November 2015

# Chinmoku MQP

Dillon Joseph DeSimone
*Worcester Polytechnic Institute*

Robert Edward McKenna
*Worcester Polytechnic Institute*

Samuel Selig Wallach
*Worcester Polytechnic Institute*

# Chinmoku

A Major Qualifying Project Report

Submitted to the Faculty of the

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

On November 9, 2015

沈黙

CHINMOKU

**Submitted by:** Robert McKenna, Francesca Carletto-Leon, Dillon DeSimone, and Sam Wallach

**Advised by:** Professor Robert Lindeman

## ABSTRACT

This report details the developmental process of *Chinmoku* (“silence”), an educational game developed to fulfill the Major Qualifying Project requirement for Worcester Polytechnic Institute’s Interactive Media and Game Development (IMGD) and Computer Science majors. This project was developed over a three month period at Ritsumeikan University’s Biwako-Kusatsu Campus in Shiga Prefecture, Japan. The game seeks to teach Hiragana, one of the Japanese writing systems, to a target audience of young adults familiar with gaming. This report covers all aspects of the team’s development process, research, playtesting, and the possibilities of future work on this project.

## Acknowledgements

The *Chinmoku* team would like to thank a number of persons who assisted us in the development of this large-scale project.

A large thanks goes to our adviser, Rob Lindeman, for providing feedback at weekly progress meetings and guiding the project from its conception to its completion.

We would also like to thank Ritsumeikan University BKC campus for hosting us in Japan during our project and for providing work space and computers for us in their lab. A particularly warm thank you goes to Professor Noma Haruo and Professor Roberto Lopez for welcoming us into their lab and assisting us whenever possible. The students of the Media Experience Design Lab were extremely accommodating and good friends to us during our stay. We hope these friendships continue to stay strong far into the future.

We would also like to thank Jenna Hoffstein of Little Worlds Interactive for consulting with us concerning the design and testing of educational games.

Music and audio credit goes to Tangent Music, LLC, a group of Berklee students and alumni. We were thrilled that *Chinmoku* could feature an original soundtrack and for that we're very thankful.

# TABLE OF CONTENTS

## LIST OF FIGURES

# 1 INTRODUCTION

*Chinmoku* is a horror puzzle game which utilizes educational elements as its core gameplay. Players progress through a mystery narrative while using Japanese language skills as tools to solve puzzles and engage in combat with monsters.

Taking advantage of our team's placement at one of the Japan Project Centers, we hoped to study our own learning of the Japanese language to create a functional and well-paced educational tool. By observing our own study techniques and pace at which we were able to retain new Hiragana characters, the team was able to make informed decisions in regards to level design (which dictated the overall pacing of our game).

The art of *Chinmoku* aims to imitate a hand-painted style. Painted textures, employing an unsaturated color palette, create a decrepit atmosphere which furthers the horror aspect of the game. These highly detailed textures were applied to relatively simplistic models, which we were able to produce quickly and efficiently to maximize the amount of assets we could create.

The game is designed to be run on a computer, while players can connect their smartphones via Bluetooth to use as a touchscreen controller to practice writing, as well as character recognition and auditory skills. The smartphone was incorporated to be an easily accessible way to allow players to interact with the game via touchscreen, even without owning a tablet or other stand-alone touchscreen device. The game can also be played solely on the computer, if the user so chooses.

Our goal for this project was to create a polished prototype of an educational game with engaging gameplay. We wanted to avoid utilizing the ‘flash card’ formatting that many educational games employ and instead form an experience where the educational elements feel seamlessly incorporated into gameplay.

This cumulative paper discusses the entirety of our development process. Chapter 2 covers the design and gameplay of the project, including much of our pre-production planning and research. Chapter 3 discusses the creation of the visual assets for *Chinmoku* and talks about our stylistic goals and artistic pipeline. The technology and systems we developed for our game are covered in Chapter 4. This chapter explains how we developed the game for use with the smartphone controller by working in both Unreal Engine and Android SDK. In Chapter 5 we discuss the modularity of both the environmental art assets and gameplay systems, designed for quick and easy implementation in the final game. Chapters 6 and 7 cover the post-production of the game, discussing playtesting and the team’s postmortem respectively.

# 2 Design and Gameplay

*Chinmoku* was designed to be a standalone experience aiming to teach the user one of the three writing systems utilized in the Japanese language. This section of the report discusses our pre-production process, including conducted research and informed design opinions made by the team.

## 2.1 Experience Goals

We attempted to create game mechanics into which language learning could be incorporated seamlessly. The game contains frequent repetition through writing activities that are directly related to the game world. The focus is on the game itself, such that the player is so immersed they forget they're actually learning the basics of a language. We believe that the intended genre of the game will also facilitate learning. The puzzle portions provide ample practice for players to use their language skills for problem solving. The pacing (how many puzzles were needed per level before the player becomes comfortable with the material) was largely based on the number of exercises present in each section of Japanese children's workbooks purchased by the development team. This number was typically between three and five. The horror/mystery portions of the game place a sense of urgency and importance on learning and being able to recall characters quickly.

Our team ventured into this project having never made an educational game before but we still had some very clear ideas about how we wanted to structure gameplay. During our own attempts to teach ourselves basic Japanese, our team had come across numerous free-to-use and paid learning applications. Almost all of these applications utilized flashcard based gameplay, where the user is shown Japanese and then has to match with the equivalent English translation (or vice versa). We found this type of gameplay boring since it didn't require any variation of interaction from the user and therefore, we felt, only taught the player how to memorize without providing them with context to make their language learning relevant.

After realizing we all felt similarly about the flashcard based language learning approach, the team focused on designing gameplay that employed a variety of interactions in hope that this would make the learning experience feel more like a game and less like a classroom memorization exercise.

### 2.1.1 Platform and Target Audience-

Initially we had proposed to develop a language learning game for mobile platforms to facilitate accessibility and ease of use. We felt, however, that the limitations of mobile platforms would restrict how immersive of a story/environment we could create and would be in direct conflict with our experience goal. Because the touch screen capabilities of mobile phones are essential to the design and gameplay of the project, we chose to develop for PC but still make use of mobile game interactivity. As such, the game allows users to connect their smartphones to their PC via Bluetooth and to use the phone as a controller.

Taking a look at the Apple app store we noticed that learning games were most commonly marketed towards children. According to Monique Zytnik of Adjust.com, in 2014 a whopping 60% of iOS apps for children are educational in nature and seek to teach skills like language, math, and creativity (among others). Educational games intended for adults tended not to teach new skills but to develop already obtained skills, such as memory. The horror/mystery aspects of this game require that it be intended for older audiences, although it seeks to teach a skill instead of heighten one. To research how to design this, our team observed both children's and adult educational apps. Designing for an adult audience also allowed us to accelerate the pace of learning and to create difficult language based puzzles/riddles.

## 2.2 Story

The protagonist of our story is a young woman who is going to Japan for the first time to visit a childhood friend. When her friend stops responding to messages and calls, she knows something is wrong. She goes to investigate her missing friend's home and is sucked into a twisted horror mystery where words and speech themselves are being stolen. To find her friend the protagonist must recover the stolen language piece by piece and use it to navigate through the game's setting.

The monster is a parasitic entity which attaches itself to a host and then is passed on. It steals the name of its host and their ability to communicate. At the beginning of the game, the monster is still attached to the protagonist's missing friend. She had been trying to contain the monster before the player's arrival but failed. The entity then passes on to the player when they start snooping around and trying to solve the mystery of what happened to the friend.

In order for the player to defeat the monster, they must recover the language that is being stolen by clearing each level and learning new language skills. After each level, the player discovers the name of one of the past hosts and clues about from and to whom it was passed. By assembling the names in the correct order, the monster can be defeated.

### 2.2.1 Characters

The protagonist of *Chinmoku* is Rita, a young woman traveling abroad for the first time. Rita is designed to be a silent avatar and, as such, doesn't have any voice lines indicating specific personality traits. All dialogue was written to provide commentary and guidance to the player, as opposed to inserting Rita's own ideas about the situation. Figure 1, seen below, shows Rita with both the original and final versions of her texture.

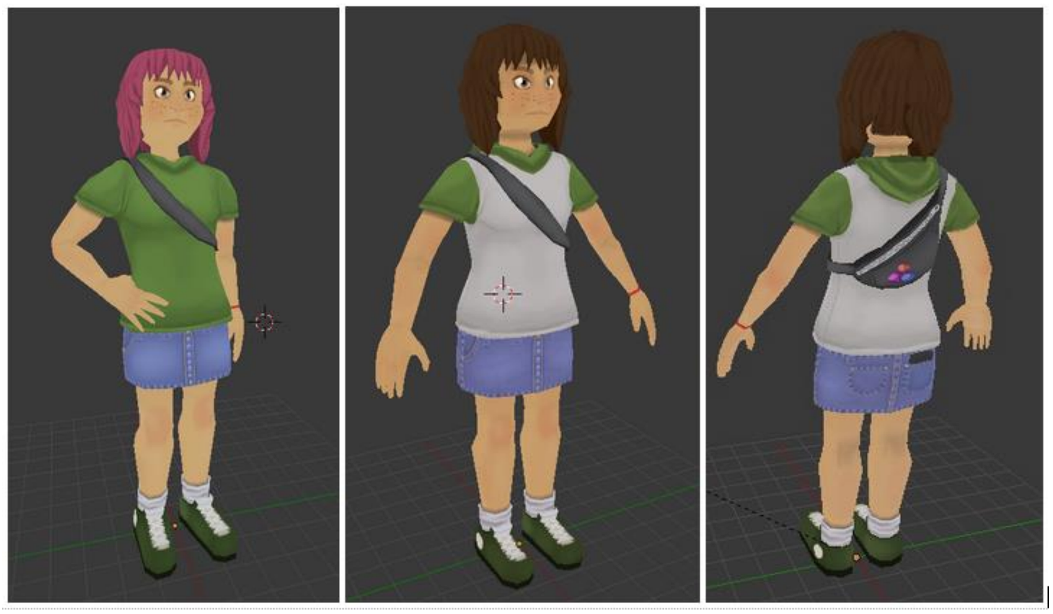


*Figure 1: Iterations of Rita's texture.*

Rita's Japanese friend (whom she is visiting) is Sachiko, a kind-hearted but mysterious girl. Sachiko becomes involved with the parasite after it was passed to her by a coworker. Though she sought to contain it, the monster eventually overwhelms her and steals her ability to communicate. Throughout the game, various clues can be found through the levels and Rita sometimes receives text messages with hints, though it is often unclear if these are from Sachiko herself.

