook sfx	Wyatt	0	
105		Barracks environment	Wyatt	0	

Appendix F: Work Logs

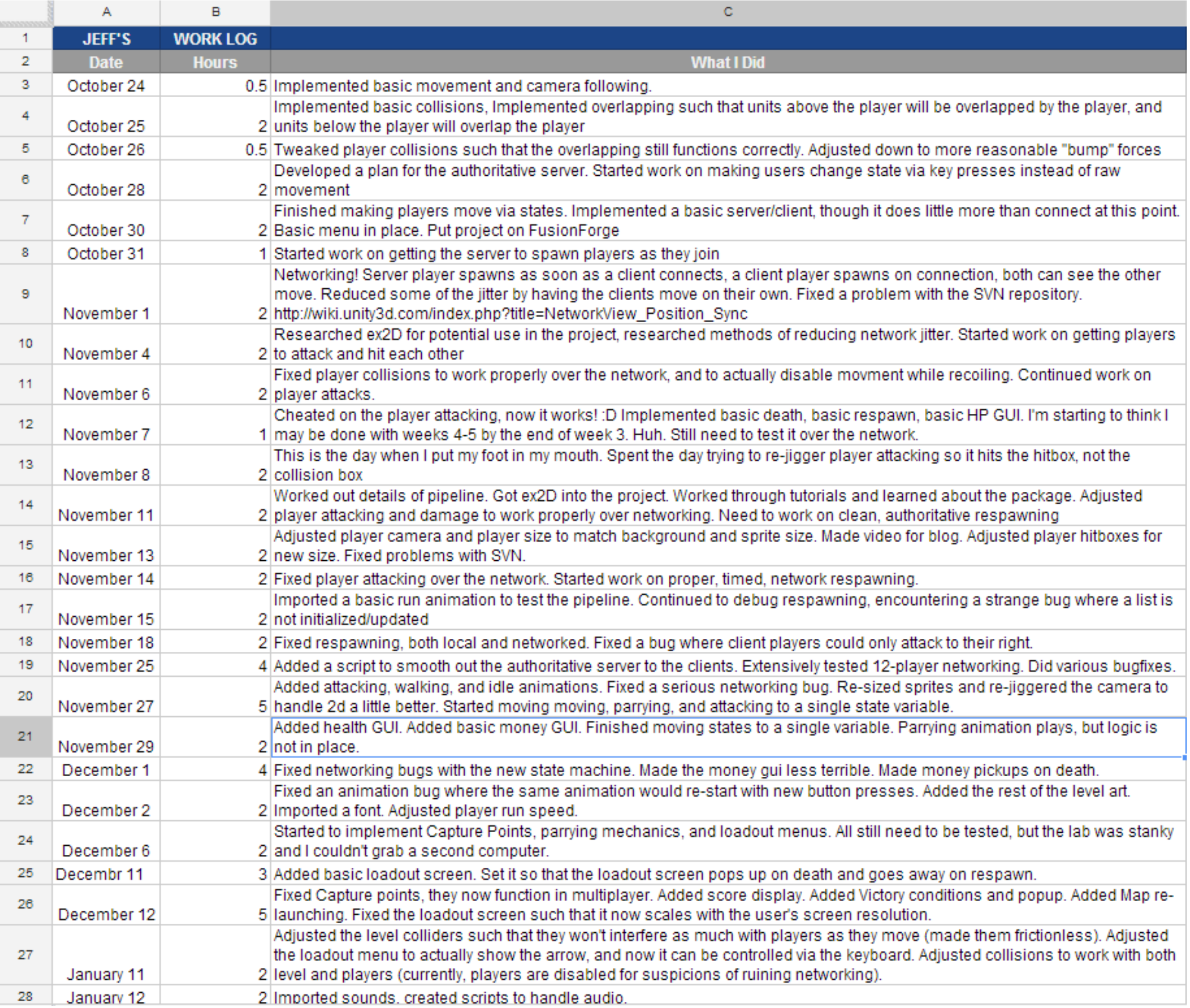
A screencap of each of our worklogs, demonstrating our method for recording what we had accomplished during a given week. We agreed to undersell, rather than oversell, the time we had invested, to encourage ourselves to spend more time working each week.



