Batastrophe

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfilment of the requirements for the
degree of Bachelor of Science

by
Aidan Buffum
Gavin Taylor
Isaac Donkoh-Halm
Marc McFatter
Matthew Figuerora

Advised by:
Farley Chery
Gillian Smith

Worcester Polytechnic Institute

Abstract

Batastrophe is a four-player, split-screen, asymmetrical party game made in the Unity engine, utilizing four standard gamepads. One player is a bat, flying through the environment, while the other three players are humans trying to catch and remove the bat player from their house. The goal of the bat player is to damage as much of the environment as possible, while the goal of the human players is to eliminate the bat player quickly, keeping destruction to a minimum. Players take turns controlling the bat, competing to reach the highest score across four rounds. Each player has one turn as the bat and three as a character pursuing the bat.

Acknowledgments

We would like to thank our advisors Professor Farley Chery and Professor Gillian Smith for their work as our project advisors.

We would also like to take a moment to thank the playtesters for their time and feedback which was used to further improve the game.

Table of Contents

List of Figures

Table of Authorship

Chapter	Primary Author	Secondary Author(s)
1. Introduction	Isaac Donkoh-Halm	Gavin Taylor
2. Background	Isaac Donkoh-Halm	Marc McFatter
3. Gameplay Design	Isaac Donkoh-Halm, Aidan Buffum, Marc McFatter	
4. Artistic and Visual Design	Matthew Figueroa, Gavin Taylor	
5. Technical Design	Aidan Buffum, Marc McFatter	
6. Production Techniques	Isaac Donkoh-Halm	
7. Testing	Isaac Donkoh-Halm	Marc McFatter
8. Post Mortem	Aidan Buffum, Gavin Taylor, Isaac Donkoh-Halm, Matthew Figueroa, Marc McFatter	

1. Introduction

Batastrophe is a multiplayer cooperative party video game where three players work together to stop one player, a bat, from destroying their house. Featuring a playful and whimsical visual design inspired by cartoons such as *Looney Tunes* and *Tom and Jerry*, Batastrophe aims to be a fun and exciting game that's easy to play and accessible to a wide audience of gamers. In order to achieve this goal, we researched the main motivators of different types of gamers, and designed elements of the game to have features that are attractive to certain people, including control schemes and general gameplay flow.

The game takes place inside an upscale suburban home, where three children are left home alone. Suddenly, a wild bat enters through an open window and decides to create mischief by flying around the house and destroying everything in its path. To protect their home, the children arm themselves with improvised weapons from pots, brooms, and tennis rackets, and make a plan to get the bat out of the house while minimizing damage. Working together, they must locate the bat, stun it, and send it flying out of the house with no means of reentry.

The team is five students; Aidan Buffum and Marc McFatter were focused on the technical aspects, while Gavin Taylor, Matthew Figuerora and Isaac Donkoh-Halm focused on the visual aspects of the game. This group also had a student working in a producer-type role, and so we had the additional challenge of trying to establish a functioning team dynamic that would best make use of having a student acting as a producer.

Lessons learned throughout the process of developing this game include basic problem solving skills. Towards to the end, learning to prioritize was high on the list. It was a team effort to figure out what we needed first to get the progress done. When we encountered issues with one approach, there was always an alternative way of solving the same problems. Similarly, we have all learned how to adapt to situations on the fly. An example of this was when the game became inoperable due to changes made shortly before our presentation at Alpha Fest and our tech team had to work quickly to find a solution in time. Working with each other overall was a learning curve itself. Some teams don’t have the luxury of befriending one another, so we’ve taught ourselves to put differences aside to come together for our common goal.

This report chronicles the development, testing, and eventual release of Batastrophe. The subjects that will be touched upon include design (visual and technical) production techniques that kept the development moving, and the testing that was done to refine what makes the game work.

2. Background

2.1 Platform and Technology

Batastrophe was created using the Unity3D Game Engine, and the final build of the game was released for Windows PCs requiring four standard USB or wireless game controllers to allow everyone to play. Unity was chosen due its ease of development to most of the team.

2.1.1: VR

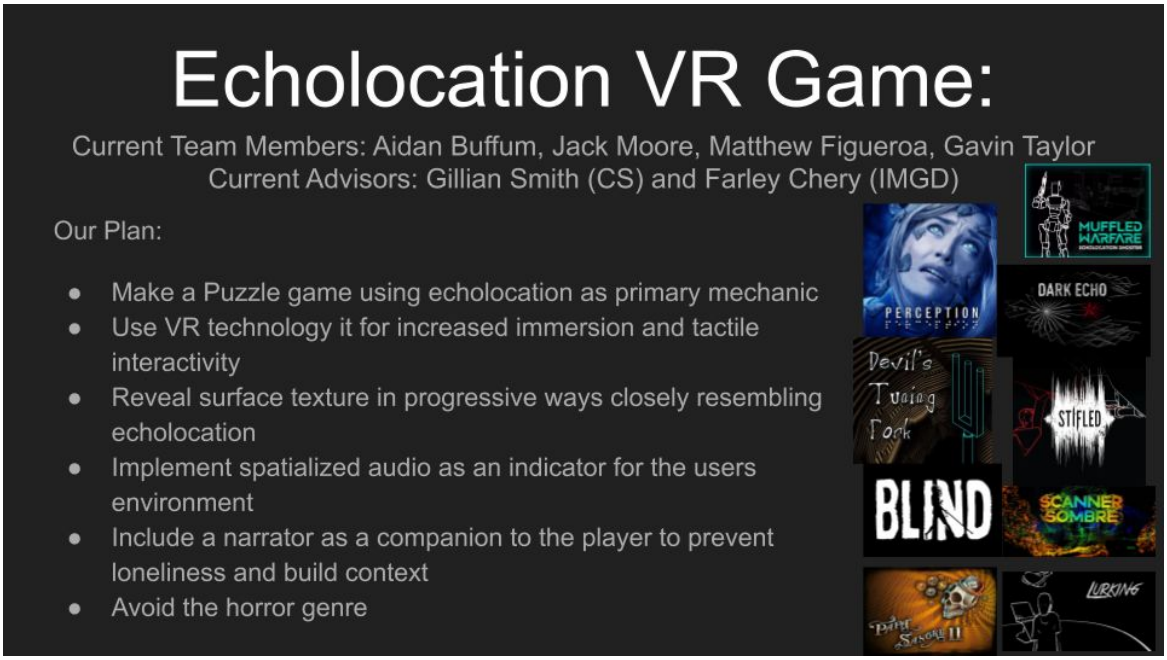

Figure 1: Original MQP Pitch Slide from 2019

This project had origins of being released for virtual reality (VR) platforms where the human players (herein addressed as "hunters") would play using controllers while the bat player would control themselves using VR, utilizing motion control technology to make the player use body motions and head movements to control a small animal flying through the air. The team eventually abandoned the VR feature, citing complexities of developing a VR game, and concerns that the player would get motion sickness flying. Whereas most other VR flight games we examined, ours featured a small player with distorted vision flying through confined spaces and intentionally crashing into other objects. There were other accessibility issues that made VR less of an appealing option to the team, such as how one of the programmers and project advisors are unable to play VR games, and the risk of cybersickness affecting players differently based on

gender, with females being more susceptible (Linda Foster, 2020). While it would have been exciting to have the player use their body to fly, becoming even more immersed in the feeling of being a bat, we felt that the planned features would have impaired many players' abilities to play the game, so we made the decision to abandon the use of VR.

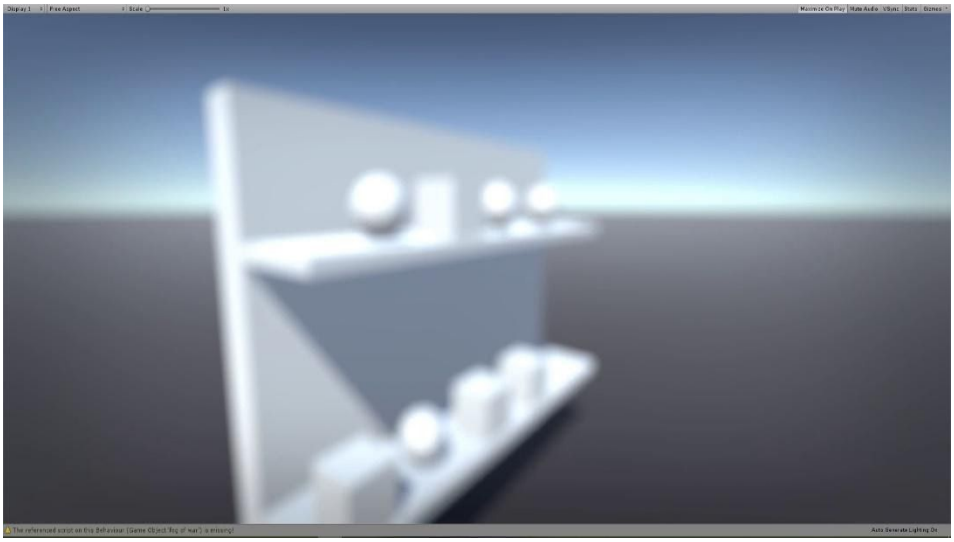

Figure 2: Deprecated Blurred Vision for the Bat. Objects would become clearer as they echolocate. The concept of having your vision mostly blurred while flying in VR, a platform that has more risks for cybersickness compared to a traditional flat screen (Linda Foster, 2020), fueled our decision to steer away from the VR implementation.

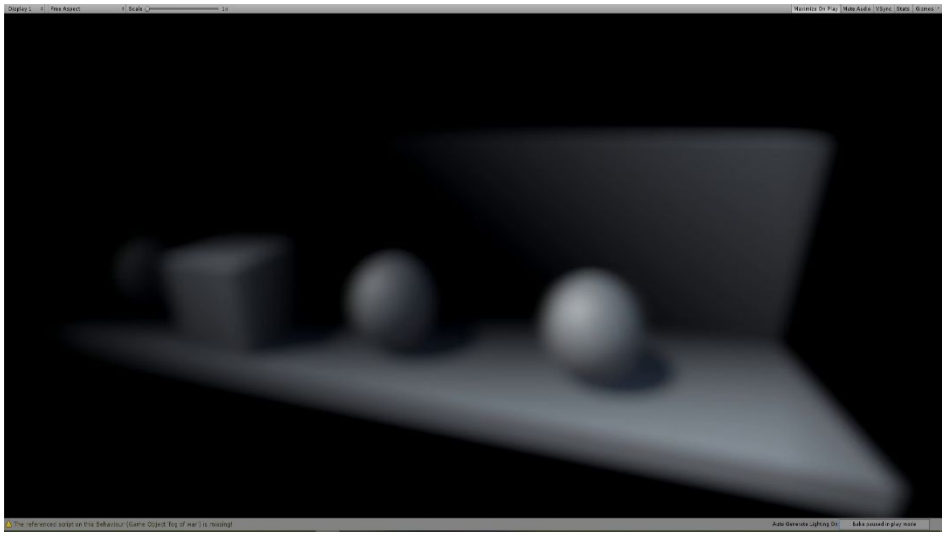

Figure 3: 2nd Iteration of Deprecated Blurred Vision for the Bat, the entirety of the bat's vision is obscured and blurred and can be enhanced by moving closer or echolocating.

2.2 Inspirations

Batastrophe shares its concept of an animal destroying a home with another game, *Catlateral Damage* developed by Manekoware. In *Catlateral Damage,* you play as a cat who must complete objectives surrounding destroying a home, unlocking new methods and abilities while doing so. Also developed in Unity, the game uses a similar physics system to move objects throughout the room, allowing the player to revel in the mess that they made.

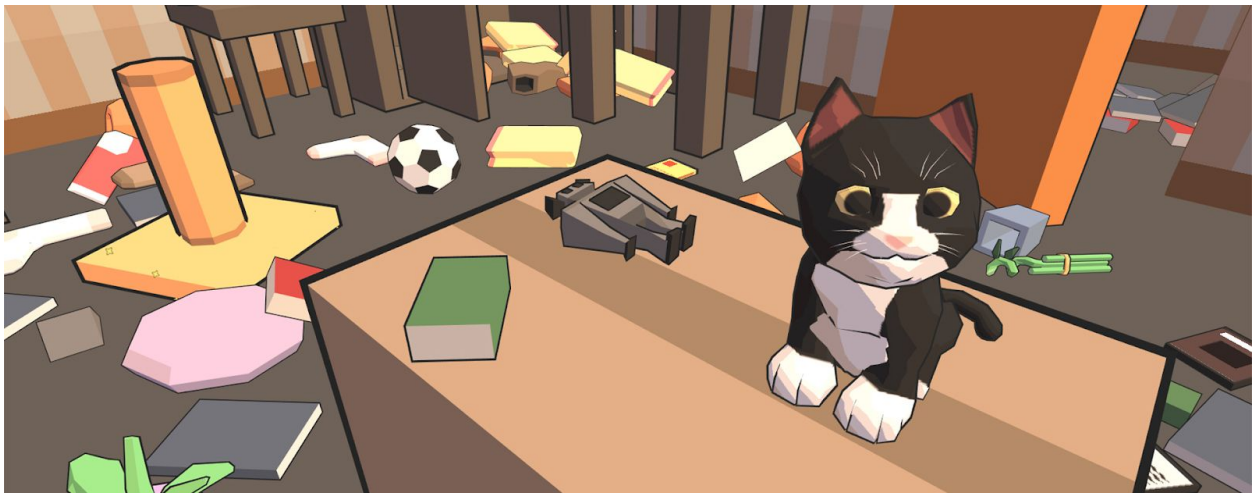

Figure 4: Press Release Image of Catlateral Damage (2015), developed by Manekoware.

Other video games that have inspired the concept or visual style of *Batastrophe* include *Octodad,* which eschews complex texture detail for its props and focuses more on the readability of them so players can identify which props can be interacted with.

Figure 5: Screenshot of Octodad

The general art style and concept is heavily inspired by slapstick comedy cartoons of the 1940s and 1950s such as *The Looney Tunes* and *Tom and Jerry*. Muted and low contrasting colors were used for the walls in order to make the props stand out and be more noticeable to the players, similar to how backgrounds were created for these cartoons to bring more focus to the animated action.

Figure 6: Animation Backgrounds illustrated by Robert Gentle. Note the difference in value of the door compared to the rest of the house. From Tom and Jerry *film* Jerry and The Goldfish*, (Barbera, 1951)*

Codename: Kids Next Door was a huge inspiration in how we wanted to create our characters. The idea of grabbing any household items and making them into weapons and armor was a fun fantasy for us to play with. Our characters will be dressed in pots and pans for head

gear, oven mitts for gloves, and a kitchen apron as armor. This fits our aesthetic of quickly grabbing what's accessible to you to protect yourself.

Figure 7 Ice Gun prop from Codename: Kids Next Door. *Prop designs by Robert Kopecky*

Figure 8: Numbuh 86 from Codename: Kids Next Door. Character Designs by Tom Warburton

2.2.1 Game Feel and Motivations

The design of the game was heavily influenced on the gameplay styles that we researched prior to beginning the project. We used the findings from market research company Quantic Dream to determine the main motivators behind the design of our favorite games. (Yee, 2019)

Figure 9: Quantic Dream Gamer Motivation Chart

Gamer Motivation Chart from Quantic Foundry.