### 2.2.2 Setting

*Chinmoku* takes place in a suburban Japanese town which draws inspiration from the residential areas surrounding our host University. The game features both indoor and outdoor environments which are based on reference materials (examples of which can be seen below in figure 2) our team collected while abroad. The outdoor areas, particularly the streets and houses, were created by imitating the patterns and textures we found while observing our surrounding environments and then tweaking them to create a dark setting for our game. It was important for the team that our story be designed to provide a setting where the incorporation of language learning elements would feel justified. Taking advantage of our time abroad in Japan to create a believable Japanese environment in the game was vital.

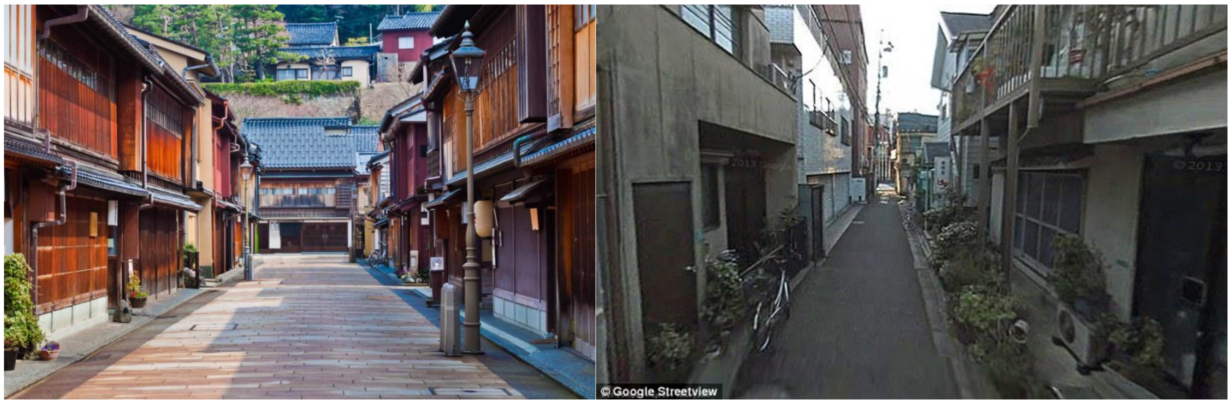


*Figure 2: Reference images of Japanese residential streets. Photo credit to Zicasso.com and Google.*

The player begins the game on a normal town street and must navigate to Sachiko's home, one block over from the spawn point. Sachiko's home is the main location of the game and is the hub area, from which the player is able to access each level of the game. The various levels take place in both indoor and outdoor settings which are not in the ‘normal’ world. The level areas are paranormal in nature. Each area is a self-contained instance not at all connected to the house. Instead, these areas each pertain to one of the previous hosts and provide clues about who they were and where they fit in the order of the monster’s hauntings.

### 2.2.3 Player Motivation

Our most important objective was to create a learning experience which motivated the user. In 2002 Dr. Mark Griffiths, a professor of Gambling Studies in the Psychology Division of Nottingham Trent University, published an academic article about the educational benefits of video games in which he says “playing computer games (irrespective of genre) produces reductions in reaction times, improved hand-eye coordination and raises players’ self-esteem”. In academic environments, one of the leading causes of decreased learning capability is the lack of motivation of students. Game-based learning, which can easily regulate the frequency of rewards, has been proven to “impact positively on problem solving skills and knowledge acquisition” (Game-based Learning: Latest Evidence and Future Directions, 2013, p. 10).

Since their conception, video games have been designed to reward players for their efforts and stimulate interest and motivation to complete challenging tasks. If gameplay is well designed, a player will never be asked to complete a task which at that point in the game is impossible for them. As an example, in Massive Multiplayer Online (MMO) games, areas of the world map are frequently divided by difficulty and a player will never receive a ‘quest’ in an area which is too difficult for their current level (McGonigal, 2010).

Though the player is given freedom to explore, quests are awarded in a semi-linear method and the player knows that though they may struggle with the challenge, that the task is ultimately achievable. This allows the player to stay motivated to retry despite previous failures. Overcoming a challenge and receiving a reward for their efforts boosts a player's self-esteem and increases their desire and ability to learn (Steinhardt School of Culture, Education, and Human Development, 2013).

### 2.2.4 Reward Systems

The formatting of games is ideal for educational purposes because of the ability to provide a structured rewards system for players. The "reward schedule" dictates when and how players will be given positive feedback for their efforts. Usually, players will be given rewards after completing a specific challenge and the reward given will match the effort exerted. When designing the magnitude and frequency of rewards the designer must consider the type of game and what challenges the player will face. In the Nintendo series, The Legend of Zelda, players are given small rewards (money or health regeneration) for defeating common enemies. After defeating larger enemies and bosses or completing a more difficult task (such as clearing an area) players are rewarded with significant items which assist the player. These include hint items such as maps or weapons needed to defeat enemies and access new areas (such as the boomerang). The game plays a cinematic and plays a dedicated jingle when presenting the player with these items, to make the reward feel meaningful (see Figure 3).

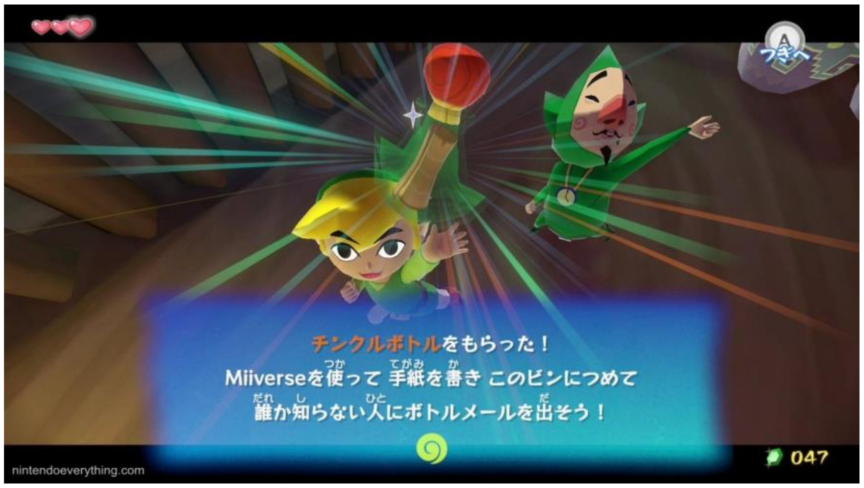

チンクルボトルをもらった！
Miiverseを使って 手紙を書き このビンにつめて
誰か知らない人にボトルメールを出そう！

*Figure 3: Acquiring an item in The Legend of Zelda: Wind Waker.*

In *Chinmoku*, there are two major types of gameplay, puzzles and combat. Each type of gameplay provides a different challenge for the player. When designing our reward schedule we felt it was important to provide positive feedback after every gameplay challenge in order to keep the player motivated to continue learning. Small but frequent feedback creates an atmosphere of positivity and accomplishment (Chatfield, 2010). Since no boss battles exist in the game, larger rewards are given after the player has cleared each level. Attempting to cater to various types of players, the rewards given also vary as hints, solutions to puzzles, narrative, and progress through the game.

### 2.2.5 Failure States

The importance of positive feedback cannot be stressed enough, though it is also vital to talk about if and when it's appropriate to give negative feedback. In most of *Chinmoku*'s gameplay, we found that negative feedback could be ignored. During teaching segments (when the player is introduced to a new Hiragana column), a failure state doesn't actually exist. If the player makes a mistake, their answer remains unrecorded and they are able to try again. During this initial learning phase, there is no way to fail. In the case of puzzles, players can fail by inputting incorrect answers, and negative feedback is instant and minimal. In this situation, the player is able to retry the puzzle instantly, without being penalized in anyway other than being told they were incorrect.

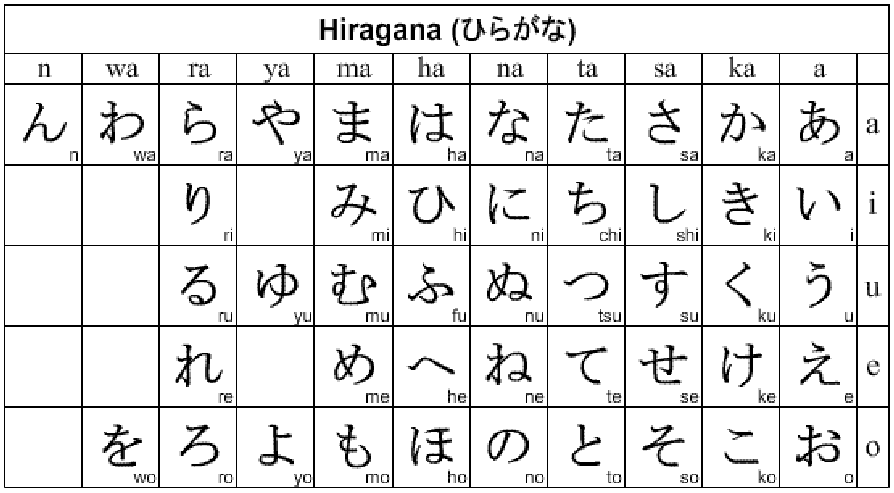
Combat is the only instance where a player can actually reach a fail state. If the player fails combat and the protagonist is 'killed', they respawn at a point earlier in the level and are then allowed to navigate back to combat and try again. The 'death' and time it takes to respawn act as negative feedback for failed combat. While we wanted the player to feel the implications of failing (thus making succeeding more meaningful), we also didn't want to punish them too much in fear they may become too discouraged to try again. We felt that in the case of an educational game, maintaining high self-esteem and positivity is a priority because it gives the players the means to learn more competently (Tekofsky, 2010).