	A	B	C
1	WYATT'S	WORK LOG	
2	Date	Hours	What I Did
3	A-Term	30	Wrote the Design Document, Created feature lists for all group members, created work logs, planned meetings and core hours, concepted "Castle" level
4	October 22	1.5	Organized and ran a group meeting, got everyone up to speed on responsibilities, work flow, vision for the game. Set weekly goal
5	October 23	2	Added an all-in-one schedule to the spreadsheet, set up and sent when2meet to Dean to plan weekly advisor meeting, began major reordering of schedule
6	October 24	1	Continued working on schedule (ordering, dates)
7	October 25	2	Finished Master Schedule, removed individual schedules for better consistency, set up weekly advisor meetings
8	October 28	2	Researched 2D maya workflow, discussed health bar options and animation approaches from technical standpoint
9	October 30	2.5	Work on Castle level art - ground, walls
10	November 1	2.5	Work on Castle level art - learning tools, angled walls (what a nightmare), doubled landmass
11	November 4	4	Work on Castle level art - experimented with day lighting vs night lighting. Got throne room 50% done
12	November 6	2	Work on Castle level art - prepped for major additions by sketching camera bounding boxes for each room, blueprinted boundaries for all rooms
13	November 8	2	Work on Castle level art - tons of rough draft work: art almost ready to be viable for use in-game
14	November 10	2	Work on Castle level art - first draft finished, ready for pipeline into Unity
15	November 11	3	Gathered stuff for blog post, wrote blog post, purchased ex2D, discussed pipeline, set up SVN, imported level into Unity
16	November 13	2	Exported level art in smaller pieces, reimported into Unity.
17	November 15	2	Made HP Gears, imported them into Unity.
18	November 18	2	Created Gold Counter HUD element, imported into Unity, blogged weekly progress
19	November 19	1	Foley -- Shane and I got a bunch of stuff, rented sound equipment, went to sound booth, made Attack/Block/+Money/Swing sounds
20	November 25	4	Refamiliarized+Learned Audacity stuff for 1 hour ---- Edited our Foley sounds for 1 hour ---- helped JT run 12-player tests for 1 hour ---- Concepted Loadout Menu, now called "Armory"
21	November 27	2	Worked on Armory. Redesigned layout, created placeholder art.
22	November 29	2	Worked on Armory EVEN MORE. Most assets ready -- need to prep them for being 'highlighted' and stuff.
23	December 2	2	Finished Armory, broke it into pieces, imported into unity
24	December 3	2	Planned and prepared powerpoint presentation for Dean
25	December 6	2	Concepted Objectives Bar
26	December 9	2	Created Objectives Bar, imported into Unity
27	December 12	2	Concepted Capture Points
28	Winter Break	8	Picked out ~60 game sound effects
29	January 10	2	Edited sound effects -- split into multiple sound files, cut out empty space
30	January 13	2	Created flagpole, flag animation
31	January 15	2	Ran group meeting, set new priorities for all group members, discussed gameplay features, created new "Capture Beacon"
32	January 16	2	Polished Capture Beacon art, added animations, uploaded to SVN
33	January 17	2	Created countdown timer for armory, created numbers for capture point HUD element -- uploaded all to SVN
34	January 20	2	Created Victory and Defeat popups, Created Scoreboard (not finalized), created and uploaded neutral capture beacon states