We took Batastrophe's primary goals and applied it to the motivators illustrated by Quantic Foundry. Since the player's goals vary between the bat and the hunters, we split up the table to illustrate the motivations of each respective playstyle.

Table 1: Gamer Motivation Chart in relation to different gameplay styles in Batastrophe

	Action	Social	Mastery	Achievement	Immersion	Creativity
Bat's Motivation	Destruction, Excitement	N/A	N/A	N/A	Fantasy	N/A
Hunter's Motivation	Excitement	Community	Strategy	N/A	N/A	N/A

In *Figure 9*, we determined that the single motivator that was shared between bat and hunter players is action. The hunter's unique motivations are social and mastery, while the bat's unique motivation is immersion.

2.2.1.1 The Bat Player's Motivation

The bat player's gameplay motivation is primarily related to immersion, and specifically pertains to the Fantasy sub-motivator. While not as detailed as other games in this category, such as Counter Strike or Street Fighter, players will begin to feel connected to the bat's lust for destruction and mischief once they knock some items down and successfully evade the other human players.

2.2.1.2 The Hunter Player's Motivation

The hunter player's gameplay motivation categories are social and mastery, and the primary motivators are strategy and communication. The strategy motivator applies to the hunters, even during high paced and hectic moments of gameplay, players must make quick judgments and decisions as a team if they want to have a chance to capture the bat. During play, it is extremely beneficial for the hunters to verbally communicate with each other to plan to corner, capture, and trap the bat outside.

3. Gameplay Design

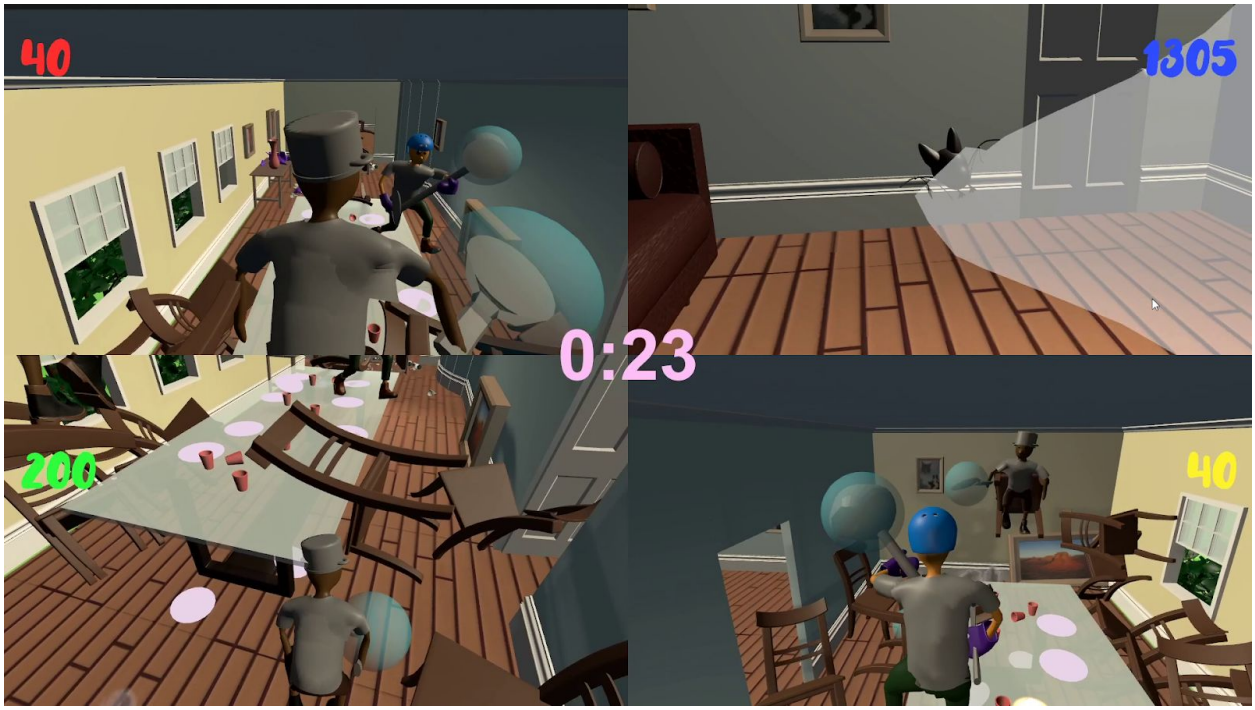

Figure 10: Screenshot of split-screen gameplay in Batastrophe.

The goal of the game is to earn as many points as possible throughout 4 rounds of gameplay, with a different player controlling the bat each round. The bat earns points by smashing into objects and evading capture by the hunters before the time limit expires, while the hunters earn points for working together to first stun the bat, throw it outside of the play area, and close all entrances to prevent it from returning.

3.1 Game Mechanics

3.1.1 Bat gameplay

Figure 11: Screenshot of bat flying around from Batastrophe.

The bat was designed to satisfy players whose primary motivations for playing games include destruction and excitement (see Table 1: Gamer Motivation Chart in relation to different gameplay styles in *Batastrophe* and Figure 9: Quantic Dream Gamer Motivation Chart.) Satisfying a player's sense of immersion was also an experience goal we were aiming to achieve, and as the bat, the player can fly around a house to knock over items that appear to be large compared to the bat. Players can ascend and descend freely in order to reach certain props, such as vases on tables or glasses on top of cabinets. The bat is constantly moving in the direction they are facing and may only stop by flying up and perching on the ceiling.

Figure 12: The Bat approaches the ceiling and a prompt asks the player if they want to perch on the ceiling to observe the room.

From here, the player remains in the place they landed, but is free to look around to survey their surroundings. This was done to add an element of strategy to a hectic playstyle, giving the bat player a chance to pause and quickly plan out their method of destruction.

3.1.2 Hunter Gameplay

Figure 13: Screenshot of a hunter navigating the environment in Batastrophe.

The hunter playstyle is like that of the bat in which it was also designed to excite the player, but what separates the bat from hunters is the focus on the element of cooperation and strategy. Each of the hunters have different levels of speed, reach, and attack, so players can decide which role they are more suited to play based on their character preference.

There are three types of character classes available: the racketeer, the sweeper, and the trapper. The racketeer has the highest mobility of the three, able to move between rooms quickly and can squeeze through tighter spaces than the others. The racketeer also has the highest knockback when swinging their weapon, stunning the bat while also flinging it across the room. To balance this, they have the shortest swinging reach in the game, which makes it difficult to stun the bat without being right in its face. The sweeper is the opposite of the trapper, having the lowest mobility but the longest reach. The trapper is a bit more average in comparison to the two, but they have the special ability of instantly capturing the bat in their net instead of having to stun and pick up the bat.

Figure 14: The Bat is caught and stunned by the Trapper.

Hunter players can successfully outwit the bat player by way of communicating with each other and setting a plan. Once a bat is successfully stunned and removed from the interior of the house (see 3.1.4 Windows and Doors), players must seal all the windows and doors to prevent the bat from entering the play area, a task that can be easily completed by having the players split up and spread out to work on separate sections of the house.

3.1.3 Props

Props are one of the core mechanics of *Batastrophe,* the hunter players must carefully avoid them while the bat players aim to destroy the props in order to gain points. The props in

the house are spread out and plentiful in each room, giving the bat an incentive to travel to all the rooms in the house in order to maximize their potential score.

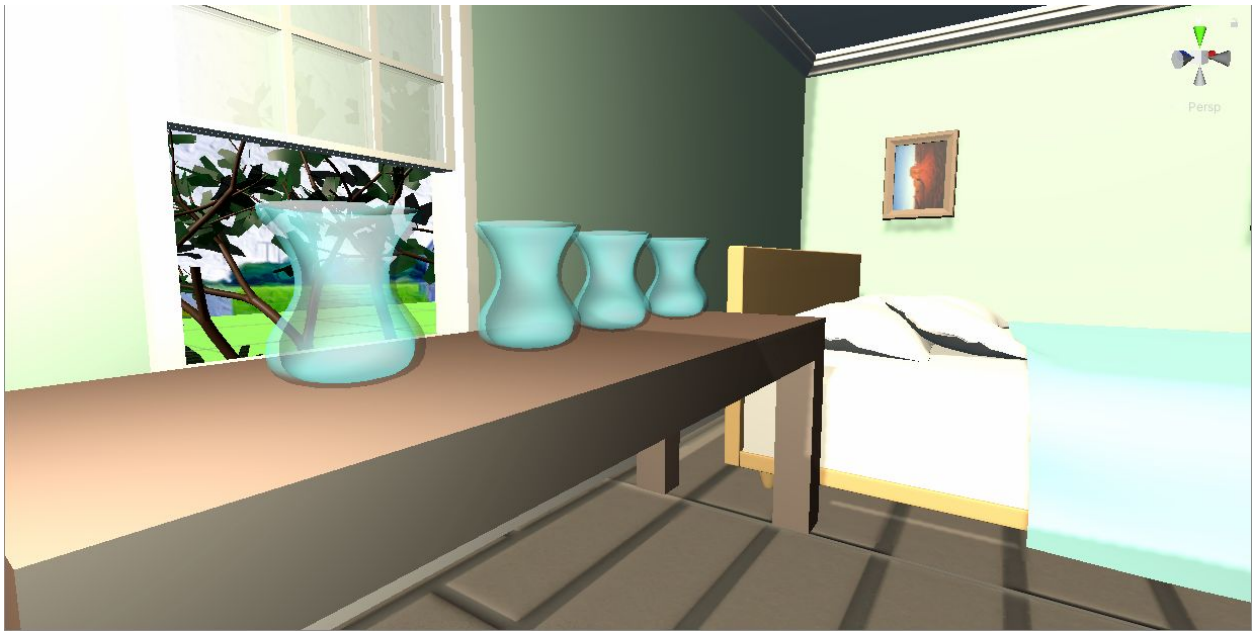

Figure 15:Glass Jars, Bedroom, 2F

Some props can be shattered by the bat if the bat flies directly into it at a certain speed or if it falls from a high distance. It can also be shattered by the player when struck by the hunters' weapons or if stepped on by a hunter. Fragments of a shattered prop on the floor will severely reduce the speed of any hunter player who attempts to walk over them. Utilizing this mechanic gives the bat a chance to directly impede on the hunter's ability to capture them

Figure 16: Shattered Glass Jars, Bedroom, 2F

3.1.4 Windows and Doors

Bats can fly through any open door or window, but the hunters are the only players who can directly interact with them by opening or closing them. By closing doors and windows, the hunters can work together to trap the bat inside a room or outside of the house entirely. This can be useful in order to limit the movement of the bat player and makes it easier for multiple players to capture the bat in one single enclosed area. The windows are also directly tied to the hunter's win condition; trapping the bat outside of the house and preventing it from coming back in.

3.2 Map Design

3.2.1 Floor Planning

We used the online architecture tool, FloorPlanner.com, to design each iteration of the game's setting. The tool gave us the ability to design our house in a seamless, interactive manner, providing a 3D View Mode that allowed us to see how the 2D floor plan translates to 3D and had options for us to control the dimensions of walls, ceilings, and square footage of individual rooms.

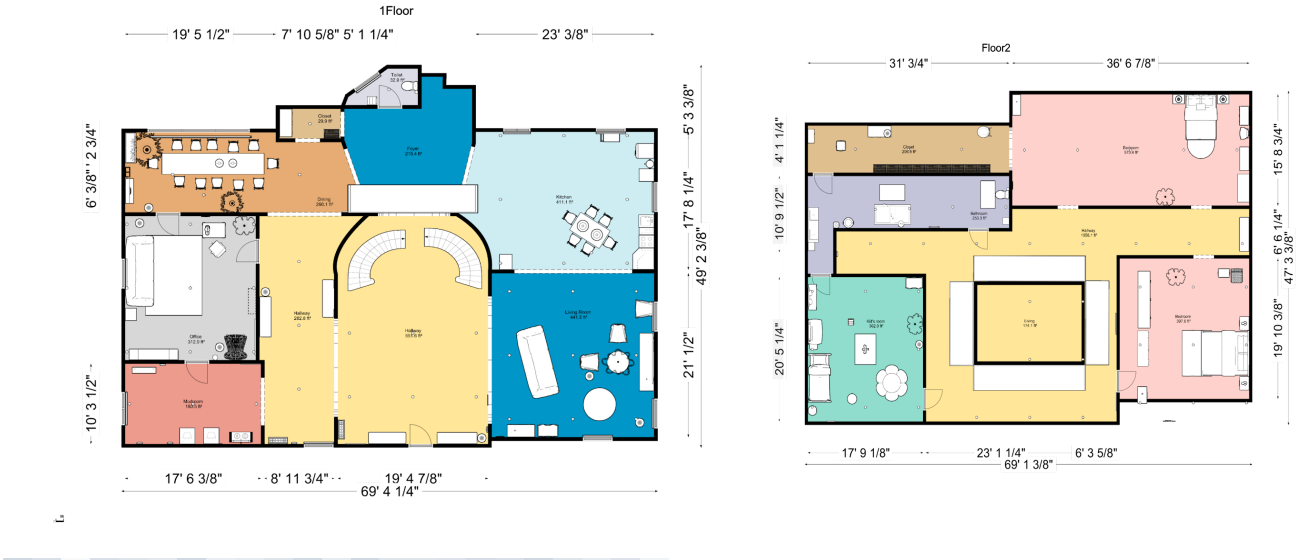

Figure 17: Initial Floor plan from FloorPlanner.com.

Figure 18: 1st Floor 3D View from Floorplanner.com

Before settling on a final interior design, we created multiple greybox versions of the house. Greyboxing is the technique of using simple three dimensional shapes that represent potential layouts and prop placement of a level before creating more detailed assets to replace them (Casen, "White Boxing Your Game"). We were able to quickly design and test new iterations of the house using this method without having to make major changes to any art assets that may need to be adjusted due to design changes.