## 2.3 Player Interactions

The user interactions in *Chinmoku* were designed to be as simplistic as possible to facilitate understanding of the game's mechanics. We have three major gameplay types which are categorized as Training, Puzzles, and Combat. These core gameplay portions of the game all occur in instances, gameplay that is distinguished by alternative controls. The other two main actions the player can make occur outside of instances and consist of avatar movement and interactions with the environment.

## 2.3.1 Training

Hiragana is one of the three writing systems used in Japan. It is a syllabary type of writing, where each character represents a mora (syllable) present in Japanese language, with exceptions given to vowels and 'N' which is classified as a stand-alone consonant. The characters are organized into a chart which is laid out by combining each possible consonant sound with the five vowel sounds (see Figure 4, below). When learning hiragana, students are most often taught the hiragana chart by columns, starting on the right with the column of vowels and moving to the left.

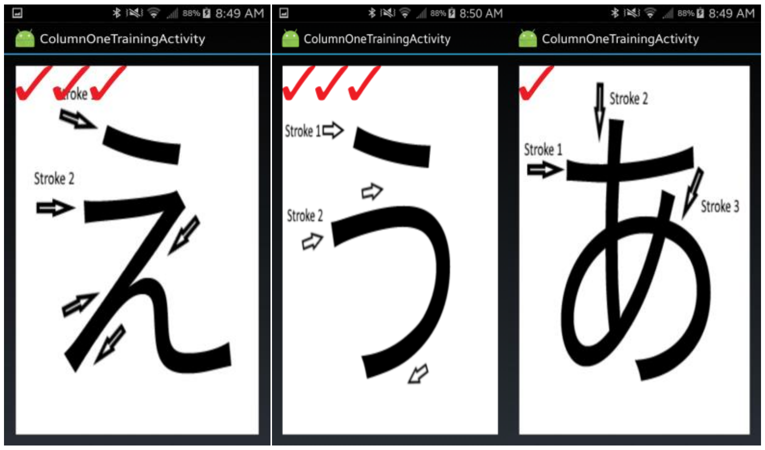


Hiragana (ひらがな)

| n | wa | ra | ya | ma | ha | na | ta | sa | ka | a | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ん n | わ wa | ら ra | や ya | ま ma | は ha | な na | た ta | さ sa | か ka | あ a | a |
| | | り ri | | み mi | ひ hi | に ni | ち chi | し shi | き ki | い i | i |
| | | る ru | ゆ yu | む mu | ふ fu | ぬ nu | つ tsu | す su | く ku | う u | u |
| | | れ re | | め me | へ he | ね ne | て te | せ se | け ke | え e | e |
| | を wo | ろ ro | よ yo | も mo | ほ ho | の no | と to | そ so | こ ko | お o | o |

*Figure 4: The basic hiragana chart.*

Training occurs in the 'Hub' area (Sachiko's house) before the player embarks into each individual level. Because each level focuses on one column of the hiragana chart, the player is slowly introduced to the chart and is able to practice just five characters at a time. During a Training instance, the player is asked to trace all of the hiragana from the new column they have just unlocked in order to proceed. Each character must be traced correctly four times. The player is introduced to new content in the Hub area because it is the safest area of the game and while the player is learning a new column it is important that they don't feel rushed or pressured in any way. After they have had time to trace over and learn the characters (shown in Figure 5), they can progress to the following level where their knowledge is tested through puzzles and combat.



8:49 AM
ColumnOneTrainingActivity
Stroke
Stroke 2
8:50 AM
ColumnOneTrainingActivity
Stroke 1
Stroke 2
8:49 AM
ColumnOneTrainingActivity
Stroke 2
Stroke 1
Stroke 3

*Figure 5: Training demonstrates instructions for the drawing of each character.*

### 2.3.2 Puzzles

For our prototype, six puzzles were designed and implemented. The puzzles were all designed for play on the PC and then ported to the Android app for play using a Bluetooth connection. Puzzles were designed roughly in the order they were intended to be encountered but were set up to allow the usage of different hiragana, so that they might be rearranged and reused at any point in the game.

The first puzzle designed was the Keypad Puzzle, demonstrated in Figure 6 below. The keypad puzzle has an onscreen keypad that allows entry of hiragana characters, and gives a string of romaji (Latin-script alphabet) characters as a clue. This puzzle is the first time a player will interact with Hiragana in the game. They player must figure out to enter the Hiragana characters based on the order of the romaji characters. This puzzle is encountered before the player is taught any Hiragana. As such, after entering the characters, the puzzle UI tells the player which characters were entered correctly and which were not. This allows the player to figure out the puzzle without having to resort to entirely random guessing.

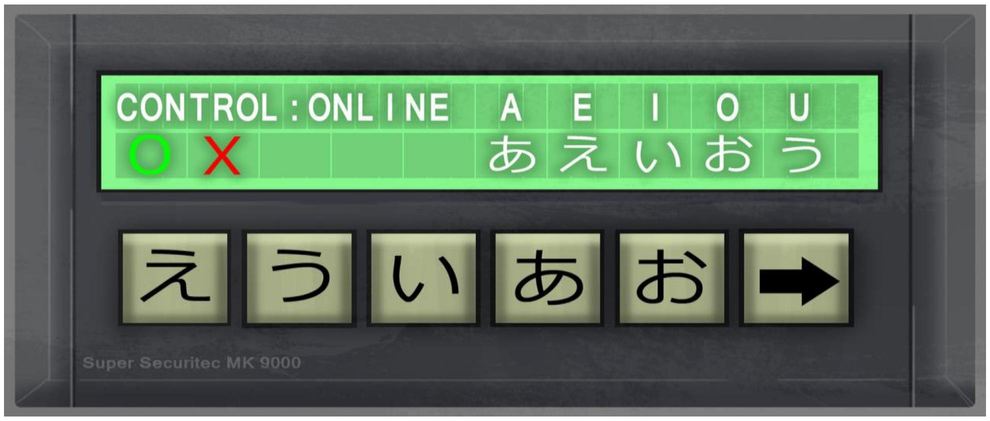


*Figure 6: The Keypad puzzle is the first to be encountered in the game.*

In the next puzzle, the Stick Puzzle, there are various “sticks” shaped like hiragana strokes, which must be dragged and dropped to shape a hiragana character. This puzzle just tests the ability to remember the general shape of a given character and was designed to be an easy, early puzzle.

The Lock Pick Puzzle requires players to use the phone controller or mouse to move a lock pick in the correct directions in the right order to open a door. Various regions of the lock are labeled with hiragana, and a note with romaji clue of the order is found elsewhere. The design of the main lock pick moving mechanic was inspired by the lock picking mechanics in games such as Fallout 3 and Skyrim.

The Sliding Tile Puzzle has a column of tiles with hiragana written on them, and when a tile is clicked it moves to the top, shifting any tiles above it down one slot. The player again receives a clue in romaji of the order for the tiles. Aside from testing the ability of the player to translate between romaji and hiragana, there is also the challenge of determining the correct order to slide the tiles in to get them where they are meant to be.

### 2.3.3 Combat

The combat system is the system through which the player is tested in-game, and the knowledge gathered is solidified. Combat involves a number of enemies, each with a romaji character floating in front of them that the player will need to be able to write (visualized below in Figure 7). While combat progresses these enemies move closer and closer to the player until they are able to inflict damage. The player must not only know the hiragana representation of each romaji character to be successful in combat, but they must also be quick, and able to prioritize which enemy to attack. The player's health, during combat, takes the form of a balancing bar where each attack to the player causes a segment of their health to be depleted, while each successful attack on enemy regains a portion of their health.

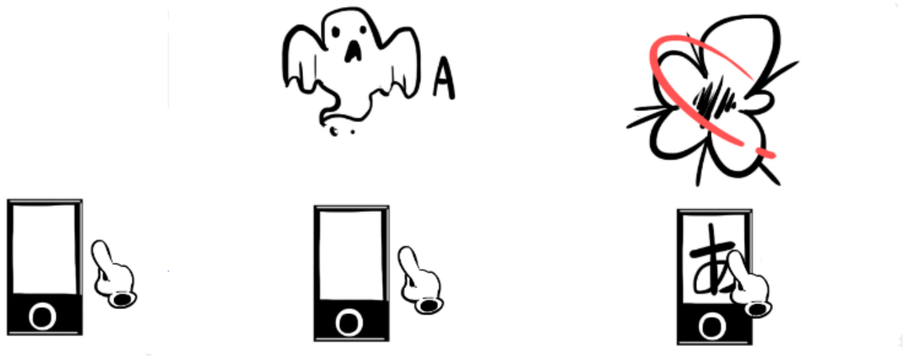


*Figure 7: Step-by-step visual of combat gameplay.*

In addition, the enemies in combat and their romaji characters are presented to the player in a queue system where the enemies that do attack the player are added back to the queue. This system ensures that the player may not progress unless they have proven that they can write and translate all the required hiragana characters. Of course, if the player fails to do so they have an opportunity to traverse and explore the level, either to do training again or to replay some of the puzzles for practice.