35	January 22	2	Spent an hour planning and filling out the schedule (I never remember to post hours for any management stuff, ever)
36	January 22	2	Discussed capture point mechanics, worked on scoreboard
37	January 23	2	Created weapon silhouettes, added to svn, discussed and planned new level layout
38	January 27	4	Created sacks of grain, crates, chainsaw, decapped arm, worked on new level layout



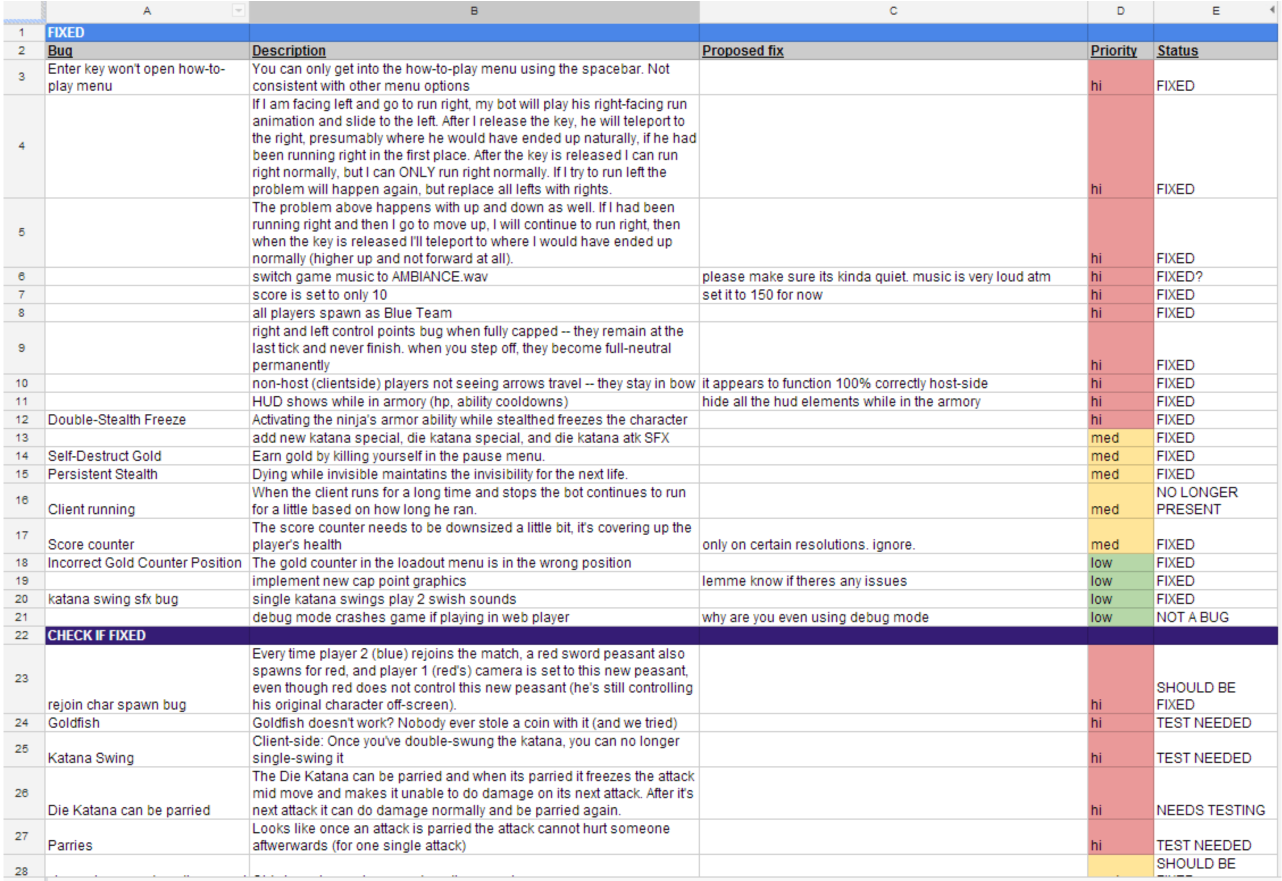
	A	B	C
1	SHANE'S	WORK LOG	
2	Date	Hours	What I Did
3	A-Term	45	Made robot body, 6 robot heads, got all armors done, discussed critical path, features, and scope, began animating
4	Break	4	Staff run animation base, Spear run animation base
5			
6	October 23	2.5	Completed Bow run animation base and other things
7	October 25	3.5	Learned Photoshop Animation tool, Idle animation base, began working on defeat message
8	October 28	2	Researched 2d Maya workflow, improved idle animation
9	October 29	1	Organized assets, began concepting main menu screen and finding reference material, downloaded Castle Crashers for further technique reference
10	October 30	0.25	Created jpegs of animation frames for easy reference and presentation, further folder organization
11	October 30	2.25	Redesigned and greatly improved idle animation, some work on defeat message
12			
13	October 31	4	Finished idle animation base, began refining in-game art style
14	November 1	4	Finished in-game art style, began "final" idle animation, created in-game katana, research
15	November 4	4	Final idle animation almost done, some work on defeat message, discussed level art, moved assets onto filer and fixed final blue ninja in-game colors
16	November 5	1	Final idle animation "done," played Castle Crashers for reference
17	November 6	2.5	Fixed idle animation, studied Dan Paladin's work flow, began run animation
18	November 8	2	
19	November 10	4	Finally broke through! Pipeline more fully fleshed out, run animation beginning to look good
20	November 11	2	
21	November 12	4	Run animation virtually done. Ready for game
22	November 14	1	Run animation touched up slightly, katana attack animation
23	November 15	2	Got some stuff ready to be put in game, assorted other things
24	November 16	2	
25	November 17	2	Drafted new attack animation
26	November 18	6.5	Got attack animation ready for game, needs polish
27	November 19	1	Foley w/ Wyatt
28	November 25	5	Finished explosion, death animation, started block, helped test 12 player networking, resized animations for Unity
29	November 27	1	Finished(?) block animation
30	November 29	2	Weapons
31	November 30	2	Recoil animation
32	December 2	4	Heads and other things
33	December 3	2	Presentation
34	December 5	3	Ninja Blade Rush concept and start
35	December 6	2	Ninja Blade Rush continued, close to done
36	December 7	1	Ninja Blade Rush oil splatter
37	December 9	3	Ninja Blade Rush oil splatter virtually done, converted some animations to Red, helped Wyatt make HUD elements
38	December 10	1	Began smoke poof for Ninja armor ability
39	December 11	0.5	Figured out how to embed .swfs into blog so animations can stop looking bad, gave Wyatt urls and html code
40	December 12	2	More smoke poof work, getting things more ready for JT (grunt work like resizing and positioning)