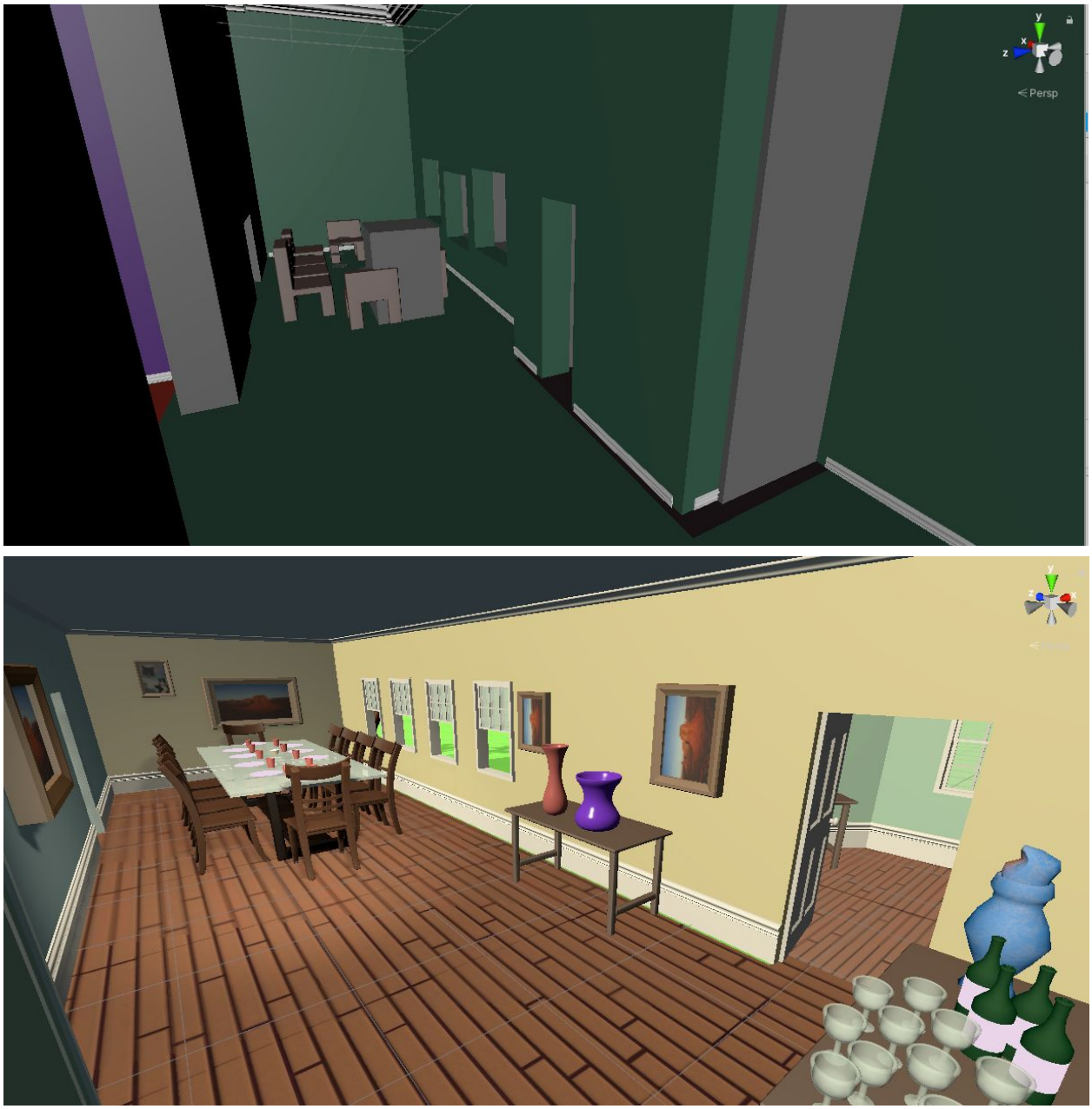

Figure 19: Comparison of graybox to final room design.

Figure x shows a gray boxed version of the first floor kitchen, while figure y shows the final result. When testing with the greybox level, we realized that the ceilings were too high, the doors were too narrow, and the chairs were too close to the office entrance. We made adjustments to the ceiling so that the bat couldn't just simply knock a few props down and then fly to a height that no hunter can reach until the time runs. We also made it easier for multiple hunters to walk through the doors and room without having to knock over too many things. Afterwards, we replaced the simple meshes with the final versions of them once we made any other additional adjustments.

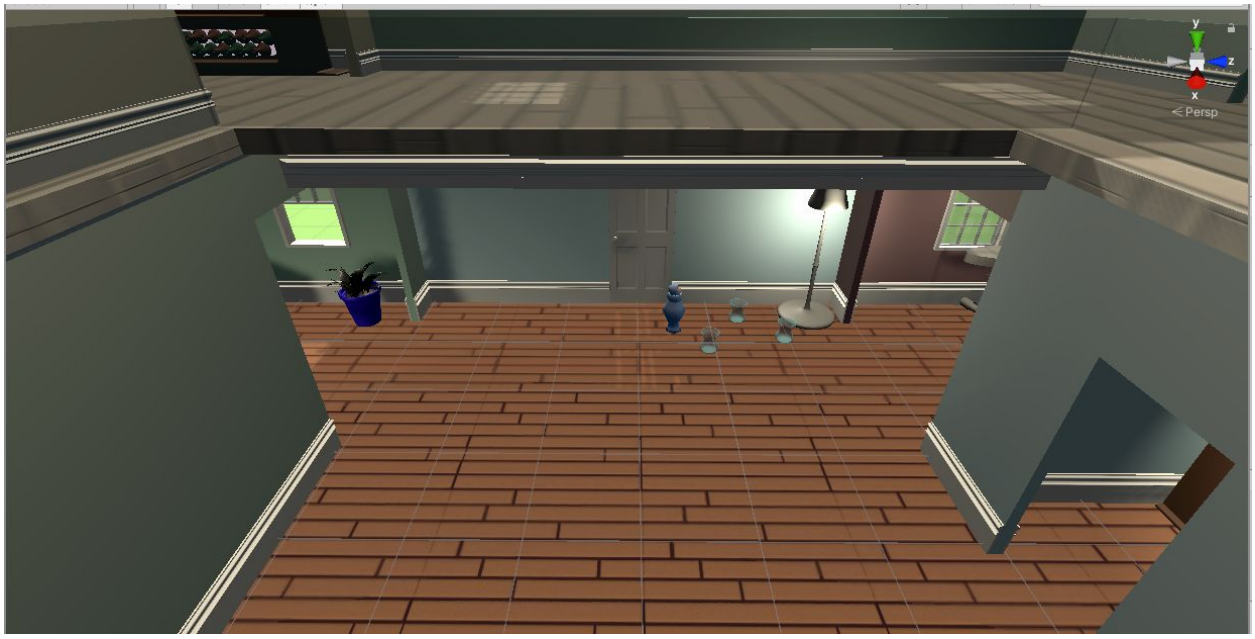

Figure 20: View from second floor balcony.

3.2.1.1 Room Breakdown: 1st Floor

Figure 21: 1st Floor Aerial, Isometric View

The rooms on the 1st floor include the foyer, the laundry room, the kitchen, the office, the living room, the dining room, and a bathroom.

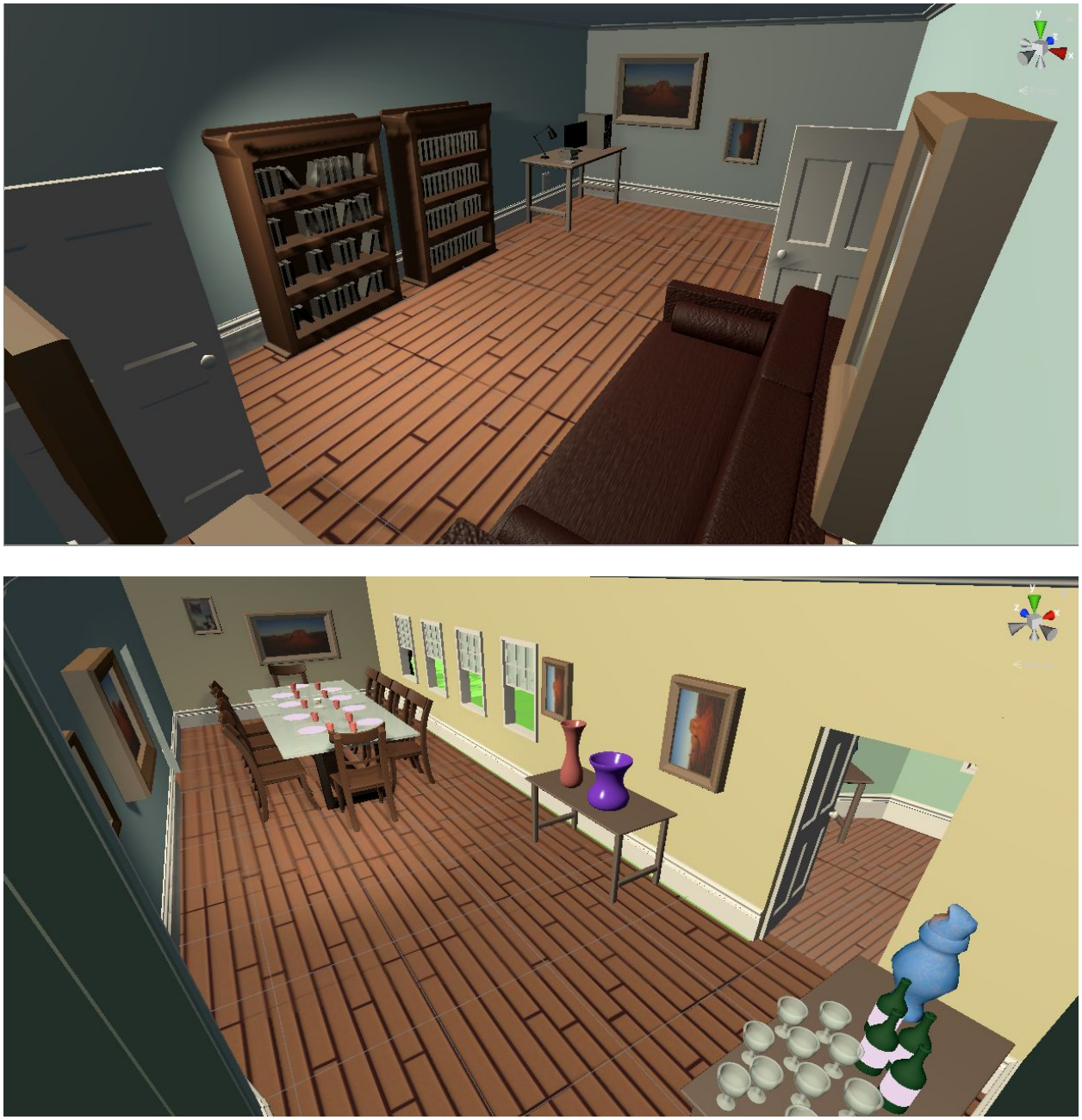

Figure 22: Office and Dining Room, 1F

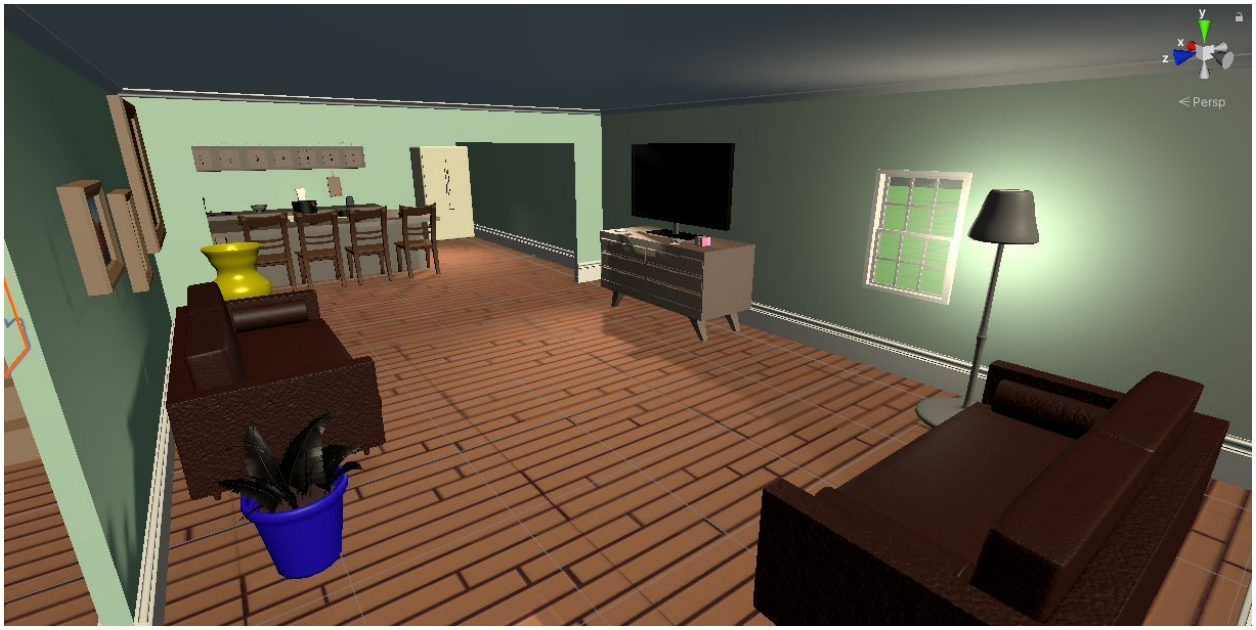

Figure 23: Living Room and Kitchen, 1F

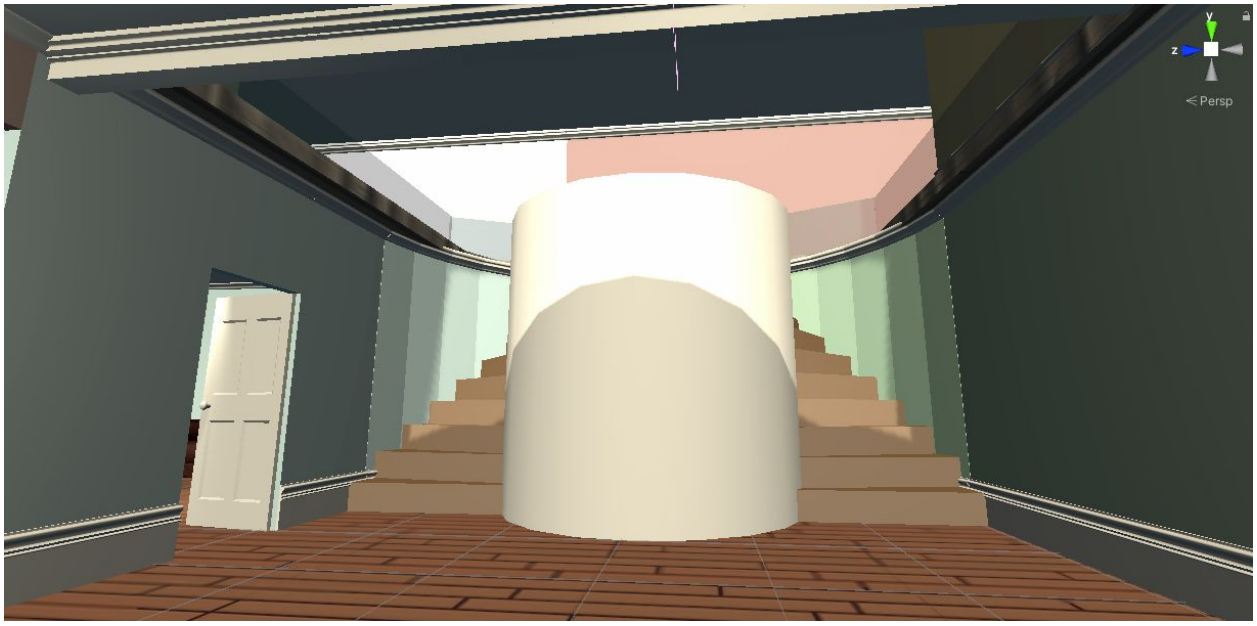

Figure 24:Foyer, 1F

3.2.1.2 Room Breakdown: 2nd Floor

The rooms on the 2nd floor include 2 bedrooms, a bathroom, and a wine room.