### 2.3.4 Win Condition

After each level is completed, the player brings back the name of one of the previous hosts of the monster and then unlocks a new hiragana column and level. Once the player has collected all of the names and mastered all of the hiragana columns (by clearing each level), they should have collected enough clues to assemble the names in the correct order. Doing so summons the monster for a final, extreme combat where the player must utilize their knowledge of the entirety of the hiragana chart to defeat the monster.

# 3 Visual Assets

*Chinmoku*, as an adventure game, required an impressive number of artistic assets. The asset list contained over 150 entries, ranging from models, to textures, to rigs and animations. Because our project was produced abroad, we did not have access to WPI's academic program licenses. Our team was forced to adapt to this new environment and create a new pipeline to produce *Chinmoku*.

## 3.1 Pipeline

The asset team for *Chinmoku* consisted of two artists. On any team with multiple artists, the cohesiveness of the art assets can become an issue. Each artist has a different method or process while building assets, and these differences can be reflected in the game. If the character is painted with a detailed, realistic texture while the environment is blocky and cartoony, the game will look jarring and amateur.

The development team found two ways to mitigate this problem. The first, which is especially helpful when art teams have many members, is to establish a visual identity for the game in a document. This document, able to be referenced by any member, should be a full visual dictionary for the game. Environment reference, color palette, character proportions, and any other important aspects of the game's visual identity should be documented. Using this document, any artist should be able to create an asset that fits in with all the other assets created by the other artists.

The second method is simply to assign asset creation roles to the artists, instead of assigning assets. That is to say, instead of having Artist A create a box and Artist B create a barrel, Artist A will model both and Artist B will texture both. Using this method, the assets are ensured to have a good measure of cohesion. Another advantage to this method is clarity what each artist's objectives are. If an object needs to be modeled or textured or animated, there's never any confusion about whose job it is to complete it. The caveat to this arrangement is that it works most easily if the art team is relatively small so that the artists don't outnumber the separate roles which need to be filled.

While working on *Chinmoku*, both methods were utilized. Because we only had two artists, dividing jobs was a simple matter. After solidifying the concept of the game in a detailed design document and gathering reference images of artistic styles the team wishes to imitate, both artists went to work creating concept art and a visual identity for the game. The roles for asset creation were divided according to interest and skill. One artist handled Modeling and Game Importing, and the other handled Texturing, Rigging, and Animation. Since the order of development generally went: Modeling > Texturing > Rigging > Animation > Importing, assets only needed to be transferred between artists twice, saving time and effort.

## 3.2 Artistic Style

All assets in *Chinmoku* were stylized to make up for our technological constraints. Knowing that we didn't have access to powerful programs like 3DS Max and Zbrush while abroad, we opted for a different approach. Since our only available programs were Photoshop and Blender, we decided to create lower-polycount characters and assets, with high-resolution painted textures. Not only did this let us create a fast-moving pipeline, but was appropriate given our limitations. Creating realistic, high-resolution characters is possible in programs like Blender, but would take a lot of effort and time that could be used making or improving other assets.

Our primary inspirations for the game's visual identity were older horror games like *Silent Hill* and *Resident Evil*. These games provided good reference for textures, camera positioning, and lighting, demonstrated in Figure 8.

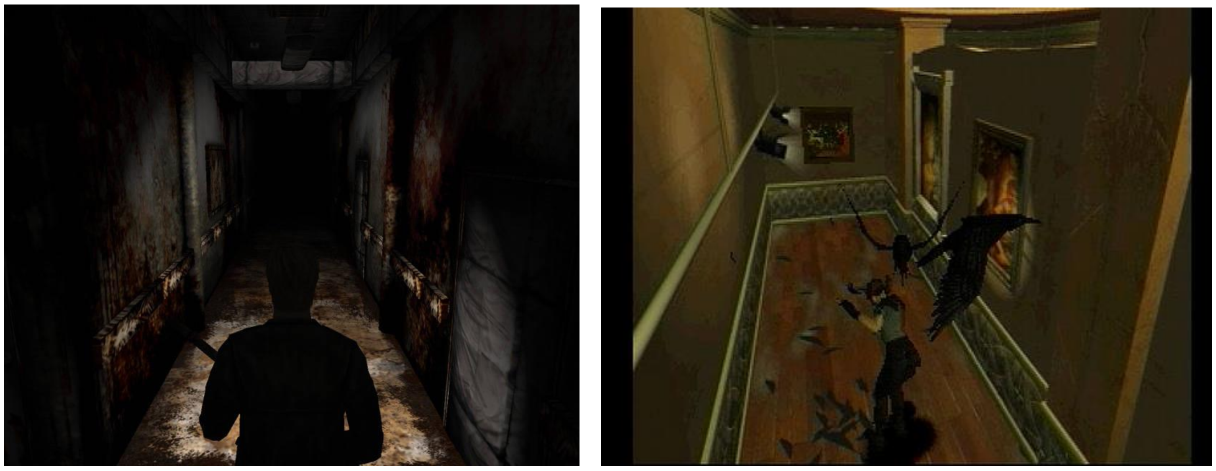


*Figure 8: Screenshots of Silent Hill (Left) and Resident Evil 1 (Right), used for reference.*

## 3.3 2D Assets

*Chinmoku*'s 2D assets were all produced in Photoshop. Textures, buttons, and puzzle assets were all created to look "painted", both to harken back to older horror games like *Silent Hill* and to mesh everything together. Many of these 2D assets were designed for use on the phone, puzzle backgrounds, buttons, and decals. Because of the small size of the screen, readability was extremely important. Simple, bright colors on top of dull backgrounds made sure that even small characters could be distinguished from each other.

Puzzle assets were often designed all together on the same document, and painted as one image. Later, once the design was complete, these images were separated and exported. This way, separate .png files could be used as buttons by themselves, reducing the workload on the programmer. Additionally, Hiragana and letters featured in the puzzle were exported as their own images to be overlaid on the buttons. This would allow us to go back and swap in new letters onto the same puzzle, if we so chose.

Textures were primarily painted in Photoshop, always drawing from a limited color pallet of dull reds and browns to make sure that the assets stayed cohesive with each other. These colors would usually be applied in Blender's UV Editor, so they could be seen on the model and checked over for quality. After laying down basic colors, the colors would be exported to Photoshop for further rendering. Details and noise would be added, adding realism and roughness. Additional 'grunginess' would be added by overlaying existing 'dirty' alphas at a low opacity. Examples of texturing are shown in Figure 9 below.

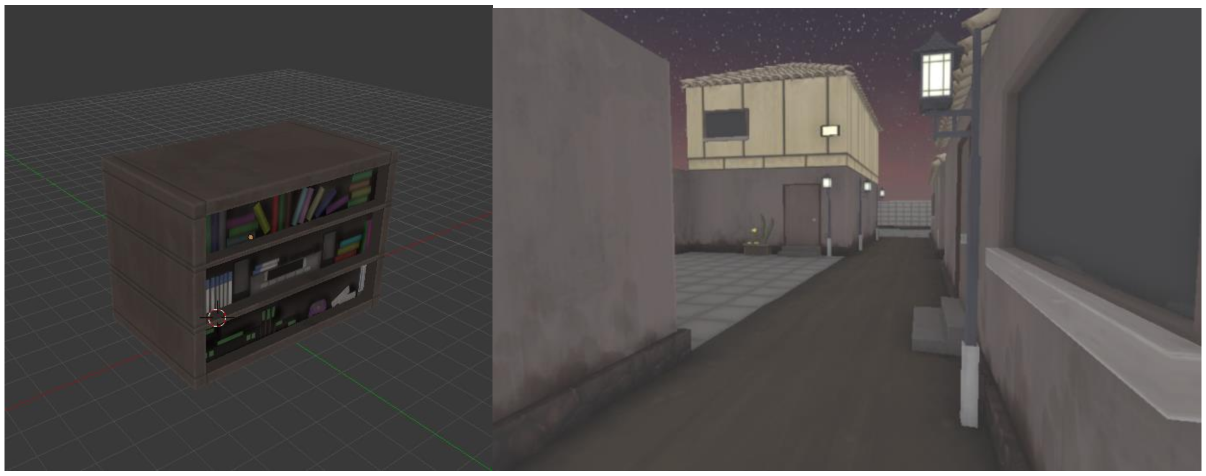


*Figure 9: Some examples of painted textures.*

## 3.4 Animation

All character animations were created inside Blender. After finishing a model, its UVs, and its texture, it would have to be rigged. Some models, like the ‘Wisp’ NPC, required a rig of only two or three bones. Others, like the main character, Rita, required around forty, to account for her regular movements, IK Constraints, and secondary animations like hair and backpack.

Naturally, different characters required different numbers of animations. Rita, as the main character, has seven, which include standing, attacking, and getting hurt. Her ‘walk’ and ‘run’ animations blend between each other, depending on how fast she is moving; this is done in Unreal. Her different ‘idle’ animations also blend smoothly between each other. Other characters, like the Spider and Mouth enemies, have only three animations each, including movement, attacking, and death.

Environmental animations were done in Unreal. The opening of doors in *Chinmoku* is done solely through Unreal movement. Careful placement of the object’s blender origin was necessary to make them turn from the proper point.

# 4 TECHNOLOGY

During the planning phase of *Chinmoku*, the team became increasingly aware of the large scope of the game being planned, and because of this realization, a great deal of importance was placed on the implementation of an intelligently designed, reusable, and extensible code base. The two foundational tech components of *Chinmoku* are the Android SDK, used for the game controller, and the Unreal Engine 4.