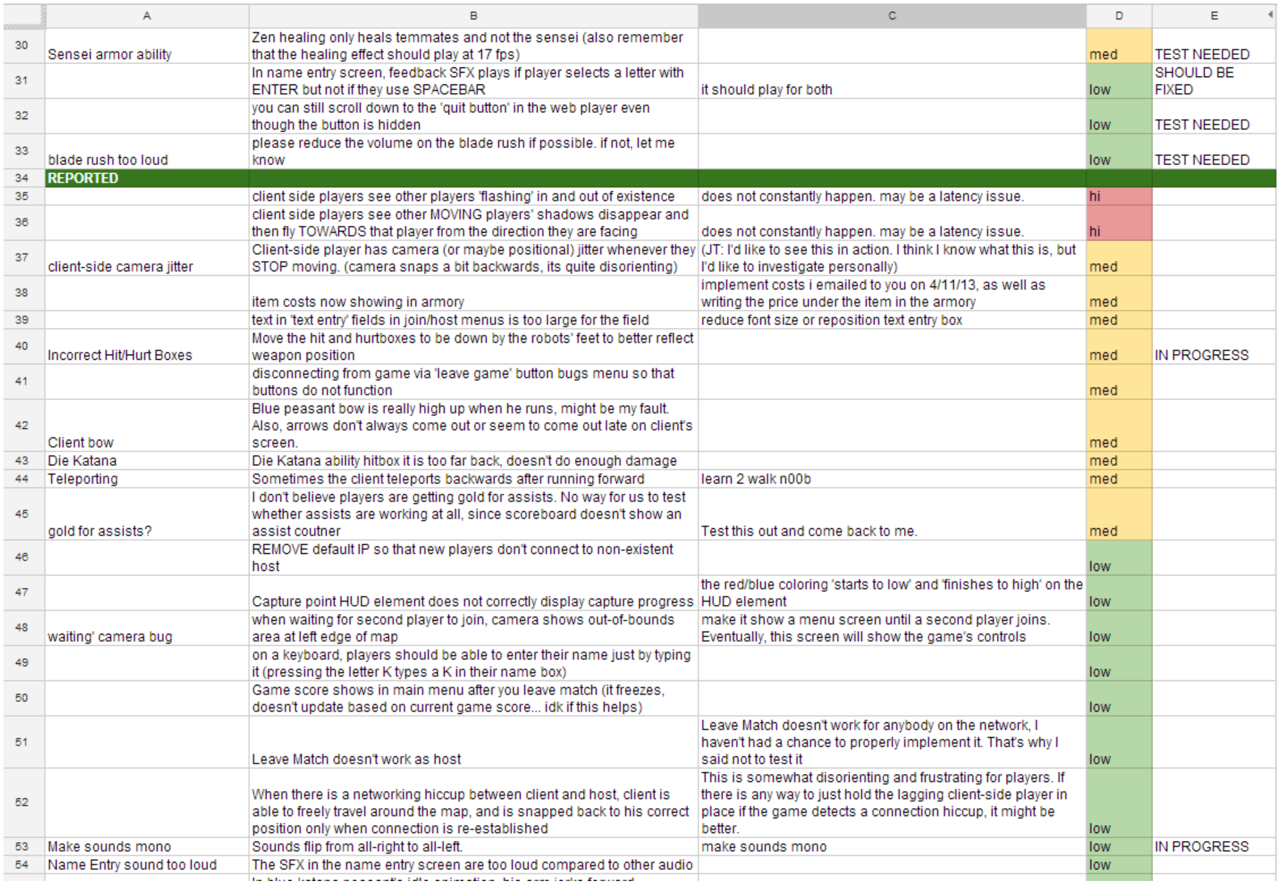
	A	B	C
1	JEFF'S	WORK LOG	
2	Date	Hours	What I Did
3	October 24	0.5	Implemented basic movement and camera following.
4	October 25	2	Implemented basic collisions, Implemented overlapping such that units above the player will be overlapped by the player, and units below the player will overlap the player
5	October 26	0.5	Tweaked player collisions such that the overlapping still functions correctly. Adjusted down to more reasonable "bump" forces
6	October 28	2	Developed a plan for the authoritative server. Started work on making users change state via key presses instead of raw movement
7	October 30	2	Finished making players move via states. Implemented a basic server/client, though it does little more than connect at this point. Basic menu in place. Put project on FusionForge
8	October 31	1	Started work on getting the server to spawn players as they join
9	November 1	2	Networking! Server player spawns as soon as a client connects, a client player spawns on connection, both can see the other move. Reduced some of the jitter by having the clients move on their own. Fixed a problem with the SVN repository. http://wiki.unity3d.com/index.php?title=NetworkView_Position_Sync
10	November 4	2	Researched ex2D for potential use in the project, researched methods of reducing network jitter. Started work on getting players to attack and hit each other
11	November 6	2	Fixed player collisions to work properly over the network, and to actually disable movment while recoiling. Continued work on player attacks.
12	November 7	1	Cheated on the player attacking, now it works! :D Implemented basic death, basic respawn, basic HP GUI. I'm starting to think I may be done with weeks 4-5 by the end of week 3. Huh. Still need to test it over the network.
13	November 8	2	This is the day when I put my foot in my mouth. Spent the day trying to re-jigger player attacking so it hits the hitbox, not the collision box