Figure 25: 2nd floor Aerial, Isometric View

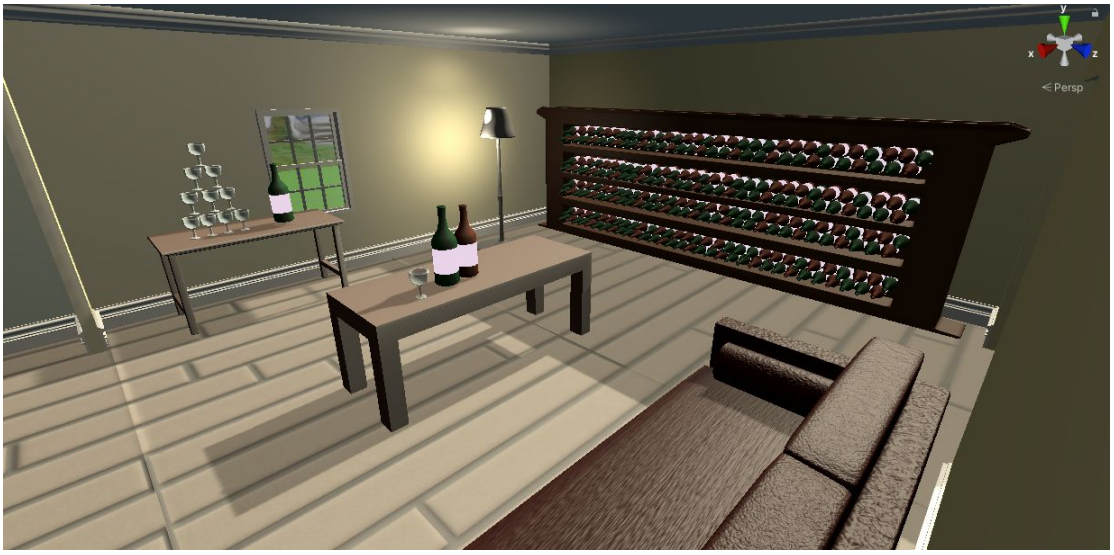

Figure 26: Wine Room, 2F

3.3 Player Controls

Technical design in this project was centralized from the beginning around the ease of player use. Since this is a party game designed around a chaotic atmosphere, our design was focused on features effortlessly working for the player to minimize frustration and maximize enjoyment. The less the player has to fight against the controls, the more they are able to focus on the other players, hitting the bat and avoiding the hunters.

3.3.1 Bat Controls

As the main gameplay for the bat, designing intuitive and consistent flight is paramount to how enjoyable playing as the bat can be. The controls should be easy to understand quickly (intuitive) and behave the way they are expected to every time (consistent). We referenced a number of games involving flight, looking to extract the movement schemes which fit the best for our game: natural flight, making sharp turns, in a closed space.

While in flight, the bat maintains forward motion facing away from the camera. Using the thumbstick, the bat can turn, moving the camera with it and banking wider turns as its speed increases. Speed is toggled between a regular and dashing speed based on whether the A button is held. The bat's altitude is controlled by trigger inputs, ascending and descending at fixed speeds when the right and left triggers are held respectively. When no input is provided, the bat will fall, accelerating downward. If the bat is dashing, it will not fall.

As the bat approaches the ceiling of a room, it may also choose to land. This provides a safe pause for the bat where it can look around and reorient itself while being difficult for the hunters to reach. In order to determine whether it can land, the bat compares its altitude with the nearest object over its head. If that object is the ceiling and the bat is close enough, it may land. Once landed, the bat can turn around on the ceiling and will take off in that new direction. Though the bat is still vulnerable to the hunters while on the ceiling, it is only accessible to the sweeper without climbing onto counters and knocking over furniture.

For control in the xz plane parallel to the ground, we explored three movement schemes including camera-independent planar movement, camera-dependent planar movement, and a steered always-forward approach. In order to test the controls, several small builds were made with slightly different control schemes. Feedback on each of these builds was gathered informally from the members of the team, the advisors, and randomly solicited members of the WPI community. In testing, the majority of players felt that planar movement offered the finest level of control but felt least natural. When dependent on the camera, it felt better but potentially allowed the bat to drift out of frame. The steered approach, which was received most positively, is the method we chose.

For altitude, we considered steering up and down, maintaining altitude and reading up / down buttons, and flapping to gain or maintain altitude. Steering felt the most mechanical and reminiscent of a plane flight simulator than an animal and likewise presented a number of problems with orientation despite offering the most explicit spatial control. The other two options both had interesting trade-offs to consider. Maintaining constant altitude felt like less of a chore but flapping felt like a more natural movement scheme. The final mode we decided on for altitude management was a flapping system that maintained constant altitude if the bat was going fast enough.

3.3.2 Hunter Controls

The control layout for the hunter characters was inspired by various first and third person action games such as *Halo 3* and *Gears of War.* Hunters may move grounded on the x-z plane with their thumbstick and jump a short height at the push of a button. Planar movement for each hunter is dependent on the orientation of the camera. Players may look up, down, left, and right, steering the "forward" direction of the player based on the left and right angle the player's camera is facing. Hunters each have a base movement speed depending on their class, potentially slowed or sped up based on environmental factors such as a pile of broken glass which the hunter must move slower and more carefully through. Each hunter similarly has a jump height associated with their class but will automatically walk over small enough obstacles. A physical size difference is also associated with each class and the racketeer, for instance, can move through a smaller opening than the sweeper.

Hunters may make one of two attacks, a vertical strike and a horizontal strike. These attacks are triggered independently on a class-dependent cooldown. Racketeer and sweeper attacks will apply a noticeable physical force on any objects in the weapon's path while creating an invisible hitbox along the path for the bat to interact with. Trapper attacks on the other hand will apply a far weaker force and will only capture the bat if the net directly connects with the bat.

3.4 Scoring

Nearly every interaction the players have with their environment and each other has an associated score value. There exists, however, the problem that not every interaction should be worth the same points. Knocking over a broom hardly feels as impactful as breaking a wine glass, but both pale in comparison to obliterating a vase of grandma's ashes. Similarly, the score a player earns as a bat should be greater than as a hunter, but the points they earn as a hunter should still feel meaningful.

A bat will be assigned points for any object it crashes into. If the bat hits that object as a result of being launched by the racketeer, it receives a reduced point value. If any object which a bat hits, then collides hard enough with another object to break it, the bat is also assigned full points for that object. Otherwise, the bat receives no extra points without individually hitting that object or restarting the chain reaction with more force. The bat will also be assigned a small number of points for colliding with and stunning players. This is a lower-value endeavor than smashing props, but rewards alternative strategies for playing around the hunters.

If a hunter knocks something over or breaks an object, that hunter will lose points related to the value of that object. Once broken, that object will not be redeemable by the bat for points. It is in the hunter's best interest not to wreak destruction on the house, but the bat player will not be rewarded for things they do not break. Hunters will not lose points for launching a bat into

objects, but will be put back by the positive points it yields for the bat. If a hunter successfully strikes the bat player however, they will be assigned a number of points. Likewise, hunters will be assigned points for tossing the bat out the window and all hunters will receive a large sum of points if the round is ended early by trapping the bat outside.

Final scores at the end of the game are measured by ranking each player's performance as the bat and their cumulative performance as a hunter. The winners of each game are identified as the best bat, best hunter, and best score overall, while being able to show individual round performance for each player. Keeping scores separate for bat versus hunter play keeps players from getting discouraged by the bat player's highest accumulation of points while keeping score largely independent across rounds makes it clearer to players that they can catch up in score to anyone ahead of them.

3.5 Menu Interfaces

The menus in *Batastrophe* are split into two Unity scenes: The main menu, and the character select menu. The main menu features access to player and system settings as well as the access to the game itself. Once entering the game, the players are sent into the second menu, that being the "character select" menu, wherein the player characters get the chance to choose their preferred class for any round of the game. The players are also sent back to the character select screen after every round, until the game is done.

Menus are designed around visual simplicity, clarity, and ease of use. To match our design fitting a party game, we wanted players to start and enter a game as fast and as easily as possible. This is why there is also focus to allow controller input in the menu systems to allow players to start and play the game without touching a keyboard, using the inputs from their controllers.

The main menu gives access to options of game resolution, audio settings for different audio sources, and player-dependent control settings. Each player may find easier controls when different sensitivity settings are chosen, so those are options for each player's bat and human controls. These player-specific menu screens are placed in each player's designated corner of the screen, to familiarize themselves with where their vision will lie during the course of the game.

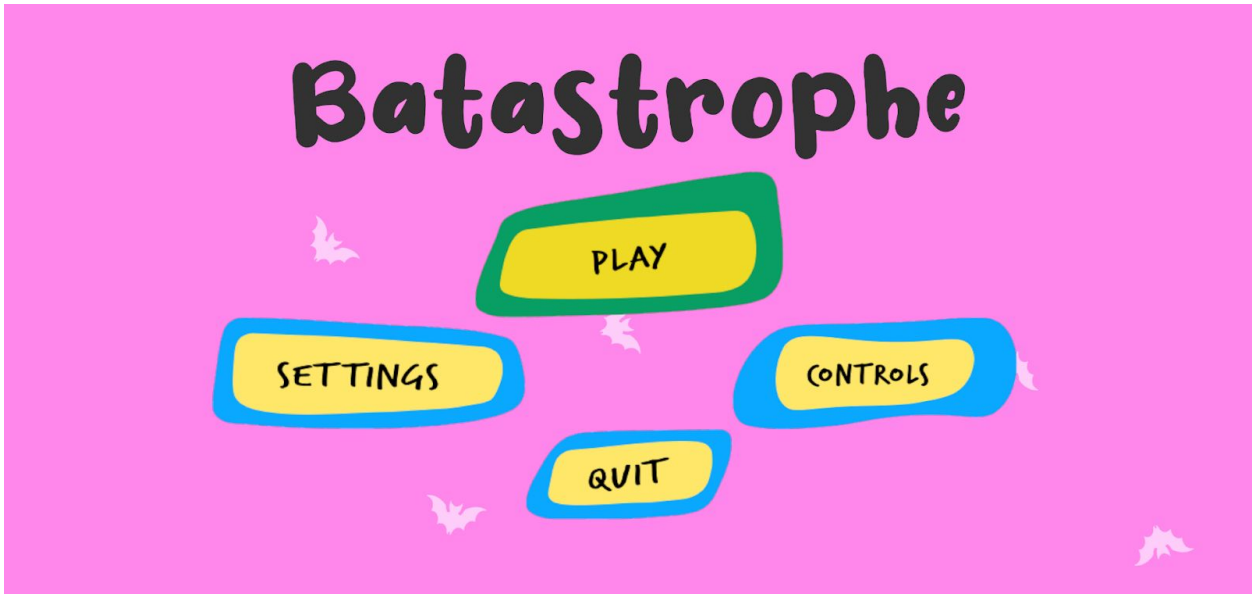

Figure 27: Main menu screen.

Upon the start of the character select screen, a random player (who has not yet played as the bat in this game) is selected to be the bat. Their personal screen is visually changed to clarify to them what character they are and what their role is in the game. The other players are given their opportunity to pick any of the three human classes.

The classes are varied in abilities and gameplay styles, so the menu is designed for each player to see the strengths and weaknesses of each class. A simple visual cue directs users to move left and right to choose between the different classes, and upon pressing the confirm button locks that class in. They can back out whenever they please, but once either all players confirm they are ready, or one person manually clicks the "Start" button, the game begins and the hunt for the bat is on.

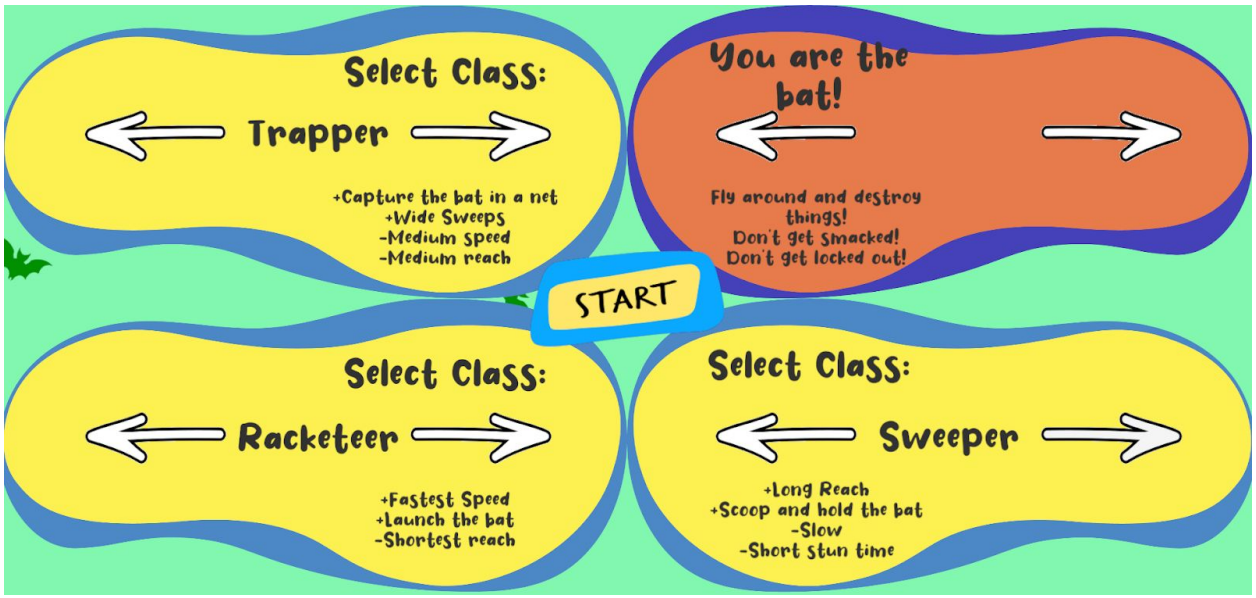

Figure 28: Character select screen.

4. Artistic and Visual Design

4.1 House Design

The play level is the main way that the gameplay design and art design combine. We strove to create a believable, coherently-designed, and easily navigable household. We designed every room to be distinct from one another while also using the layout of the house to give advantageous positions to the hunters and the bat. We also aimed to have as many believable household items around as possible for the bat to always have plenty of things to break. When all of these intents are stacked on top of one another, creating the house became a core point of merging gameplay and art design together.

Our design choice for the house is highly influenced by large luxurious homes with open concepts and spacious rooms to give the hunters more room to walk around in and the bat the opportunity to smash a lot of expensive objects. Important and costly objects that people do not want destroyed is a good motivator for chaos. When expensive things are broken it lowers the value of that object because it puts it in the same category as other broken objects that weren't as expensive to begin with. The concept with valuables is that they are made with higher quality. We did a fair amount of research into McMansions because they are known for appearing expensive. There were multiple trial levels that changed due to the play area being difficult to navigate. We experimented with different ceiling heights, door and hallway widths, and on turn sharpness.

Figure 29: Botero Custom Home's French Manor IV (Botero Homes, 2013).

We came together for a specific style to stick to and we chose urban-modern with a touch of shabby chic. Shabby chic is described as an eclectic style that consists of distressed wooden furniture, shiny and polished metals, and soft pastel tones (Mastroeni, Defining a Style Series:

What is Shabby Chic Design?). The style originated in the 1980's, when opulence and decadence was important to everyone (Simmons, What Is Shabby Chic? Everything You Need To Know). The amount of curvature seen in home pieces will determine if pieces are considered shabby chic. As for urban-modern, it's about having home accessories that stand out, but are still clean in design. The furniture is typically attractive, but not too loud in appearance, with the addition of more muted colors (Decor Aid, Urban Modern Interior Design Defined: Everything To Know). It's less furniture and opens up more space for the players.