## 4.1 UNREAL

Unreal Engine 4 (UE4) is a free, open source game engine written mainly in C++ that asserts an extremely rigid code structure. The fact that the engine is open source, extensible, and extremely rigid are three of the main factors in the team's decision to use UE4 over other viable engine options. Aside from basic game code and scripting, the team made a number of core extensions to the UE4 API specifically for use in meeting the requirements of our game. These extensions include a generic Bluetooth Receiver Component, a Bluetooth Manager, a Hiragana Object, a generic Zone Object type and many derivations of that object. These extensions and their respective UE4 classes are visualized in Figure 10.

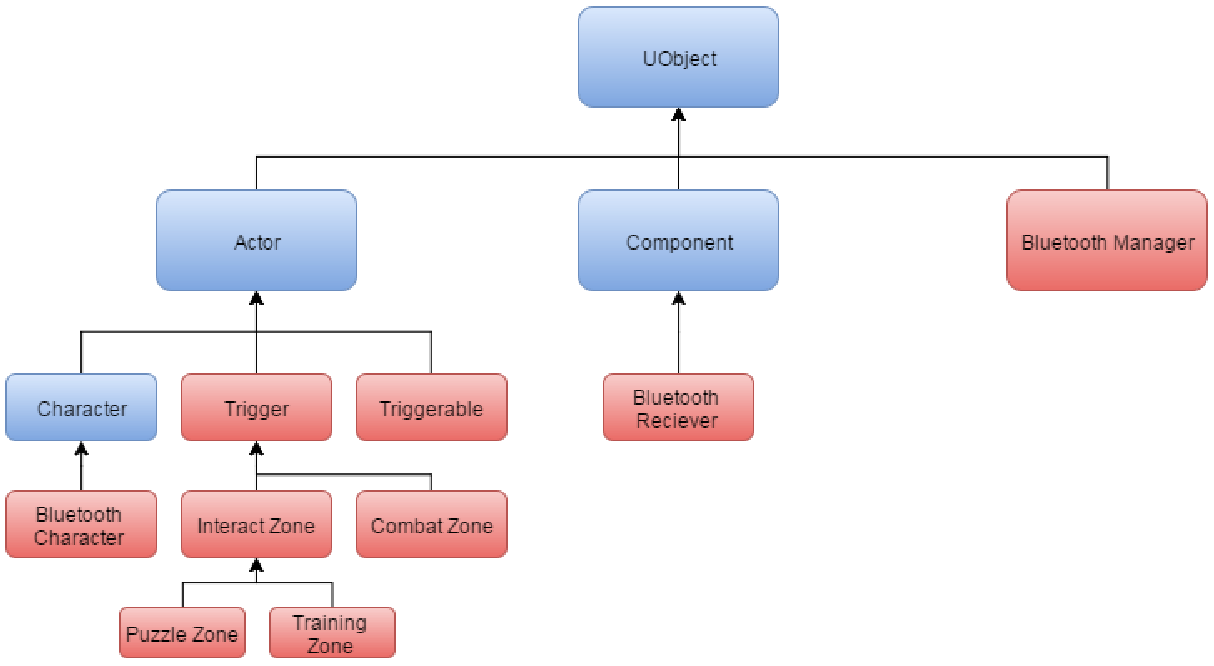


*Figure 10: Graphical summary of custom engine extensions (red) derived from ue4 classes (blue).*

### 4.1.1 Bluetooth (Computer)

The Bluetooth system in UE4 was implemented with the foremost goal of ensuring that the integration of any UE4 actor or entity with Bluetooth be as simple as possible. There were multiple implementations that the team could have followed through with, the most prominent of which were to either create a base Bluetooth class that would be the parent to all Bluetooth objects or to integrate Bluetooth by leveraging UE4's component based structure. The latter of the two choices was most enticing because it ensured that Bluetooth functionality could be attached to any of the already provided UE4 actors and entities, where a Bluetooth parent class meant that the team would have to rewrite much of the functionality already provided by the engine. To leverage the component based system of UE4 two core classes were created, the Bluetooth manager and the Bluetooth component.

The Bluetooth manager is the structural backbone of the Bluetooth network system that drives the interactions from the phone controller. The manager handles the creation and management of a connection thread that listens, connects, and exchanges data with the phone controller. The Bluetooth manager is the only singleton implemented for the game, a choice made out of necessity to ensure that the manager was always accessible to allow the sending of data.  The Bluetooth manager acts as the interface through which the game may communicate with the phone controller and vice versa. To ensure that the creation of a Bluetooth controllable game object is as simple as possible, the “receive” functionality of the Bluetooth manager takes the form of an event driven design pattern. The manager maintains a list of registered Bluetooth Receiver Components to which data received from the network is broadcast. This event driven system ensures that no code needs to be changed in the Bluetooth Manager when new Bluetooth controllable objects are implemented.

A Bluetooth receiver component, unlike the Bluetooth manager is an object that technically exists in the game world. The component may be attached to any actor in UE4 which then provides Bluetooth functionality to that actor. As alluded to in the Bluetooth Manager section, this component can be registered and unregistered to the Bluetooth manager. When registered the Bluetooth manager will call an overridable function called OnBluetoothReceive() on all registered Bluetooth receiver components. The form this implementation takes ensures the compatibility of the interface with any UE4 object because it does not rely on an inheritance hierarchy not fully under the control of the team. In addition, the component also allows for the binding of a function delegate or pointer that may be executed when OnBluetoothReceive() is called. This last detail is invaluable from a programmer’s point of view because instead of having to define a new child of the base Bluetooth component every time different functionality needs to be implemented, one need only assign the address of the function that needs to be called to the component.

### 4.1.2 Hiragana Object

The hiragana object is a class designed to allow more modular and generic design to make development overall faster and more efficient. The hiragana object is essentially just a set of data, containing an English character as a string and image files for the associated romaji and hiragana characters. These hiragana objects can then be accessed throughout the game (for example in the puzzle system). This means content can be designed being told to take in a hiragana object without specifying a particular one at the time, making it easy to switch which one is used in any given place.

Hiragana is one of the alphabets used in Japanese written language. The hiragana image is simply an image file depicting a character from the hiragana alphabet. For example the 'a' character: あ. romaji is a way to represent Japanese characters using the English alphabet. So the character above written in romaji is simply 'a'.

### 4.1.3 Player Movement

The player's movement is implemented using UE4's pre-made character movement controller which provides a set of functionality impossible to recreate given the time constraints. These include stair stepping functionality, an implementation independent of global physics, and a full force based movement system. The main addition to movement was the use of relative movement to allow single joystick input. The relative movement system defines forward movement as always being in the direction that the active camera is looking on the X, Y-axis. While not totally free of issues, the current implementation causes the relative forward direction to only switch from the previous relative forward direction when the player has stopped moving. A possible future implementation is to interpolate from the previous forward movement direction to the next over a defined duration.

### 4.1.4 Zones

Zones are the core abstraction for the level design in *Chinmoku.* All gameplay components are packaged into zones to facilitate the rapid building of levels. As such, a great deal of focus was given to ensuring all zones were editor friendly, providing all the attributes needed to edit gameplay without recompiling.

Zones, technically, are all customized implementations of the trigger boxes. They make use of overlap shapes that trigger some event either defined in code or through Unreal's blueprint system. Some examples of zones are the puzzle zone, combat zone, and camera zones. Each has functionality unique to their purpose which adds to the rigidity of the code structure and ensures a great deal of decoupling between the different zones and their functionality. Because zones need to be fully customizable the base zone actually provides very little functionality, which takes the form of OnActorBeginOverlap and OnActorEndOverlap functions that are meant to be overriden for each zones needs. From there, each unique zone can be sub-classed to provide more and more functionality while still remaining highly decoupled which adds to the stability of the game as a whole.

## 4.2 Android Controller

The smartphone is the intended controller for the game, but is not a necessity. The smartphone facilitates the puzzle design and interactivity of the game overall, by allowing for touch screen features like drawing, and easy dragging. The game in its entirety, however, is a game intended for PC and as such allows players to play the game without a touch screen. This version features the same puzzles adjusted for mouse and keyboard controls.

Apart from the phone being a thematic prop which increases immersion in the game world, the phone also allows for innovative gameplay. The second screen allows us to control where the player is looking and how they interact with both screens. Horror games often rely on lack of visibility to stress and scare the player. Controlling where the player is looking at various moments in the game provides ample opportunity for jump-scares. The second screen also allows us to clear up the main screen by moving UI elements to the phone, which the team believes makes the game visually more appealing and immersive.

The phone is also able to receive text messages and audio recordings. When playing without the phone and touchscreen, these elements are still available but will be activated by keys on the PC's keyboard.

The controller aspect of the game is implemented like any other app for Android, using the android SDK. The android SDK provides a substantial API for quickly implementing many of the features needed for the phone, but unfortunately also greatly increased the complexity of the project due to the SDK's Java implementation. The core components of the controller consist of the Bluetooth Manager, Bluetooth Activity, Puzzle Activity, and Joystick View.

### 4.2.1 Bluetooth (Smartphone)

The Bluetooth Manager of the controller is almost identical to the Bluetooth manager implemented in UE4 in both function and form. Its core difference, aside from being written in Java, is that instead of a list of registered Bluetooth Components, it contains a list of Bluetooth activities to broadcast to. An activity in the android SDK is the parent of all graphical/interacting screens that display on the phone. The Bluetooth activity is our team's custom derivation of the android SDK activity. The Bluetooth Manager is contained in the Bluetooth service, a child of the Service class which is managed by the android OS. The Bluetooth service may be considered a semi-singleton because it is not instantiated and maintained strictly like a globally accessible object, but is instead managed by the android OS so that only one instance may exist at any given time and that instance may run in the background, even upon minimization of the app.