14	November 11	2	Worked out details of pipeline. Got ex2D into the project. Worked through tutorials and learned about the package. Adjusted player attacking and damage to work properly over networking. Need to work on clean, authoritative respawning
15	November 13	2	Adjusted player camera and player size to match background and sprite size. Made video for blog. Adjusted player hitboxes for new size. Fixed problems with SVN.
16	November 14	2	Fixed player attacking over the network. Started work on proper, timed, network respawning.
17	November 15	2	Imported a basic run animation to test the pipeline. Continued to debug respawning, encountering a strange bug where a list is not initialized/updated
18	November 18	2	Fixed respawning, both local and networked. Fixed a bug where client players could only attack to their right.
19	November 25	4	Added a script to smooth out the authoritative server to the clients. Extensively tested 12-player networking. Did various bugfixes.
20	November 27	5	Added attacking, walking, and idle animations. Fixed a serious networking bug. Re-sized sprites and re-jiggered the camera to handle 2d a little better. Started moving moving, parrying, and attacking to a single state variable.
21	November 29	2	Added health GUI. Added basic money GUI. Finished moving states to a single variable. Parrying animation plays, but logic is not in place.
22	December 1	4	Fixed networking bugs with the new state machine. Made the money gui less terrible. Made money pickups on death.
23	December 2	2	Fixed an animation bug where the same animation would re-start with new button presses. Added the rest of the level art. Imported a font. Adjusted player run speed.
24	December 6	2	Started to implement Capture Points, parrying mechanics, and loadout menus. All still need to be tested, but the lab was stanky and I couldn't grab a second computer.