Figure 30: Photo of shabby chic interior. Courtesy of hqdesigns.

Figure 31: Photo of urban modern interior (Decor Aid, 2020).

The modern look provides more open space for the bat player to fly through and provides big floor plans to spread the level across. Modern is more about building out and wide rather than building up and closing in. Shabby chic provides the character in the home. One of the aesthetics we were initially aiming for was that of Octodad and Gang Beasts because of the wacky atmosphere and colorful graphics they have to offer. We updated our final design to include more structure and paths for the bat so that the bat can make either a loop or figure eight on the first and second floor. Initially it felt like a stretch to have all of the rooms on the second floor connected, but we later learned that the gameplay would benefit from this design choice because there are no deadends the players can run into.

4.2 Prop Design

The modeling process for some of the assets varied in difficulty, but used the same approach. Each prop is based on reference images from the internet, modeled using basic geometric shapes and sculpted to achieve a more polished look. Some of the props were made using MASH in Maya. MASH is a system within Maya that works on node based networks to create motion graphics, but can also be used to create models. The process of creating anything starts off with reference images. To create the colander, we create a simple sphere and cut it in half. Extruding all the faces will give it thickness which will make it resemble a bowl. Using a MASH network of a separate cylinder, we attached one to individual points on the target mesh, that being the semi-sphere. Using the boolean function to delete the spaces in between both meshes, we achieved the general shape of a colander. Subdividing the object with smooth it out, and last to be attached are the handles. Another function within Maya called NURBS (Non-Uniform Rational B-Splines) curves, were used to draw out lines in the shape of handles. Using a torus, a shape closely resembling a donut, was cut in half and extruded along the line to make handles on either side. Adding a basic metallic texture will give us a colander. The plan was to separately create a shirt, pants, and apron from scratch with no reference to the actual model, but we quickly realized that method is extremely inefficient. What we ended up doing was making the mesh live, and simply drew out the geometry directly onto the character. This method stays attached to the character, so when it's done all you do is extrude and give the mesh thickness.

Figure 32:Desk Lamp

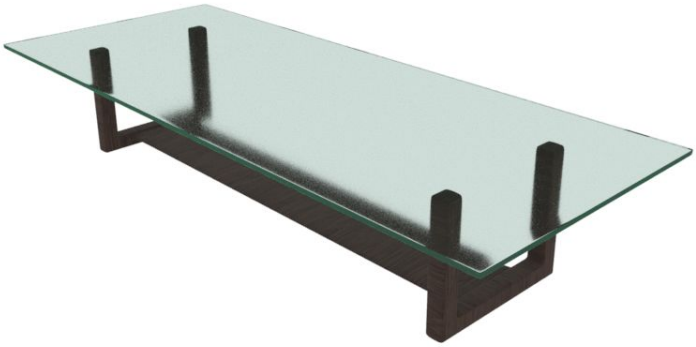

Figure 33: Dining room table

Figure 34 -37:Pots & pans, Potted plant, office desk

Figure 38: Bed

Figure 39: Chair

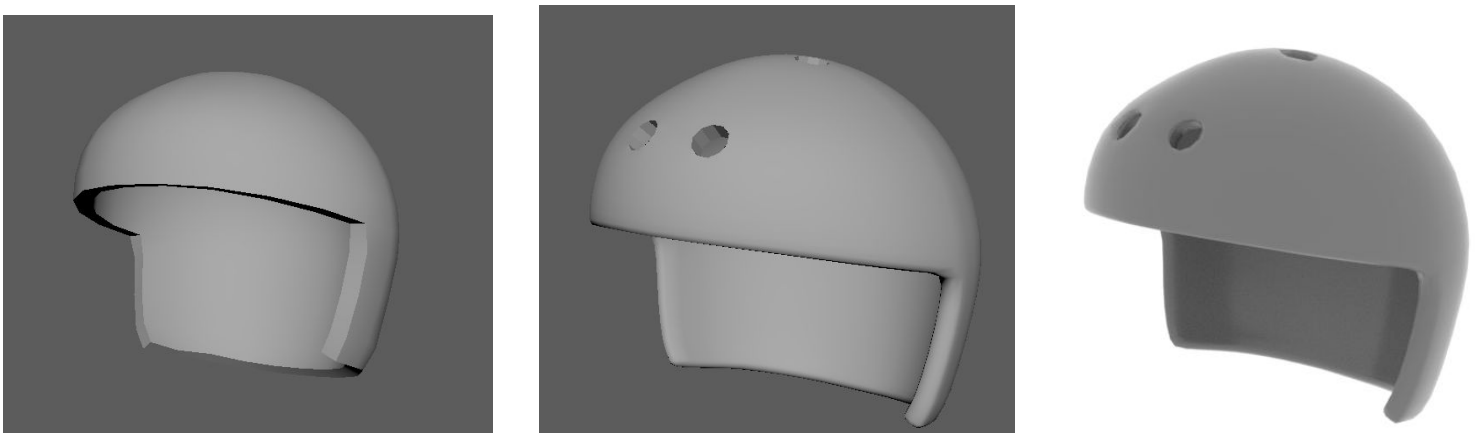

Figure 40: Helmet

4.3 Character Design

Our character models take inspiration from *Codename: Kids Next Door*, a Cartoon Network television series that follows a group of children who fight the forces of adult tyranny powered by improvised household objects.Similarly to the television show, our characters are outfitted with household objects for gear. We have bike helmets and pots for headwear and oven mitts for gloves. Since the characters are mostly covered up in armor, we were able to keep their model structure simple in terms of geometry. The bat was less difficult to create than we initially believed. With our style being cartoon-like, the bat would become a small, round, neck-less creature in the attempt to make it more aesthetically pleasing.

Figure 41: Racketeer

Figures 42 & 43: Sweeper(left), Trapper(right)

4.4 Particle Effects

The bat's flight trail is a straightforward grey particle that gives players a view of where the bat is. Players can track the bat's position easier with this feature, because the bat is small and hard to spot. Particle effects also found use in the menu screen behind all of the UI elements, before players start the game. We are using Unity's particle effects because they require less memory and processing. When the bat and other objects hit the floor, another particle system creates small clouds of dust. Using a still frame of smoke, we can adjust the transparency, lifetime size, rotation, and various other parameters to achieve the desired look of dust being kicked up.

Figure 44: Looping background created in After Effects

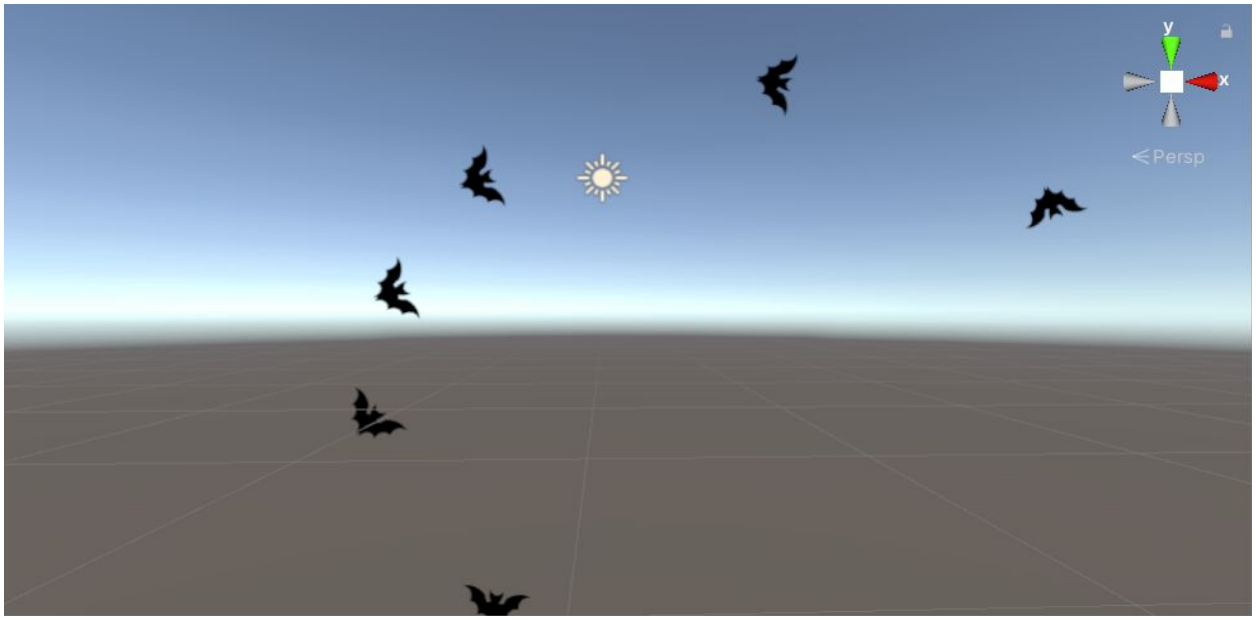

Figure 45: Particle system of bats flying

4.5 Rigging and Animation Transferring

\

The characters are rigged up in maya with their graybox models initially in order to create test animations for gameplay. There's a tool in Maya, HumanIK, that makes retargeting animations easier. Retargeting is the process of transferring animation from one rig to another by mapping the joints of the child character to the parent character. The finished character model

and rig receives all its animation data from the older test character.

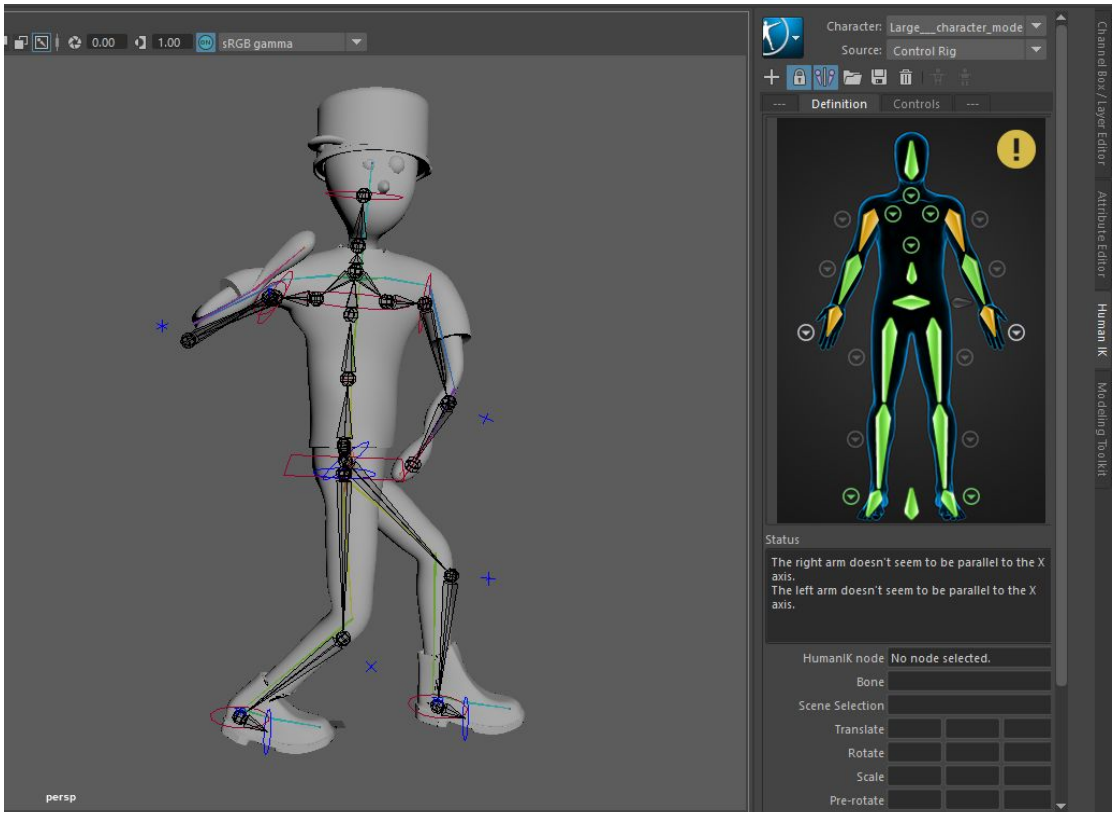

Figure 46: Sweeper rig

The thicker joints are the parent rig, and the thinner joints are the child rig. The controllers belong to the auto-rig feature in HumanIK, which creates a bipedal rig with arm and leg IK. The parent rig has the movements on it, and the definition allows for the joints of the child to mimic the parent. As noted in the figure, parallel joints are necessary for this step, as they can give strange animations with incorrect movement. We mitigate this by hand animating afterwards to adjust the character according to the twelve principles. In this game, timing and follow through are most important, since they determine the hitboxes the player will use for the bat and determine their function. For example, a big, powerful swing has a lot of start up time and plenty of follow through to make a large swing, and in the game world, this is now a big punishment tool for the player when they can predict the bat's flight pattern or cut off large areas of space.

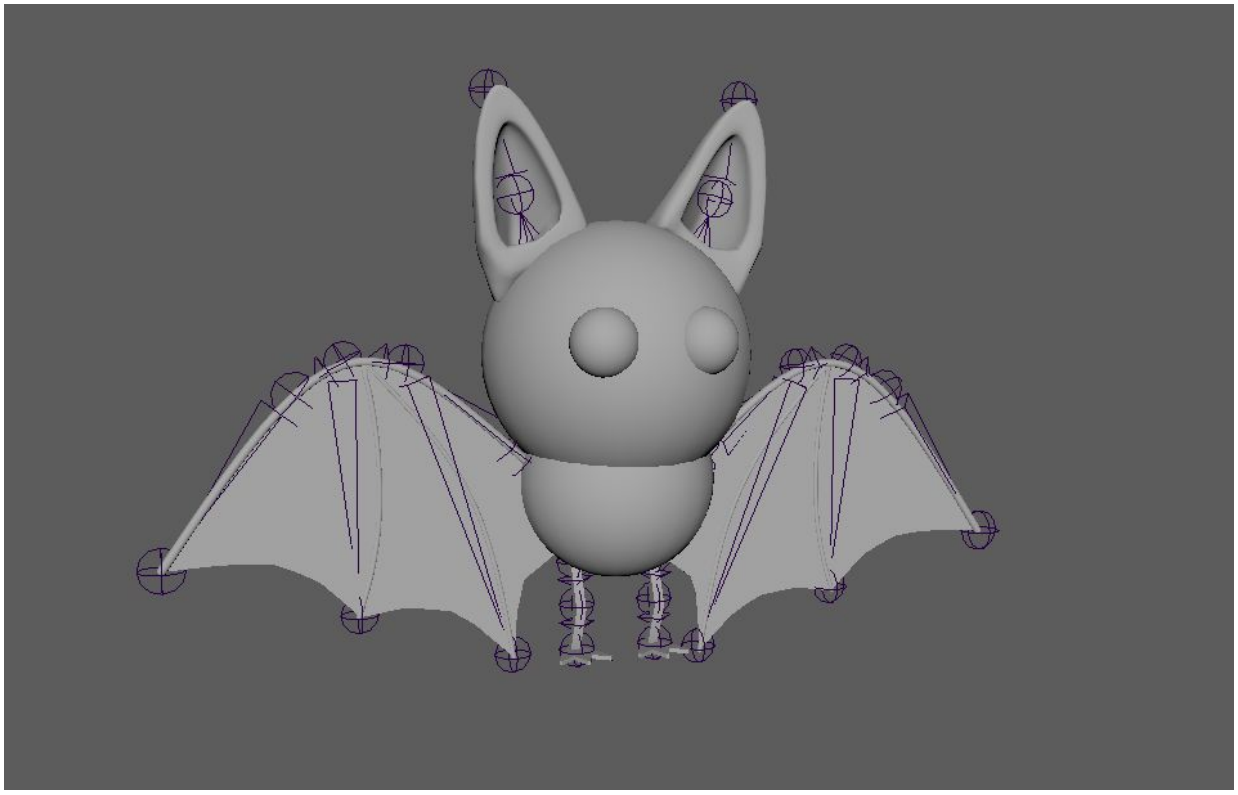

Figure 47: Bat rig

The rig for the bat is similar to human arms, however the elbow has two joints in them rather than the one joint. The extra joint allows for better control over folding the bat's wings up while hanging upside down. It also gives more liberty in banking animations and exaggerations.