The Bluetooth Activity assumes the role of the Bluetooth receiver component in UE4. It is the base class of all activities that need to interface with the Bluetooth network, both to send and receive. All Bluetooth activities contain a reference to the Bluetooth manager and all contain an overridable function called OnBluetoothReceive(), called when the Bluetooth Manager broadcasts a Bluetooth message, in an event driven manner.

### 4.2.2 Joystick

The Joystick view is the control scheme present on the controller throughout the majority of the game. It is a simulated Joystick that sends direction and scale (distance from the center of the joystick, which translates to speed of movement) in the form of bytes to the PC to drive movement. In addition, when the Joystick view is double tapped it causes an interaction data message to be sent. A great deal of effort was given to ensuring that the joystick view felt accurate to the player when being used. Whenever the joystick view's state differs from the state during its previous "tick" a byte message containing an x float, y float and magnitude of the given vector, all truncated to 3 digits, is sent to the PC. Including the operation code of the Bluetooth payload, this message this totals a size of 11 bytes which is a very small payload that incurs no noticeable delay and therefore makes for an intuitive joystick. It is also important to note that the Joystick makes no assumptions about the object utilizing its data, and therefore can be used to drive any manner of movement that only requires a single joystick (see Figure 11).

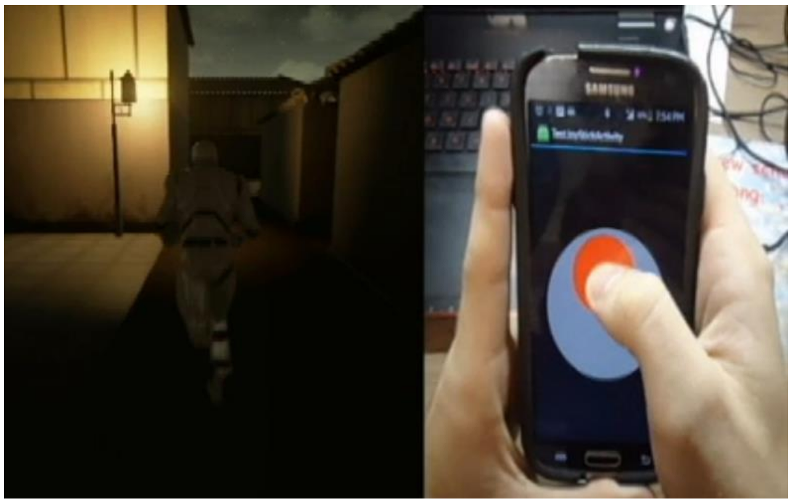


*Figure 11: Testing joystick's analogue movement in an early gameplay build.*

### 4.2.3 Puzzles

Puzzle Activities are graphic intensive activities in the android app that are often created in response to a message being sent from the PC. The base puzzle activity implements an interface, Puzzle Interface, which requires that the class contain an OnPuzzleStateChange(bool isCorrect) function. This function is implemented to determine if, when a puzzle has changed any sort of state, it is correct. When it is correct, a message is sent to the computer from the controller and interactions may continue.

### 4.2.4 Writing Recognition

The writing recognition system consists of all the tools needed to enable combat and training in game. The system itself resides on the android phone and interfaces with the pc game with network messages consisting of an operation code and a payload, usually containing the Unicode string representing the Hiragana character recognized by the system. The writing recognition system consists of four parts, the touch screen painting activity, the recognition system, the machine learning system, and finally the data model representing the recognizable characters. The writing recognition software is a third party library called Zinnia which is written in C and wrapped in a Java native interface for use on the phone.

The painting component of the writing recognition system allows users to draw hiragana characters using only their fingers on the android touch screen. It constructs a set of data from the drawn characters by recording a high resolution composition of points where the user draws on the screen. These points are constructed into strokes for each part of the hiragana characters and the strokes then make full characters which can be processed by the Zinnia recognition system. The construction of character data is a purely in-house implementation that was built to interface with the recognition software.

The recognition component of the software is a c library wrapped in the Java Native Interface (JNI). It takes a Hiragana character object built out of the points the user draws and returns an arbitrary number of best matched characters, matched against the recognition software's data model. The best matched character is what is eventually sent to the pc to drive combat and training interactions.

The Machine Learning aspect of the system is how the software's data model is constructed. As developers we can create custom data models that will be compared against user's input in the same way that the user plays the game. Using a dev tool we can input an arbitrary amount of data for each character using the painting activity to increase the accuracy of the recognition system. The data model currently being used consists of all hiragana characters, but can be expanded to include katakana and kanji (the other Japanese writing systems) if needed. Further, it can also be repurposed to recognize the Latin alphabet.

The data model is a structure defined in the third party Zinnia library and resides in the apps data on the phone. It is easily added to and improved using the dev tools implemented on the phone without the need for a recompile of the app.

# 5 Level Construction

As stated previously, *Chinmoku* was an ambitious project that challenged the time restrictions of our team. In order to complete our goal of finishing a polished prototype we created an assortment of systems and modular assets to facilitate level construction. This chapter discusses level design, the development of modular art and zones, and lighting.

## 5.1 Level Design

The level design in *Chinmoku* largely dictated the pacing of the game. After observing the development team's own memorization capabilities while learning Japanese we were able to estimate the pacing which would be required for players to adequately learn the Hiragana writing system. In the prototype version of the game which the team completed during the duration of our stay in Japan, each level of the game was designed to teach one column of the Hiragana chart. To achieve this, we designed each level so that the player would first encounter a puzzle, in order to practice the column at a relaxed pace and use their skills to accomplish a goal. Ideally, and particularly in the first level, the player would encounter a few puzzles before they were faced with combat. Combat acts as our fast-paced testing of knowledge and is meant to provide a greater challenge and reward to players.

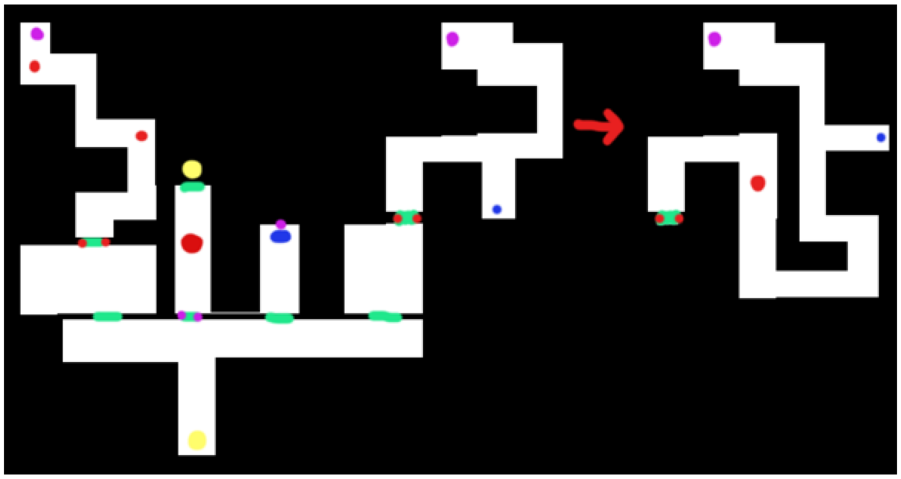


*Figure 12: All levels/maps were initially designed as color-coded 2D blueprints.*

In the above figure depicting a level design draft, green marks door placement while red marked on door indicate locked door that require the solution to a puzzle to open. Red markers in hallways indicate combat zones. In all levels of the game, combat is accessed by unlocking doors and is never directly accessible from the spawn point of a level. The levels are designed in this way to ensure that the player has had ample practice solving puzzles and has had time to get acquainted with the material before they are placed in a do-or-die gameplay state.

When designing levels it also became important for us to think about camera zone placement and camera movement. Initially, camera movement was going to be always facing in one direction (in the above figure example, the camera would be facing 'up' in the direction of the exit) and would pan and move with the player. We realized this would limit our ability to design certain areas of our games and eliminate certain angles and views. Instead, the camera was set to a fixed point in each room or area, panning when necessary. In order for this to work, each area of interest (puzzles, interaction zones, and combat) had to be visible as soon as the player entered the area while keeping the entry/exit doors of the room visible. Hallways and rooms were designed together with the fixed camera angles to provide smooth transitions between areas.

## 5.2 MODULAR ART

In order to create the expansive world of *Chinmoku*, we decided to utilize a modular environment. We created a reservoir of environmental assets- walls, doors, roofs, and floors- that all were measured and designed to fit together seamlessly. These assets, like building blocks, can be assembled in a myriad of ways after duplicating them.

Thanks to the modular nature of the assets, as well as creating different "kits" for creating outdoor and indoor environments, we were able to assemble various rooms in *Chinmoku* given limited resources. Bedrooms, hallways, alleys, and houses could all be assembled quickly and differently. After constructing the levels in Unreal, we then populated them with "clutter". Clutter is the term we used for smaller environmental assets that served to differentiate areas from each other. Shelving, tables, chairs and plants are all examples of clutter and are demonstrated in Figure 13.