25	Decembr 11	3	Added basic loadout screen. Set it so that the loadout screen pops up on death and goes away on respawn.
26	December 12	5	Fixed Capture points, they now function in multiplayer. Added score display. Added Victory conditions and popup. Added Map re-launching. Fixed the loadout screen such that it now scales with the user's screen resolution.
27	January 11	2	Adjusted the level colliders such that they won't interfere as much with players as they move (made them frictionless). Adjusted the loadout menu to actually show the arrow, and now it can be controlled via the keyboard. Adjusted collisions to work with both level and players (currently, players are disabled for suspicions of ruining networking).
28	January 12	2	Imported sounds, created scripts to handle audio.

Appendix G: Bug List

Screencaps of our buglist towards the end of development. We implemented a public buglist far too late into development. It would have made tracking and reporting bugs vastly easier in our early stages. We decided to prioritize bugs as high, medium, and low priority and color-code for easier reading.



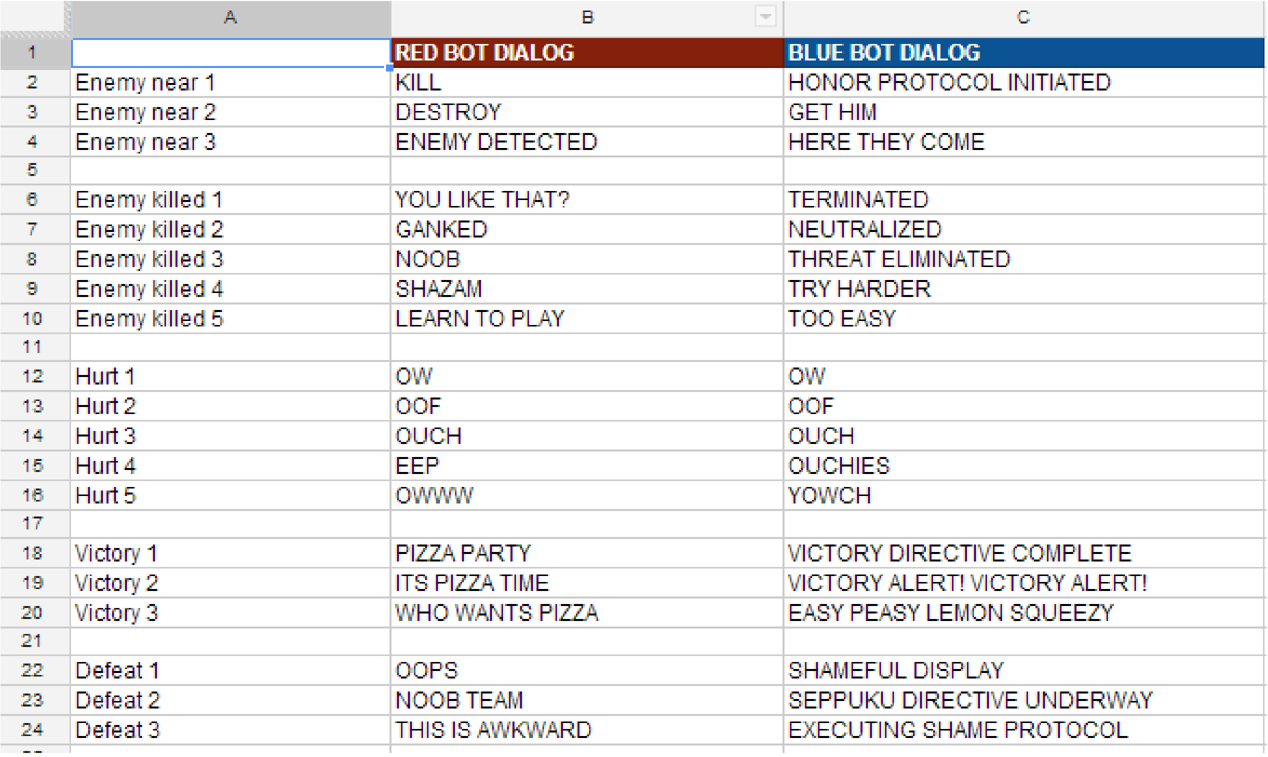
	A	B	C	D	E
1	**FIXED**				
2	**Bug**	**Description**	**Proposed fix**	**Priority**	**Status**
3	Enter key won't open how-to-play menu	You can only get into the how-to-play menu using the spacebar. Not consistent with other menu options		hi	FIXED
4		If I am facing left and go to run right, my bot will play his right-facing run animation and slide to the left. After I release the key, he will teleport to the right, presumably where he would have ended up naturally, if he had been running right in the first place. After the key is released I can run right normally, but I can ONLY run right normally. If I try to run left the problem will happen again, but replace all lefts with rights.		hi	FIXED
5		The problem above happens with up and down as well. If I had been running right and then I go to move up, I will continue to run right, then when the key is released I'll teleport to where I would have ended up normally (higher up and not forward at all).		hi	FIXED
6		switch game music to AMBIANCE.wav	please make sure its kinda quiet. music is very loud atm	hi	FIXED?