5. Technical Design

5.1 Overview

In this chapter, we will discuss the core technical areas of the game, and how systems were deliberated over and improved upon to suit the player's needs and desires in game. Throughout the development of Batastrophe, we wanted to avoid the use of external libraries and packages as much as possible. At the beginning of the project, we determined that we would learn the most from it if we implemented everything ourselves. This philosophy also required us to take a critical look at the technical scope of our designs. Concepts neither programmer was familiar with, such as networking, were identified as a much higher investment of time. Generally tasks were delegated by whichever developer had the time to handle them, but the disparity in Unity knowledge impacted this. One of our programmers had much more familiarity

and experience in Unity than the other which impacted delegation. The other programmer's specialization is in front-end UI/UX, so that is majoritively where his work was throughout the project. As he became more familiar with Unity, his range of work became much wider.

Both programmers made sure to be involved in the whole project and understand how everything works, especially as both programmers became more comfortable with the engine. This assisted both with bug fixing knowledge as well as design workflow, as the next task became easier to delegate when tech fully understood what was in the game and what needed to be added. The following chapter discusses these features and the deliberation of their design from a technical standpoint.

5.2 Shared Input Management

For ease of implementation, input management needed to be independent of player number and class. The detection of a button press should be identical for a player controlling a hunter or the bat, with the only change being in the input handling. As part of its built in input manager, Unity provides the option of mapping controller inputs to individual buttons and axes which can then be retrieved by passing the name of the desired button or axis. Because each button or axis can only read a single input from a controller, 19 of them must be defined to cover a single controller. In order to prevent individual players' inputs from overlapping, each button and axis must be redefined for every player. On top of that, different gamepads may also have different mappings at a hardware level further requiring new sets of button and axis definitions for each type of controller a developer wishes to support. Once all buttons and axes are set up, it is left to the responsibility of each script to correctly query the appropriate input by name for the player and controller type. Rather than running through massive conditionals every frame to poll the appropriate controllers from within the bat and hunter control scripts, we opted to automatically generate the lists of appropriate buttons and axes for each player at runtime. As a result, each script polling the X button, only needs to poll the X input assigned to it at runtime. A bat controlled by player 1 automatically knows to poll the button "X_1" while a hunter controlled by player 2 knows to poll the button "X_2". Though the buttons and axes are configured for a variety of standard controller hardwares, Batastrophe only supports standard PC gamepad input. This decision was made to reduce the necessary testing to ensure that player experiences are truly identical across different types of gamepads and because we as an MQP team did not have sufficient access to multiple kinds of controllers.

5.3 Cameras

The screen for the central game of Batastrophe is split into four individual cameras. The cameras themselves were placed behind the characters, coded such that they follow behind the shoulder of the hunter characters. The camera follows behind the bat with the bat being near the center-bottom of the screen. Camera placement is critical in a game designed around chaos, as visual clarity makes the difference between chaotic nonsense and chaotic fun. If the players had to fight against the camera instead of each other, we considered that a failure.

Figure 48: Split-screen Gameplay

With all of the implications and the critical importance of the camera explained above, it is unsurprising that the camera system is among the most reworked and re-implemented portions of the game. The initial camera system we implemented was uniform across all player types. The player could freely orbit the camera relative to their model while the camera would dynamically zoom in and out to avoid collision. This camera model was fine for the hunters but only serviceable for the bat. It could make the environment feel tighter as a result of the zooming and even then, the collision detection was incomprehensive so we were still forced to put increasing loads of work into clipping fixes. Due to the different shortcomings our system had with the hunters versus with the bat, we opted to make distinct camera controllers for the different player types.

5.3.1 Bat Camera, Cinemachine

For the bat's camera, we chose to use an existing tool set packaged within the Unity engine. Cinemachine is a suite of tools for all types of games, including several presets such as a freelook camera and collider control. This makes implementation of several camera features easier and more effective than hand-coded camera calculations. At large, we tried to avoid using existing packages and assets in places where we could have implemented them ourselves. Moving over to Cinemachine however covered many of the shortcomings of our previous system right out of the box.

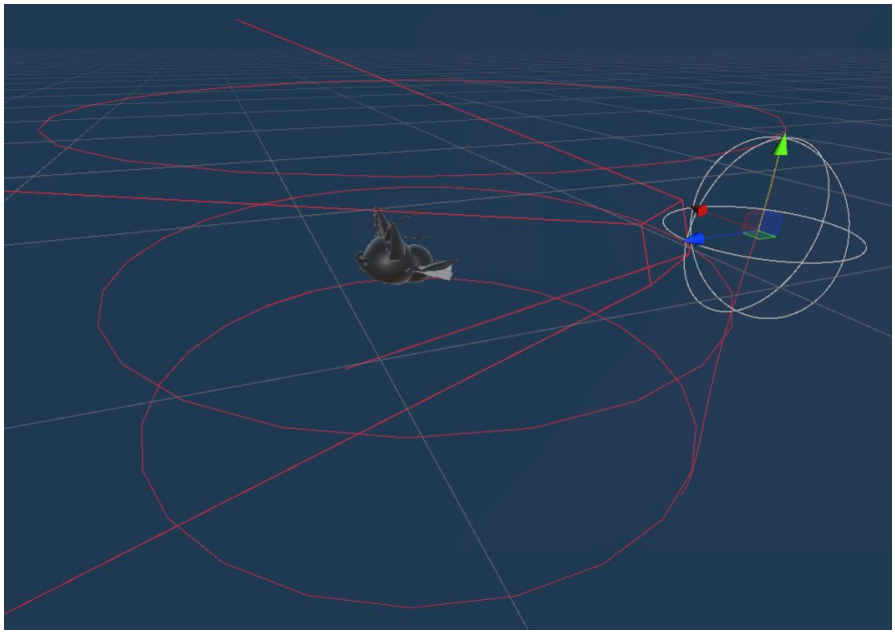

Figure 49: The Cinemachine free orbit camera attached to the bat

The particular Cinemachine camera we opted to work with was the free orbit camera. The camera has three radii at different heights relative to the origin of the camera's focus; in our case, the bat. In adjusting the radii, we are able to limit the angles the bat can look up and down while in flight to keep its direction of movement in focus while clearing that restriction when perched to offer more visibility. We scaled the left to right orbit of the camera to exactly the same turning rate as the bat in flight so the camera will always face toward the bat from behind. Fortunately, changing rotation radii was already a feature we had figured out with the old camera so the technical debt to move over to Cinemachine was very minimal.

5.3.2 Hunter Camera, a Manual Approach

Though Cinemachine ended up covering our needs for the bat's camera, it still didn't offer the level of control we wanted for the hunters. Although we still wanted vertical orbiting of the player model, we wanted the orbit to be directly tied to the direction of the hunter's aim. The result of this is that vertical orbit paths could no longer be interpolated between a couple of radii like with Cinemachine but rather a distance determined by the positions of several targets in each hunter to track the state of their animation. Likewise, because the camera is at a much more predictable location at any given frame, the simple collision detection our old camera system had was more than sufficient. Small objects which could obscure a great deal of the bat's vision are of no major concern for a substantially larger character.

In a reversal of the situation with the bat's camera controls, the level of work required to work with Cinemachine for the hunter's controls ended up being much greater than modifying our existing controller. Though a packaged suite for camera solutions was available to us, relying on it only where it excelled and continuing to implement our own systems alongside it helped us

to maintain the level of flexibility we wanted with our camera systems while simultaneously providing an excellent learning opportunity about the inner workings of the package.

5.4 Object Interaction

5.4.1 Collisions

The main mechanic of the game revolves around smashing into things. As such, it is imperative that objects collide with each other in a consistent and realistic way.

5.4.1.1 Collision Shapes

With a broad variety of props of different shapes and sizes, each prop requires a unique collision shape to make sense within the game's physics. Unity offers several built-in collider shapes, those being box, sphere, capsule, and mesh (based on the mesh geometry). A mesh-based collider would be ideal for all objects and works great for static geometry but becomes generally infeasible for non-convex props. As a constraint of Unity's rigidbody physics system, mesh colliders may only be used on physics objects if the collider is made convex. For a donut, this would simply fill in the hole but for a table, this would fill in the entire space under the table. Props which are small enough or which can sacrifice precision of collision, a convex collider is more than sufficient for the purpose of this game. This would include a bottle, a small vase, the toaster, etc. Props which are much larger or require convex interactions however, such as a table or a trash bin, must have a custom set of colliders to ensure that they work correctly.

We identified two options for covering this limitation: fragmented collision meshes, and attaching multiple simple colliders to an object. Both of these options also offer an interesting boon in that each collider on an object can have different behavior; the handle of a hunter's weapon versus the head for instance. The first option, fragmented collision meshes, involves breaking the mesh into several smaller convex shapes, each with the same origin and scaling factor relative to the base mesh. Each of these shapes can then be used as an individual mesh collider under the object. The second option, attaching simple colliders to an object is similar to the first but with more restrictions on the convex shapes composing the object. With multiple similar colliders, the box and sphere colliders can be adjusted and moved around to cover different parts of the object but capsule colliders must be centered around the object's origin.

The first method is generally more expensive technically and artistically but produces the most precise collisions for an object. The second method on the other hand is much faster to set up imprecisely but takes a lot more work to be precise. As such, the collision shapes of any non-convex prop had to be weighed for importance of precision against time complexity. Meshes like the hunter's weapons which have highly distinct shapes and completely separate behavior from the handle to the head of the weapon are well worth the time to get their collisions just

right. A prop such as a rectangular table however may only need around five simple box colliders, one for the table top and one for each leg, so it would not be worth the additional scope to ensure flawless interactions. Likewise, a rectangular waste bin would only need box colliders for the bottom and sides but a round bucket would need considerably more effort to act round from the outside while still allowing things to fall inside of it.

5.4.1.2 Applying Forces

Physical forces can be applied to props by hunters, weapons, the bat, and other props. Physical forces however can only be applied to players by other players. In dealing with props, the Bat player has two distinct flight speeds and each hunter has a unique walking speed and two attack speeds. These values all present a different effect that will be imparted onto struck items in the environment. At a technical level, players are excluded from the physics system with the exceptions of the physical space they occupy (each player has a special capsule collider encompassing their model) and the ragdoll of the bat. Instead, player controllers listen for collision events between their own collider and that of a prop.

If the bat is hitting a prop, a force will be applied to it based on the bat's current speed and the mass of the object. At normal flight speed, the bat may nudge a pan off of a countertop and knock a cup a few feet aside. During a boost, the bat may knock over a chair or send a cup flying across the room. If a hunter is walking into a prop however, the object is only pushed lightly in the direction of the hunter's movement; not even budging if its mass is greater than that of the hunter. A prop struck by a hunter's weapon is imparted a mass-dependent force based on the hunter's class. The racketeer's weapon behaves similarly to the bat with a greater force on horizontal swings than vertical attacks. The sweeper's weapon applies a great deal of force on the horizontal swing but unlike the racketeer, applies almost no force on the vertical swing, even slowing objects it strikes. The trapper's weapon applies a small, fixed, mass-independent force to any props it hits. A prop which was hanging from a wall or other object will be dislodged and fall once hit.

The bat's ragdoll imparts the same forces to any prop it collides with but receives proportional opposite forces from the props it strikes. The ragdoll will bounce around the room, quickly losing momentum as it crashes into more objects or rolls with friction across the ground. Props similarly apply forces to each other on collision. A prop struck by a player that is knocked into another will apply a dissipated force and lose momentum. These props will however attribute the source of the collision to the player that hit them as long as the force remains above a certain threshold. Props struck by a player will not go as far from subsequent player hits. This helps show that a prop has already been scored with as well as reduces the amount of objects flying around the scene thereby making it easier for players to identify what's happening around them as well as reducing the amount of physics calculations which need to occur each frame.

5.5 Player Interaction

5.5.1 Between Players

The typical gameplay loop for a hunter consists of finding the bat, attempting to strike it, and removing it from the house. As described in chapter 3 section 1.2, the behavior of the bat when struck differs based on the hunter that hit it. In order to handle this interaction, the bat player controller manages a separate ragdoll state. Depending on a set of parameters associated with each type of hunter, the bat will stay in this state for an allotted period of time with slightly altered behavior. If struck by the racketeer, the bat will act as a free physics body, launched quickly in the direction of the racketeer's aim, continuing to smash anything it is knocked into. If any hunter reaches the bat in this state, they scoop the bat until the end of its stun duration and carry it according to the rules of their class. If struck by the sweeper, the bat will simply drop from where it was flying, only interacting with what it hits on the way down and staying stunned longer than the racketeer. If struck by the trapper however, the bat will enter immediately into the carried state for the trapper and will neither act as a physics object or be possible to pick up for any other hunter. When each hunter chooses to toss the bat, it will re-enter its ragdoll state for a short distance as if being hit by the racketeer.

There is not, however, only a single hunter and bat in the level at a given time. Hunters running around may bump into another. A hunter bumping into someone else will be stopped in their tracks, unable to advance unless the other hunter gets out of their way. For flexibility in implementation, this interaction is the same for the collision of multiple bats, in a ragdoll state or otherwise. Likewise, hunters might strike one another while aiming for the bat or be struck by the bat itself. In these cases, the hunters will each be dazed for a short period depending on what struck them. A hunter struck by another's weapon will be stunned only briefly. A hunter struck by a bat, however, will be stunned longer depending on their class, trappers being stunned the longest, and racketeers the shortest.

5.5.2 With the Environment

In order to keep the bat out of their house, hunters must be able to open and close the windows and doors of the house. If a bat is outside while all portals into the house are closed, the hunters have successfully trapped it outside. If it is trapped inside, the hunters must be able to reopen windows and doors in order to get it out. Bats may freely travel through any open window and doorway, but will be blocked, and therefore be trappable, by any which are closed. Should the bat be trapped outside, the round will end early and the hunters will have been successful. Should it be trapped inside, the hunters will be notified as such so they can track it down and get it out.