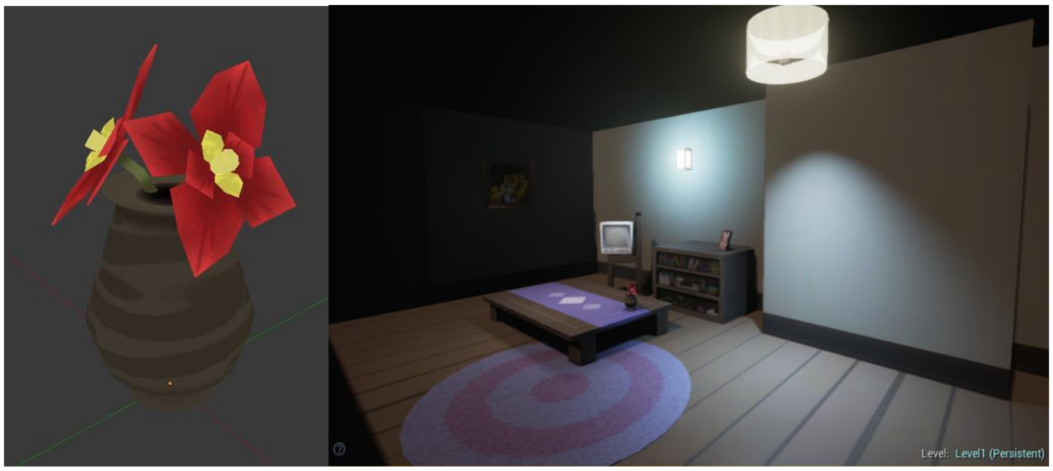


*Figure 13: Close-up of a vase (clutter) item which is then placed into a scene.*

For large-scale environment building blocks, we designed each piece using a tile system. Each tile was measured and constructed so that we were able to assemble them however we wanted without having any visible seams (see Figure 14). This included carefully creating textures for each tile that would flow into one another without showing where the tiles were divided.

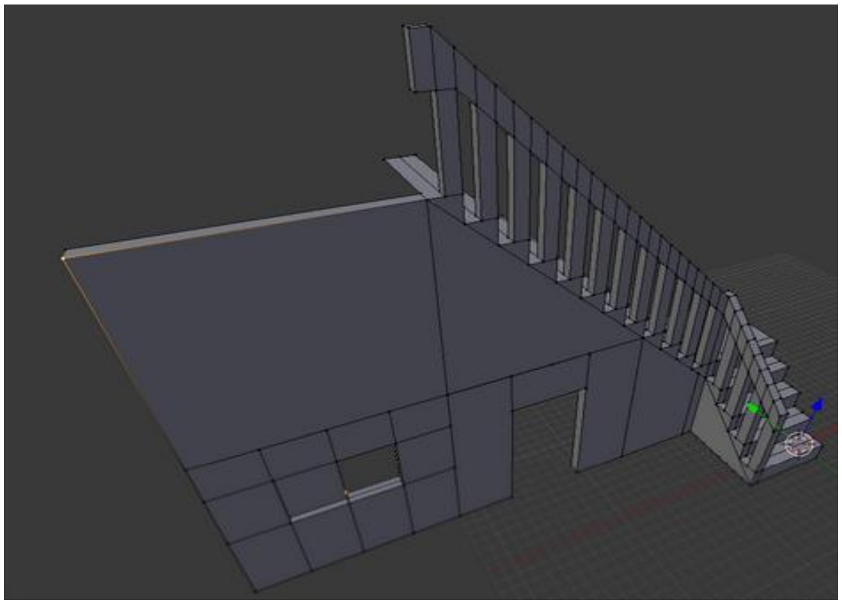


*Figure 14: Tiles for the indoor environments.*

Each tile measures 256x256 units in Unreal. A unit is the equivalent of one centimeter, making each tile approximately 8.4 feet. Using this measurement system, we were able to ensure that each asset was sized appropriately to the rest of the scene and to the height of the player avatar. This was particularly helpful when designing assets such as tables, countertops and interactable objects, which needed to be a certain height in relation the avatar.

## 5.3 Modular Gameplay Zones

Zones are our team's way of creating modular gameplay. Each zone is a self-contained piece of gameplay that can be placed anywhere in a map, adjusted to any size and can be set to expose attributes within the editor to customize the traits of each zone quickly, in a data driven manner. A visual example of zones in a game scene can be observed in Figure 15.

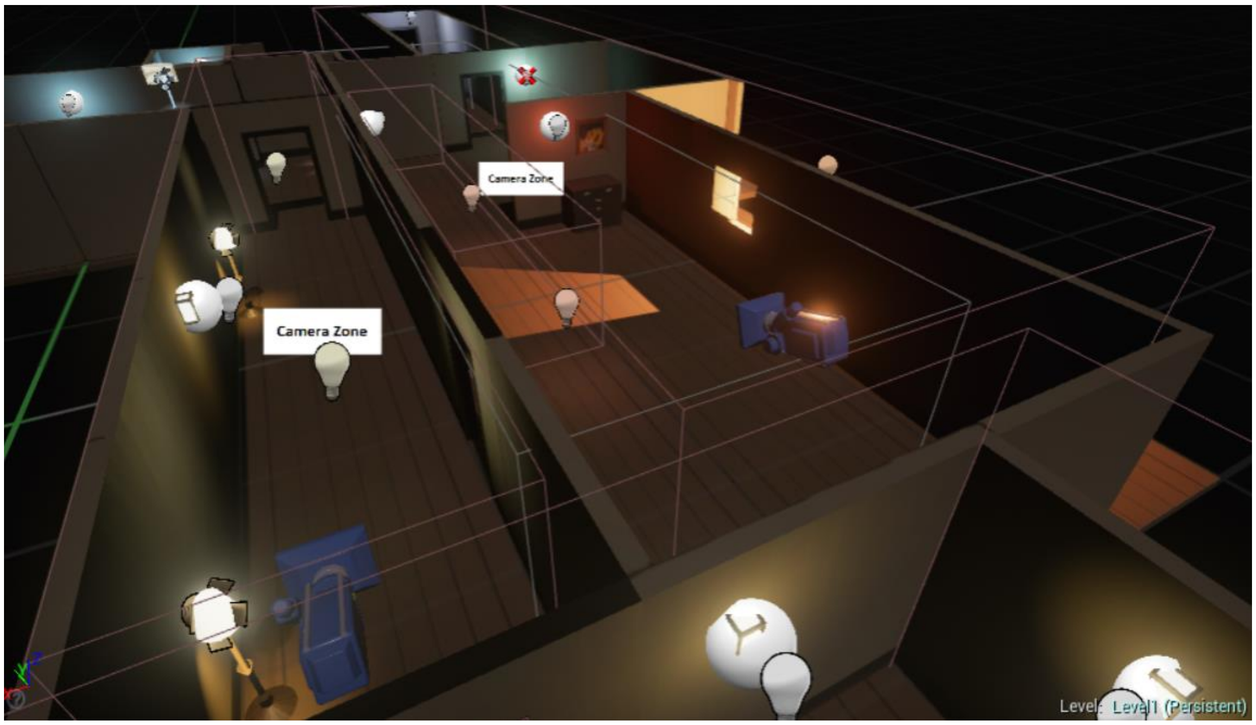


*Figure 15: Zones placed in a level dictate camera movement and gameplay.*

### 5.3.1 Interaction Zones

The most common zone is the interaction zone, which provides functionality for a user to use the interaction button on the keyboard or phone to cause some n-events to occur in game. The majority of zones in-game are derived from the interaction zone, as the majority require the user to perform an interaction. Aside from being interactable, interaction zones have a number of unique characteristics that are also applied to other derived zones. They include the functionality to notify a player that they have entered an interaction zone which is accomplished by playing an animation in which the in-game character lifts up her phone, as well as by sending a notification to the player's phone which causes it to play a sound and vibrate. Interaction zones can also be set as repeatable, or one time use. These zones also hold a list of Triggerable objects that may be added to in the level editor to add extra events that might occur when the box has been interacted with.

### 5.3.2 Camera Zones

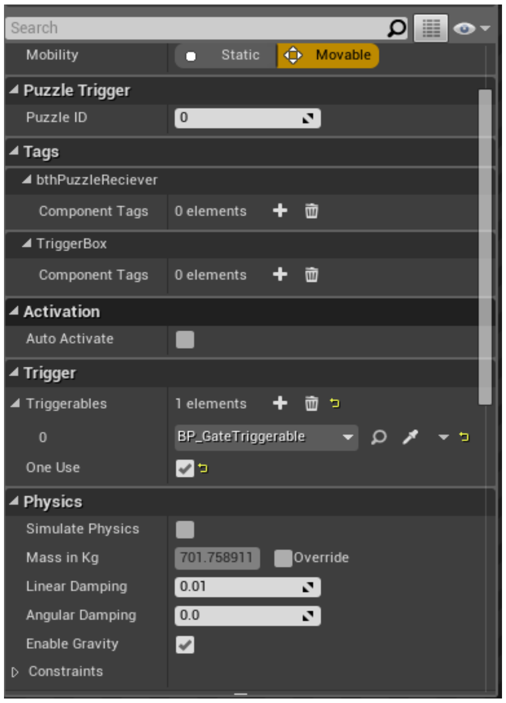
The camera zone is another unique type of zone that requires no interaction from the player, but acts as the distributed manager of the camera system. Camera zones must exist anywhere that the player can move if they are to be seen in that area. Each camera zone has one or more cameras that it associates with in the scene. When a player enters a camera zone it may enable the next available, a random, or the only camera associated with that zone. Camera zones are compatible with any type of camera derived from the dynamic camera base class. The camera zone does not define the behavior of any camera associated with it, but it acts as the interface between the player character and the cameras in the scene, so that they may be enabled and manipulated based on the actions of the in-game character.

#### *5.3.2.1 Camera Movement*

All camera functionality is defined in one base camera called the dynamic camera. This follows the subclass sandbox design pattern in which the camera derived from and specific function calls are made. These include panning with the camera while locking the X, Y, or Z axis, looking at the player, and zooming in on the player. Cameras are attached to camera zones and "turned on" when players enter those camera zones. When the player leaves they are reset and ready to be re-enabled.