7		score is set to only 10	set it to 150 for now	hi	FIXED
8		all players spawn as Blue Team		hi	FIXED
9		right and left control points bug when fully capped -- they remain at the last tick and never finish. when you step off, they become full-neutral permanently		hi	FIXED
10		non-host (clientside) players not seeing arrows travel -- they stay in bow	it appears to function 100% correctly host-side	hi	FIXED
11		HUD shows while in armory (hp, ability cooldowns)	hide all the hud elements while in the armory	hi	FIXED
12	Double-Stealth Freeze	Activating the ninja's armor ability while stealthed freezes the character		hi	FIXED
13		add new katana special, die katana special, and die katana atk SFX		med	FIXED
14	Self-Destruct Gold	Earn gold by killing yourself in the pause menu.		med	FIXED
15	Persistent Stealth	Dying while invisible maintatins the invisibility for the next life.		med	FIXED
16	Client running	When the client runs for a long time and stops the bot continues to run for a little based on how long he ran.		med	NO LONGER PRESENT
17	Score counter	The score counter needs to be downsized a little bit, it's covering up the player's health	only on certain resolutions. ignore.	med	FIXED
18	Incorrect Gold Counter Position	The gold counter in the loadout menu is in the wrong position		low	FIXED
19		implement new cap point graphics	lemme know if theres any issues	low	FIXED
20	katana swing sfx bug	single katana swings play 2 swish sounds		low	FIXED
21		debug mode crashes game if playing in web player	why are you even using debug mode	low	NOT A BUG
22	**CHECK IF FIXED**				
23	rejoin char spawn bug	Every time player 2 (blue) rejoins the match, a red sword peasant also spawns for red, and player 1 (red's) camera is set to this new peasant, even though red does not control this new peasant (he's still controlling his original character off-screen).		hi	SHOULD BE FIXED
24	Goldfish	Goldfish doesn't work? Nobody ever stole a coin with it (and we tried)		hi	TEST NEEDED
25	Katana Swing	Client-side: Once you've double-swung the katana, you can no longer single-swing it		hi	TEST NEEDED
26	Die Katana can be parried	The Die Katana can be parried and when its parried it freezes the attack mid move and makes it unable to do damage on its next attack. After it's next attack it can do damage normally and be parried again.		hi	NEEDS TESTING
27	Parries	Looks like once an attack is parried the attack cannot hurt someone aftwerwards (for one single attack)		hi	TEST NEEDED
28					SHOULD BE



	A	B	C	D	E
30	Sensei armor ability	Zen healing only heals temmates and not the sensei (also remember that the healing effect should play at 17 fps)		med	TEST NEEDED
31		In name entry screen, feedback SFX plays if player selects a letter with ENTER but not if they use SPACEBAR	it should play for both	low	SHOULD BE FIXED
32		you can still scroll down to the 'quit button' in the web player even though the button is hidden		low	TEST NEEDED
33	blade rush too loud	please reduce the volume on the blade rush if possible. if not, let me know		low	TEST NEEDED
34	**REPORTED**				
35		client side players see other players 'flashing' in and out of existence	does not constantly happen. may be a latency issue.	hi	
36		client side players see other MOVING players' shadows disappear and then fly TOWARDS that player from the direction they are facing	does not constantly happen. may be a latency issue.	hi	
37	client-side camera jitter	Client-side player has camera (or maybe positional) jitter whenever they STOP moving. (camera snaps a bit backwards, its quite disorienting)	(JT: I'd like to see this in action. I think I know what this is, but I'd like to investigate personally)	med	