5.5.3 Destructible Objects

Some of the props in the house are destructible. With that status, comes a number of special interactions between players, those props, and the environment. For a non-destructible prop, the only direct effect a player can have on it is altering its position. Likewise, the only effect a non-destructible prop can have on a player is being in that player's way. Destructible objects however, when certain conditions are met, can have their entire shape and behavior adjusted. These objects can be shattered upon the application of a sufficient breaking force by the bat, a hunter's weapon, or collisions with other objects in the environment. If for instance, a destructible prop is knocked off of a table, the force knocking it aside may be insufficient to break it but the collision with the floor may break it. Any destructible prop is automatically shattered on contact with the bat or any hunter's weapon but otherwise requires a unique force in order to break. Once broken, the prop will visibly shatter, spreading shards of its previous form on the ground. Though each shard is a physics object, they will not impart any force onto adjacent props. Each shard has a small convex collider around it which will slow any hunters attempting to walk over it.

5.6 Menus and UI

5.6.1 Menu Format

The main menu scene is split into multiple different layouts, which become invisible and visible accordingly when the navigation buttons are pressed. The previously mentioned options menu, the main menu itself, as well as a controls menu which displays controller layouts for both the human and bat player, allowing players to see the mechanics of each player. The options menu interacts with the GameSettings script to alter player and game variables.

The Main menu consists of four different "bubbles" representing a separate character each. These each take in controller input from each individual player, allowing players to select and control their own menu without having to move from their seats. A timer was implemented for each player's left stick for character selection, as these are registered as axes with constant updates on its position. To get one input, we check if the stick's axes are greater to or less than 0, and that the timer has reached 0, determining that a player can act again. It then rotates in the array of classes, displaying information about the new class and their gameplay style, while updating the timer back to its designated starting time.

5.6.2 In-Game User Interfaces

Since our game is based in a four-player split screen setting, we need all of the in-game content to be easily readable regardless of the screen resolution. UI development was centralized around maximization of visual clarity with simultaneously minimized screen clutter.

Different iterations of interfaces were used in regards to tracking the scores of the players, including a "combo-based" method involving displaying recent scorings. The other

method thought of was a constant score display in each user's individual portion of the screen. There was deliberation regarding which would assist the game the most and be the better implementation of a score display UI. The combo display involved less constant screen-space, with the downside of not tracking score constantly. We ended with the decision to display the scores constantly, with the color of the individual score text matching that assigned to each player number. For gameplay purposes, we theorized that the constant player knowledge of all scores improved the competitive nature of the game. Urgency to catch the bat is increased when you know the bat player may surpass your own score.

Another element of the game we wanted to highlight to players is the human's abilities to close windows, and the Bat's abilities to hang from ceilings. Whenever a player gets in the range that they can do either of these things, a boolean for that player titled "promptB" is set to true. To make sure that the players are aware of when and how they can do this, whenever a player's promptB is true, an image following the art style of the game is set to visible, either saying "press B to use" if they are the hunter, and "press B to land" if they are the bat. This affixes to the bottom of that player's screen until promptB is set to false again from them leaving that range, at which point that player's display is set to invisible again.

Outside of the individual UI for each screen, there are also Global User Interfaces which are designed to be easily visible by all 4 players. The main of this is the Timer which has been placed in the center of the screen, equidistant from every player's point of focus. The timer ticks down from 2 minutes and 30 seconds, until it reaches 0. At this point a subroutine is run which checks if every player has gotten a chance to play as the bat. If anyone has not played the bat character, it goes back to the character select screen, randomly selecting a player who has not yet played as the bat to be the bat for next round. If everyone has played the bat at the end of the timer, it loads an end-game screen, declaring the winner of the game as well as every player's scores.

The other global UI implemented are notifications regarding the location of the bat. The house itself has collision detection around the outside walls, allowing the bat to detect whether it is in or outside of the house. There also exists the array of window objects, being the windows of the house. Each window has a boolean stating whether it is opened, or closed. If every window is closed, a large UI message is placed across the top of the screen. If the windows are closed and the bat is inside the house, the message states "The bat is locked inside!". If the bat is outside of the house, the message states "Locked the bat outside!". This also triggers the hunters to receive points, with the hunter who threw the bat outside receiving more points. This sets the timer to 0:00, as the hunters have won this round and the next round can begin.

Another key element that is less UI as it is visual clarity improvements and quality of life changes to the game is an implemented trail on the bat. Since the bat can appear as a very small speck on players' screens while hunting for him, spotting and accurately attacking the bat can be difficult for many players. This is where a unity Trail-Renderer was implemented into the prefab for the bat character. After tweaks and implementation into code, as well as tentative video review of different lengths and trail styles, the trail for the bat has been quantitatively adjusted to fit the style and gameplay of the bat, making the bat visually apparent enough to the human

characters, but not so visually apparent that the bat's location is always known to the other players. The bat has the ability to go fast and slow, as well as perch, and the trail alters its length gradually based on what actions the bat is performing, such as a longer trail when the bat is moving at full speed.

Reflecting on our decisions regarding tech in this project, one of the stranger decisions that was made and then never improved upon is making the script where User Interfaces are altered the same script that a good number of important game events occur. Scene transitions, round determination, the only method the hunters can score and more are all determined by the GameUI script, instead of in their own individual script files. Some aspects of this decision made sense, for example the timer. All the math for calculating the round times is done in the script for UI, so adding more scripts would be redundant and unnecessary. This is one example however, and much more of the script could have been properly organized and coordinated.

6. Production and Project Management

To maintain the flow of contributions and keep everyone on task and focus, we organized the development of our progress through the use of the Kanban method of project management.

6.1 Kanban Boards and Trello

Kanban, which translates from Japanese to billboard or "visual signal", is a project management method that allows individuals or teams to visualize and organize tasks that need to be completed in a way that promotes consistent delivery and prevents work in progress bottlenecks. Using this strategy, team members are able to quickly examine what work is done, what is currently being worked on, and what needs to be started. In our case, we used kanban to help us work towards developing specific features, fixing any outstanding issues, and making sure we were on track for specific milestones. (Rehkopf, "What is a kanban board?")

Trello is a free web-based productivity tool that utilizes the Kanban scheduling format to illustrate the work that has already been completed and what needs to be worked on next. Each board is separated into columns of "To-Do" "In Progress" and "Done", and cards are created that describe an action item. Each of these cards can be moved under one of these categories to show the current status of that task, and assigned to key members of each task. Towards the end of A-Term, we created several Trello boards for tasks based on its field (Art, Code, MQP Paper and Presentations, and a general board.)

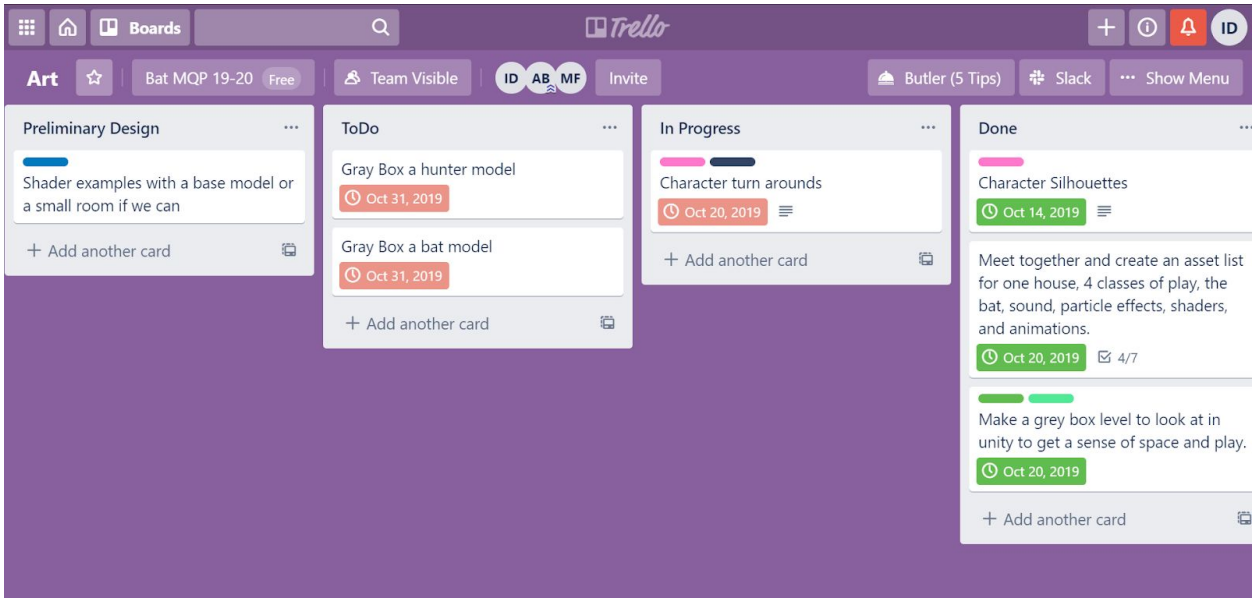

Figure 50: Trello assignments

While progress was steady and more focused with the help of the kanban boards, it was becoming increasingly less apparent as to what each member was to work on next due to having a miniscule amount of assignments spread across too many boards, and the team not consistently following the boards, which led to a reduced workflow, and thus, a decline in productivity. To fix this, we decided to consolidate every single action item into one board named "Road Map" and change the format of completing and assigning tasks through this board. Instead of having separate columns for what is currently being worked on and what is completely spread across different disciplines the tasks are now assigned on a weekly basis, and for every task completed, it moves into the singular "DONE" column.While we were able to get the team back on track, by consolidating the boards to make it easier to follow, the damage was already done, and we lost a ton of time that could have been better spent working on the project.

6.2 Google Drive

Google Drive is a cloud based storage solution that allows users to share and upload files with each other. We used Google Drive as a way to store art assets before introducing them to the project proper by using Git.

6.3 Git and Github

Github is a version control platform that allows for a stable project directory to be easily shared and edited by contributors. Whenever a teammate edits a file, they must "push" their changes to the project, and every member will get those changes once they "pull" data from the repository. Github also allows for branches, which allows individual users to make separate changes to files. We created a private repository for this project so that only members of the team are allowed to contribute. Stable releases were also released on our project page for easy distribution and documentation.

6.4 Video Conferencing Tools

Google Hangouts is a free video and voice chat web program that allows users to communicate with each other via the internet. In addition to video, you can also share your desktop screen with whoever is on a call. In this project, we used it for all team meetings to share progress with members of the team who could not physically make it to the meeting.

Zoom is an enterprise level video conferencing tool that we also used to host virtual meetings. In addition to allowing us to meet via voice or video chat, Zoom also provides options for meeting via telephone call, sharing a desktop screen that others can draw notes on, and built in recording support so we can document meetings and save for the use of note taking. We shifted from Hangouts to Zoom during the 2020 coronavirus outbreak, which is detailed in section 6.6.

6.5 Meetings

At the beginning of each meeting, each member shares their progres, details any problems, and asks questions to the rest of the team about their progress. After that step, we come together to discuss the current status of the game build, talk about improvements, impressions made from playing it. Afterwards, we create assignments for the team at large to be completed and adjourn after any questions and concerns are addressed.

In addition to team-wide meetings, we also had weekly breakout meetings for artistic and technical progress. The artists and technical developers would then be able to discuss matters that are more relevant to their responsibilities and ask for any assistance if needed. The producer would attend both meetings to understand the progress of the project at a deeper level. In these meetings we discussed individual progress and implementation details more deeply, often using the time as a scheduled collaboration session between the artists or programmers on the team.

6.6 Changes in production during the Covid-19 Pandemic

The last term in this project was unexpectedly interrupted by the coronavirus (covid-19) outbreak of 2020. As a result of this and to protect everyone, WPI announced that the campus would be closing and the rest of D-term would be carried out remotely.

This put us at a huge disadvantage, as we could no longer all meet together in person to discuss and work on the project. In addition to this, additional playtesting would not be possible due to the game requiring 2 or more people in close proximity with each other with the appropriate amount of controllers, something we can not monitor while quarantined at home.

Despite these issues, we managed to continue development at a relatively smooth pace thanks to regular meetings and aggressive scheduling for each individual task. Our Trello boards were updated to list each task that needed to be completed for each week, and the team were able to give consistent updates by using another video conferencing tool, Zoom. With Zoom, we

scheduled different meeting times for tech, art, students, and the entire team throughout the week so we were always in contact with each other.

7. Testing

Describing the methods and results of user testing.

While finding out what people like/don't like about our game was the main goal from testing, another insight we were interested in seeing was whether playing this game caused any sense of discomfort, motion sickness in the user. While this is more important for a game that utilizes Virtual Reality technology, where cybersickness is more likely in VR compared to a standard flatscreen viewing (Foster), we felt that this was still an important data set to gather. While playing as the bat, the player is making large, sweeping movements constantly, and we want to make sure that players who play this game for long sessions are safe while playing as the bat. Especially players who typically do not play games with similar manners of motion such as *Microsoft Flight Simulator.*

7.1 Methods

Testing was done towards the end of C-Term from the weeks of February 22nd to March 4th of 2020. Informal testing, which encapsulates having the team's relatives or friends play in progress builds of the game, was done intermittently throughout the development of *Batastrophe.* Each session began with a brief explanation of the game, its controls, and the goal. During the first few days, we had subjects play with one of the team members in a one on one situation. Towards the end of the testing period, we had 3-4 students play with each other, with each student getting a chance to play as both the bat and the hunters. Sessions lasted from 3 minutes for group playtesting sessions, to 5 minutes for solo sessions, after which they switched from bat to the hunters. After the sessions, the player was invited to describe their thoughts to the facilitator and ask questions before they fill out an exit survey that detailed their experience with the game. Selected questions from this survey include:

- How would you describe the speed of the bat/hunter?
- What are your feelings about the camera control of the bat/hunter? Did it feel easy to
- control? Difficult to get used to?
- Out of all of the playstyles, which one did you enjoy the most?

with most questions allowing the subject to elaborate further on their thoughts.

7.2 Formative Testing Results

We used the data we gathered to help guide us on how to improve the game for its final iteration at Showfest. For example, we had a subject note that the items in the dining room were the most fun to knock down as the bat. That helped influence the decision to include a wine room

on the 2nd floor of the home, which makes use of some props that are prominently featured in the dining room on the first floor. Another example of us taking feedback to improve the game occurred when many people noted that they did not make use of the bat's ability to perch on a ceiling. To encourage use of the feature, we made tweaks to the UI so players know that they have the option to perch if they fly close enough to the ceiling.
All the data from the survey is included in the appendix section.

7.3 Testing Post Mortem

Testing should be done the moment you have anything that moves or reflects gameplay. It can be a paper prototype, or you can have people running around with some instructions of what to do. We did a live playthrough of our game with holding real brooms and having a player be the "bat." We hunted the "bat" down throughout the house and talked out our actions in order to get an idea of what our players would do in the scenario, both in and out the game.

If given the chance to, we would have created more questions that were directly related to gaming motivations and if we were able to meet our design experience goals. While playtesting, one subject was really impressed by the simplicity of the bat controls, which they found extremely helpful due to their lack of familiarity of video game controls. Finding more subjects who aren't as adept at video games would have also given us more insight as to how the game performs with that specific demographic.