38		item costs now showing in armory	implement costs i emailed to you on 4/11/13, as well as writing the price under the item in the armory	med	
39		text in 'text entry' fields in join/host menus is too large for the field	reduce font size or reposition text entry box	med	
40	Incorrect Hit/Hurt Boxes	Move the hit and hurtboxes to be down by the robots' feet to better reflect weapon position		med	IN PROGRESS
41		disconnecting from game via 'leave game' button bugs menu so that buttons do not function		med	
42	Client bow	Blue peasant bow is really high up when he runs, might be my fault. Also, arrows don't always come out or seem to come out late on client's screen.		med	
43	Die Katana	Die Katana ability hitbox it is too far back, doesn't do enough damage		med	
44	Teleporting	Sometimes the client teleports backwards after running forward	learn 2 walk n00b	med	
45	gold for assists?	I don't believe players are getting gold for assists. No way for us to test whether assists are working at all, since scoreboard doesn't show an assist coutner	Test this out and come back to me.	med	
46		REMOVE default IP so that new players don't connect to non-existent host		low	
47		Capture point HUD element does not correctly display capture progress	the red/blue coloring 'starts to low' and 'finishes to high' on the HUD element	low	
48	waiting' camera bug	when waiting for second player to join, camera shows out-of-bounds area at left edge of map	make it show a menu screen until a second player joins. Eventually, this screen will show the game's controls	low	
49		on a keyboard, players should be able to enter their name just by typing it (pressing the letter K types a K in their name box)		low	
50		Game score shows in main menu after you leave match (it freezes, doesn't update based on current game score... idk if this helps)		low	
51		Leave Match doesn't work as host	Leave Match doesn't work for anybody on the network, I haven't had a chance to properly implement it. That's why I said not to test it	low	
52		When there is a networking hiccup between client and host, client is able to freely travel around the map, and is snapped back to his correct position only when connection is re-established	This is somewhat disorienting and frustrating for players. If there is any way to just hold the lagging client-side player in place if the game detects a connection hiccup, it might be better.	low	
53	Make sounds mono	Sounds flip from all-right to all-left.	make sounds mono	low	IN PROGRESS
54	Name Entry sound too loud	The SFX in the name entry screen are too loud compared to other audio		low	
		In blue katana peasant's idle animation, his arm jerks forward			

Appendix H: Robot Dialog

Dialog was performed by Wyatt Gray and Keegan Leitz. The audio was edited in Audacity by merging the dialog with a white-noise channel and combining using a vocoder to produce a robotic voice. All dialog can be seen below.



	A	B	C
1		RED BOT DIALOG	BLUE BOT DIALOG
2	Enemy near 1	KILL	HONOR PROTOCOL INITIATED
3	Enemy near 2	DESTROY	GET HIM
4	Enemy near 3	ENEMY DETECTED	HERE THEY COME
5			
6	Enemy killed 1	YOU LIKE THAT?	TERMINATED
7	Enemy killed 2	GANKED	NEUTRALIZED
8	Enemy killed 3	NOOB	THREAT ELIMINATED
9	Enemy killed 4	SHAZAM	TRY HARDER
10	Enemy killed 5	LEARN TO PLAY	TOO EASY
11			
12	Hurt 1	OW	OW
13	Hurt 2	OOF	OOF
14	Hurt 3	OUCH	OUCH
15	Hurt 4	EEP	OUCHIES
16	Hurt 5	OWWW	YOWCH
17			
18	Victory 1	PIZZA PARTY	VICTORY DIRECTIVE COMPLETE
19	Victory 2	ITS PIZZA TIME	VICTORY ALERT! VICTORY ALERT!
20	Victory 3	WHO WANTS PIZZA	EASY PEASY LEMON SQUEEZY
21			
22	Defeat 1	OOPS	SHAMEFUL DISPLAY
23	Defeat 2	NOOB TEAM	SEPPUKU DIRECTIVE UNDERWAY
24	Defeat 3	THIS IS AWKWARD	EXECUTING SHAME PROTOCOL