8. Post Mortem

8.1 Producer Role and Project Management

One of the students chose to become a producer for the team (in addition to his artistic responsibilities); handling matters such as scheduling meetings, setting deadlines, and communicating any outstanding issues to the appropriate advisors. His experience in managing events such as the 2018 Different Games Conference and the events for the Game Development Club in 2017 made him a strong candidate for taking on this role in the project. With a student working primarily on project management, it would become a lot easier for the tech and visual art students to focus on their individual tasks while the producer primarily works on maintaining the project.

The producer's job was to maintain communication between the art and tech students, and keep the production of the game going. Similar to what a producer would do in the video game industry, they act as the glue that creates a schedule, keeps the team together and maintain progress to hit the deadlines (Schrier, "What A Video Game Producer Actually Does") An asset list was created by all of the members on the team, but overseen by the producer in order to determine progress of the current build of *Batastrophe*. They also were responsible for leading the research on our game for style and setting. There were initial check-ins for each group member in order to keep track of our individual work. To make this easier and more obvious, the producer created a timesheet to keep track of the work done by each student and to see how much time each student spent working as a way of holding everyone on the team accountable for their contributions to the project. Lastly, due to their role, they were responsible for setting up the team's weekly meetings and keeping note of progress, attitudes, and concerns.

The primary benefits to having a student producer were recognizing who possesses certain skills and correctly delegating who should lead each task. The presence of a student producer did however have a few downsides which negatively affected the team's progress. As a result of not diligently following and maintaining the schedule, the work that team members may end up doing may be of a lower priority, already completed by another team member, or the work would be unfairly distributed. Having a member of the team whose main job was keeping everything organized divested the rest of the team from taking responsibility with maintaining order. As a result, when the asset list and delegation of work became poorly managed, team members quickly lost sight of their responsibilities and clashed over incomplete or late assets. The lack of accountability team members faced for not completing work on time was the source of an overwhelming level of conflict that our team had to overcome towards the tail end of the first semester.

While there were cases where the producer's work was invaluable, such as scheduling the meetings, playtesting sessions, and organizing the project as a whole, the lack of control and responsibility that the producer ended up exercising was a key factor in the lack of progress and

eventual crunch the team had to undergo towards the end of the project. A producer that performs well in their role can be incredibly helpful but a producer who performs poorly may hurt a team more than having no one in the role at all. If a future IMGD MQP decides to have a student producer, we would recommend the following:

1. During the early stages of development (1-2 Terms before the project begins), have them take any available courses that teach them the process of team development, scrum scheduling, sprints, etc.
    a. IMGD 5400. Production Management for Interactive Media is course that students may benefit in taking, or at least observe
2. Be mindful of ALL deadlines for the project ahead of time, including due dates of the IRB, submission deadlines for school and local game events, and playtesting sessions. While it is expected that everyone in the group is mindful of these deadlines, it is especially imperative that the producer knows when they are so they can properly plan ahead
3. Work closely with faculty advisors and report any issues with progress, no matter how big or small.
    a. Faculty can offer advice on how progress can continue, or use their power to reprimand students who aren't performing at their expectations.
    b. Similar to having a tech and visual arts advisor, a producing advisor that helps guide a producer student may also be beneficial.

8.2 Art Pipeline

The pipeline for this MQP consisted of multiple drafts and versions of models to keep the flow of productions. Throughout the year we have learned a slew of techniques on how to approach certain tasks. When it comes to making assets, it is important to keep in mind the resolution of the model being created relative to what the machine can handle. If the model is too dense in geometry and the computer can't handle it, it will slow down, possibly freeze, and is also susceptible to crashing. This will create a domino effect of slowing down the pipeline and creating lag in progress. It's okay to attempt the same method once or twice to see if errors were a one time thing, but artists need to know when to stop and think about a different method of approach. There is never one way to do things so being able to adapt and figure out alternate ways to create what you want. It's also smart to look at the asset list your team has developed and sort out what are simple models and what will take longer to create. In the long run, the entire pipeline will benefit from greyboxing models that are anticipated to take longer than to spend extra time making a polished model in one go. It's relatively a poor management of time and for assets which have a major impact on gameplay such as rough player models and the environment, it is extremely important to have basic versions ready to implement in order to keep the rest of the team from getting blocked waiting for them. However on the contrary, with much smaller items, it is more acceptable to make a polished version early on. The important thing with the art pipeline is keeping it efficient.

Speaking of time management and which assets need more attention, you should also find ways to cheat your modelling process and use your advisors. Our art advisor provided a character creation workshop per request, because our team struggled to make playable characters quickly. Using the presets in the Maya toolkit or in the ZBrush models is an acceptable practice for creating assets that can be stylized. Greyboxing and creating a base mesh for your pieces allows for clear communication on what scale the team is working with. Shortcuts to make life easier are welcomed when working with tight deadlines. A good example is using the auto-rig tool in Maya to animate first, and then creating your own rigs later on. It saved the project to have animations done that quickly and easily transferable.

8.3 Tech Management

Our two programmers began work on the project with vastly different levels of experience regarding working in the Unity Engine. One was very comfortable and familiar, having worked in the engine a multitude of times before Batastrophe. The other contrastingly began work with no experience outside of the tutorials and lessons he did in preparation for this project. Since the start there was an inherent difference in the speed of which tasks could be performed, and this was taken into account regarding the pipeline and the implementation of Github version control.

Our pitfalls on the tech side often fell to issues regarding communication and clarity, as well as issues with the workflow being linear. The linearity of our pipeline involved more problems regarding blockages and redundant work. Scenarios such as one teammate finishing their task faster than anticipated lead to unnecessary halts in workflow. Likewise, if one student had more difficulty than expected on their task, the other would be blocked waiting on results. Our faulty communication also led to hours of wasted work when both people ended up implementing the same feature without communicating about it. As later work on the project became more parallel, we found that the amount of time each teammate spent waiting on someone else's work reduced drastically but issues such as merge conflicts began to arise. These issues were generally mitigated by careful consideration of where and when we were working but brief lapses in communication could result in conflicts ranging from inconvenient to disastrous.

Despite this, the tech workflow for the majority of the project was passable and devoid of significant problems. When there were problems however, they halted the project more intensely and longer than they needed to. This was slowly over the course of the project as communication between the programmers on the team steadily improved. Early communications were kept mostly to laid back group meetings when everyone met, with infrequent communication regarding tech specifically. Throughout D-Term, there was a shift to discuss tech-specific aspects more frequently. A weekly meeting was scheduled to exclusively discuss tech and informal check-ins were made almost daily to ensure everything was on track. This also

shifted the discussion of roadblocks. Whereas we would typically only discuss bugs and issues during team meetings, regular informal check-ins shifted that balance to handle issues immediately.

As our communication improved and organization improved, the workflow became more streamlined and flowing, with less stops.. Our project would have benefited greatly from finding a better balance in the beginning and taking communication more seriously.

8.4: Final Conclusion

Consistency is key for your game to feel progressed and complete. It was difficult for our art team to settle on a style due to the varying abilities of each artist. Asset mapping everyone's skills initially is important, but also have everyone sit down and focus on how the game should look. Each teammate should provide inspiration and reference for what they want to do in the game, but you need to remember that it all needs to align with a core style. Our style was coherent, but it wasn't consistent among the gameplay and models. Make sure each artist is communicating and editing each other's work. This will ensure that there are no surprises with assets not matching up with the world or breaking the flow of gameplay.

An MQP is a great time to learn new skills and test them out in a real game example. Define your goals early on what everyone wants to accomplish with the project. This will make delegation easier for the team as a whole. Front load as much research as possible early on. The team should research each thing that will be a challenge or new in the game. A solid recommendation is to spend around four hours of your work time learning the skill. It sounds repetitive, but team cohesion and communication is what pushes a project forward. The artists and coders should have a basic understanding of each other's work, because this will make critiquing, testing, and deadlines smoother and less stressful. Sharing your research and what you're learning is important for this process.

9. References

Barbera, W. H. (Director). (1951). *Jerry and the Goldfish* [Motion Picture]. Metro-Goldwyn-Mayer.

Botero Homes. (2013). BOTERO French Manor IV. Retrieved from http://boterohomes.com/botero-french-manor-iv/

Casen, Sara. "White Boxing Your Game." *Gamasutra*, 13 July 2016, www.gamasutra.com/blogs/SaraCasen/20160713/276970/White_Boxing_Your_Game.php.

Decor Aid. (2019, June 12). Urban Modern Interior Design Defined: Everything To Know. Retrieved from https://www.decoraid.com/blog/interior-design-style/urban-modern-decor

Linda Foster, C. F. (2020). Virtual Reality Is Sexist: But It Does Not Have to Be. *Frontiers in Robotics and AI*, 4. doi:10.3389/frobt.2020.00004

Mastroeni, T. (2020, April 16). Defining a Style Series: What is Shabby Chic Design? from https://freshome.com/inspiration/shabby-chic-design/

Rehkopf, Max. "What Is a Kanban Board?" *Atlassian*, 2018, www.atlassian.com/agile/kanban/boards.

Schreier, Jason. "What A Video Game Producer Actually Does." *Kotaku*, 22 Apr. 2016, kotaku.com/what-a-video-game-producer-actually-does-1772519753.

Simmons, K. (2020, April 8). What Is Shabby Chic? Everything You Need To Know. Retrieved from https://www.decoist.com/shabby-chic-style/?chrome=1

Yee, N. (2019). A Deep Dive into the 12 Motivations: Findings from 400,000+ Gamers. *Game Developers Conference.* Quantic Foundry.

10. Appendices

10.1 Playtesting

10.1.1 Playtesting Informed Consent Agreement

Informed Consent Agreement for Participation in a Research Study

Investigator: Isaac Donkoh-Halm

Contact Information: ikdonkohhalm@wpi.edu

Title of Research Study: Batastrophe

Sponsor: N/A

Introduction
You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

For this study, you will be playing *Batastrophe*, a competitive cooperative party game where one player is a mischievous bat who wants to cause chaos and knock over items, while the other 3 players must work together to stop them

Purpose of the study: The purpose of this study is to gather user impressions of the design, difficulty, accessibility, and the overall quality of a video game.

Procedures to be followed: When the study begins, the participant(s) will be asked to fill out a survey with their preferences of video game types. After completing that survey, the participant will play through the video game while the proctor takes notes on any observations they make during the session. After the session is completed, the participant is then asked to fill out an additional survey to document their comments and suggestions for the game.

Risks to study participants: There are no risks to participating in this survey.

Benefits to research participants and others: The survey data will be used to improve the controls, design, and accessibility of the video game.

Record keeping and confidentiality: All records during this study will be kept confidential, and results will be coded as to protect the identities of the subject. The student investigators and project advisor will have access to the coded results.

Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators, the sponsor or it's designee and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or presentation of the data will not identify you.

Compensation or treatment in the event of injury: While there are minimal risks to participating in this study, you do not give up any of your legal rights by signing this statement in the event of any injury sustained while participating in this study.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact:

Student Investigator: Isaac Donkoh-Halm Email: ikdonkohhalm@wpi.edu

IRB Manager: Ruth McKeogh

Tel. 508 831-6699

Email: irb@wpi.edu

Human Protection Administrator: Gabriel Johnson

Tel. 508-831-4989

Email: gjohnson@wpi.edu

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent ag reement._____

_____ Study Date: _____
Participant Signature

_____
Study Participant Name (Please print)

_____
Signature of Person who explained this study Date: _____

Special Exceptions: Under certain circumstances**,** an IRB may approve a consent procedure which differs from some of the elements of informed consent set forth above. Before doing so, however, the IRB must make findings regarding the research justification for different procedures (i.e. a waiver of some of the informed consent requirements must be necessary for the research is to be "practicably carried out.") The IRB must also find that the research involves "no more than minimal risk to the subjects." Other requirements are found at 45 C.F.R. §46.116.

10.1.2 Playtesting Survey

Start of Block: Informed Consent

QA

Welcome to Batastrophe!

You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits,

risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

You will be presented with information relevant to our game and asked to answer some questions about it. Please be assured that your responses will be kept completely confidential.

The study should take you around 15 minutes to complete. Your participation in this research is voluntary. You have the right to withdraw at any point during the study, for any reason, and without any prejudice. If you would like to contact the Principal Investigator in the study to discuss this research, please e-mail Isaac Donkoh-Halm at ikdonkohhalm@wpi.edu.

By clicking the button below, you acknowledge that your participation in the study is voluntary, you are 18 years of age, and that you are aware that you may choose to terminate your participation in the study at any time and for any reason.

Please note that this survey will be best displayed on a laptop or desktop computer. Some features may be less compatible for use on a mobile device.

O I consent, begin the study (1)

O I do not consent, I do not wish to participate (2)

End of Block: Informed Consent

Start of Block: Control - Bat

Q01 How would you describe the speed of the bat?

- O Too slow, (couldn't navigate the world fast enough) (1)
- O A little slow (2)
- O Just about right (5)
- O A little fast (6)
- O Too fast, (easily lost control of the bat) (3)

Q02 What are your feelings about the camera control of the bat? Did it feel easy to control? Difficult to get used to?

_____

_____

_____

_____

_____

Q03 How did it feel to knock down items as the bat?

_____

_____

_____

_____

_____

Q04 How often did you make use of the landing/take off feature?

- O Never used it once (1)
- O Not that often, used it once or twice (2)
- O Often, used it more between three to five times (3)
- O More than often, used it more than five times (4)

Q05 What are some control features you feel that the bat was lacking/missing?

_____

Q06 Describe any additional thoughts you have about playing as the bat.

_____

_____

_____

_____

_____

End of Block: Control - Bat

Start of Block: Control - Hunter

Q07 How would you describe the speed of the hunter?

- O Too slow, (couldn't chase down the bat fast enough) (1)
- O A little slow (2)
- O Just about right (5)
- O A little fast (6)
- O Too fast, (easily lost control of the hunter) (3)

Q08 Describe your feelings regarding the camera control of the hunter. Did it feel easy to control? Difficult to get used to?

_____

_____

_____

_____

_____

Q09 How effective was closing/opening the windows to corner the bat?

- O Not very effective, I did not really pay attention to them (1)
- O Average, I closed maybe 1 or 2 windows during my time with the game (2)
- O Effective, I closed/open the windows at any chance I got (3)

Q10 How difficult was it to knock down the bat?

O Easy, hit the bat 5+ times in this session (4)

O Average, hit the bat 3-5 times this session (5)

O Difficult, only hit the bat 0-2 times in this session (6)

Q11 What are some control features you felt that the hunter was lacking/missing?

_____

Q12 Describe any additional thoughts you have about playing as the hunter.

_____

_____

_____

_____

_____

End of Block: Control - Hunter

Start of Block: Visual Design

Q13 Describe what you liked the most about the environment:

_____

_____

_____

_____

_____

Q14 Describe what you liked the least about the environment. Was there anything that we were lacking?

_____

_____

_____

_____

_____

End of Block: Visual Design

Start of Block: Final Impressions

Q15 Out of all of the playstyles, which one did you enjoy the most?

O The Hunter (General) (1)

O The Bat (5)

Q16 Please list any of your final thoughts here:

_____

_____

_____

_____

_____

End of Block: Final Impressions

10.2 Concept Art and Grey Boxes