### 5.3.3 Puzzle Zones

Puzzle zones are one such zone derived from an interaction zone. In addition to all the functionality of the interaction zone, puzzle zones allow designers to specify which puzzle to start when the zone has been interacted with. Once set, if the player interacts with the puzzle zone, an attempt will be made to start the specified puzzle on the phone, but if no phone is connected the puzzle will start on the pc monitor. The puzzle zone will also not trigger any events upon interaction, but only upon the correct completion of a puzzle. The correct completion of the puzzles is determined via an overridable function that always returns a boolean but must be implemented on a per puzzle basis (see Figure 16 below).

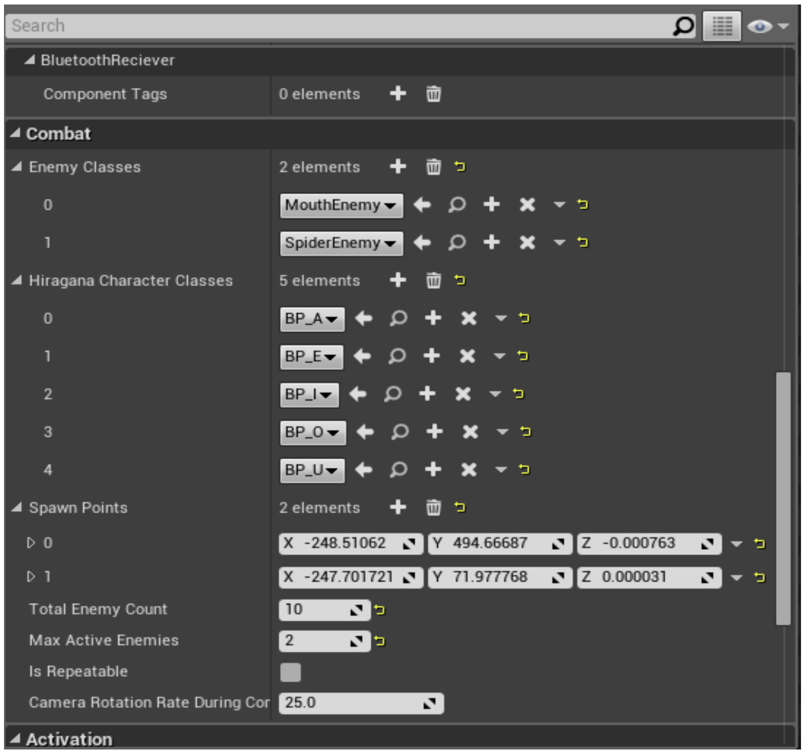


| | |
|---|---|
| Mobility | Static [✓ Movable] |
| **Puzzle Trigger** | |
| Puzzle ID | 0 |
| **Tags** | |
| bthPuzzleReciever | |
| Component Tags | 0 elements |
| TriggerBox | |
| Component Tags | 0 elements |
| **Activation** | |
| Auto Activate | ☐ |
| **Trigger** | |
| Triggerables | 1 elements |
| 0 | BP_GateTriggerable |
| One Use | ✓ |
| **Physics** | |
| Simulate Physics | ☐ |
| Mass in Kg | 701.758911 ☐ Override |
| Linear Damping | 0.01 |
| Angular Damping | 0.0 |
| Enable Gravity | ✓ |
| Constraints | |

*Figure 16: In-game attributes of a puzzle trigger. The triggerable, a gate to open, is set in the Trigger category.*

### 5.3.4 Combat Zones

Combat Zones are the most unique of all zones implemented in the game. The combat zone triggers combat immediately when entered. The combat zone also defines what combat consists of. With-in the attribute browser of this zone the designer can select the types of enemies for the zone to spawn, the number, their spawn points, and finally the set of Hiragana characters to spawn from. Regardless of where the combat zone is placed, the only requirement for functionality is that there be a nav mesh generated. A tool to do so is already provided by unreal engine. Combat zones perform a number of tasks including the management, creation, and deletion of all enemy actors within the zone, as well as the distribution of attacks performed by the players. As seen in Figure 17, these values can be adjusted easily to moderate gameplay difficulty.



| | |
|---|---|
| **BluetoothReciever** | |
| Component Tags | 0 elements |
| **Combat** | |
| Enemy Classes | 2 elements |
| 0 | MouthEnemy |
| 1 | SpiderEnemy |
| Hiragana Character Classes | 5 elements |
| 0 | BP_A |
| 1 | BP_E |
| 2 | BP_I |
| 3 | BP_O |
| 4 | BP_U |
| Spawn Points | 2 elements |
| 0 | X -248.51062 Y 494.66687 Z -0.000763 |
| 1 | X -247.701721 Y 71.977768 Z 0.000031 |
| Total Enemy Count | 10 |
| Max Active Enemies | 2 |
| Is Repeatable | ☐ |
| Camera Rotation Rate During Cor | 25.0 |
| **Activation** | |

Figure 17: The various attributes of a combat zone that can be edited and tweaked in game.

## 5.4 LIGHTING

A huge advantage gained in the use of Unreal Engine 4 is the lighting and rendering system that I provides. UE4 provides dozens of tools directly relevant to lighting that allowed the final implementations of each level to really shine. Knowing the tools at our disposal, lighting was decided to help level design early on in the project. This is done by manipulating the in engine lighting to provide the player with information. The most prominent use of which were to guide the player to objectives and highlight import objects or areas of the game world. In all the demo levels, lighting plays a role, for instance in the introduction level. In the level, lighting is used to guide the player, who is knew to this world, to their objective without introducing any obtrusive arrows, objective markers or text. The idea for the use of lighting in this way came about in the hopes of creating more immersive gameplay.

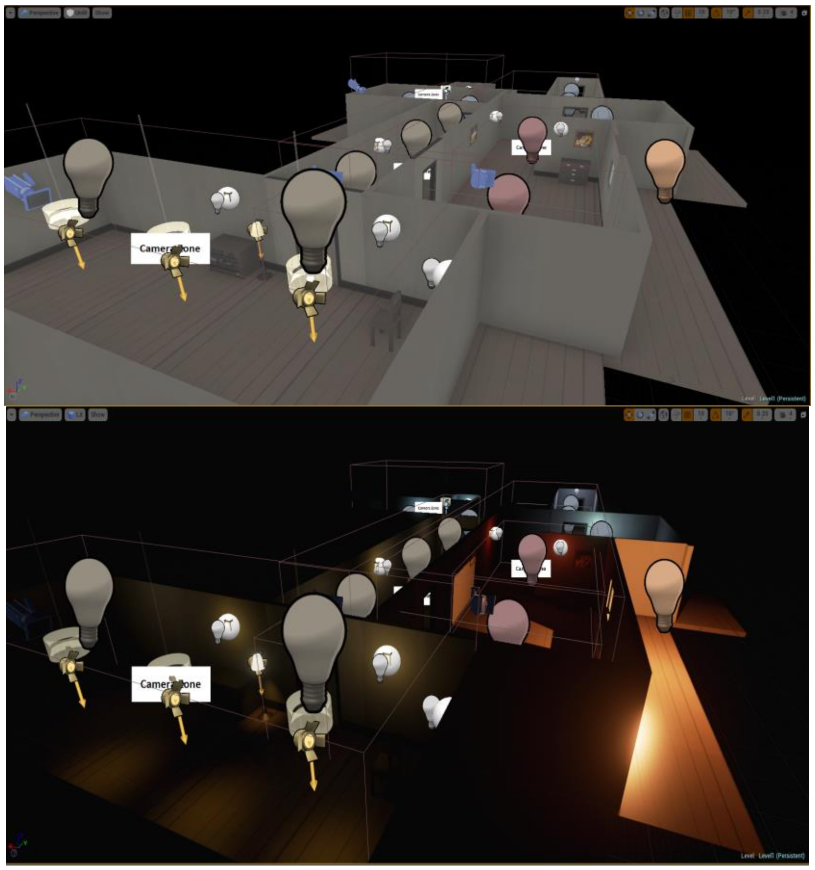


*Figure 18: In-engine lighting comparison. Unlit (top) and lit (bottom).*

Pictured above in Figure 18 is an example of how lighting really brings the levels to life. Also seen are the differences in lighting between specific areas, the red or “hotter” areas mean danger, while the blue or “colder” areas represent safety.

# 6 Playtesting

Playtesting was a vital part of our development process, and one which the team wished we could have spent more time on. Due to the educational nature of *Chinmoku*, pacing was an important aspect that we quickly found was impossible to predict correctly. Though we observed our own learning capabilities, our experience was slightly warped by being immersed in Japan and we understood our target audience would not have this advantage. It also proved difficult to test the pacing of content introduction while we were abroad, because all of our peers were already fluent in Japanese. We therefore play tested and looked at how players interacted with tutorials and instructions, player movement, and other basic controls. We initially did not have any sort of story or tutorial implemented and saw immediately that our players were confused. Tutorial text was quickly added as a remedy to this situation. Other notable problems we became instantly aware of by watching the Japanese students play our game included hard to see font colors and camera panning issues. Though these features have since been adjusted, they still need fine-tuning which we intend to complete after more thorough testing.

Once we returned to our home campus in Worcester, we began playtesting with native English speakers (our target audience). The team was surprised to notice the vast differences between our Japanese testers and our American testers. Certain instructional portions and puzzles that the Japanese students struggled with were barely glanced at by the American students. We explained this by understanding that our game was designed with English speakers in mind, therefore assuming the ability to read the English alphabet. As such, all clues and codes given during puzzles are presented in romaji. Combat has the opposite issue, where our team had been adjusting pacing to match the Japanese students' abilities. In actuality, English speakers who picked up our game for the first time weren't able to use the touch screen and write the Hiragana characters quickly enough for the answer to register. We've since drastically increased the allowed response time for the first level and each subsequent level is only slightly more challenging.

# 7 Postmortems

This section includes postmortems from each of the four development team members. Each member discusses their role in the project and the tasks they were in charge of completing. Completing this project was an enormous task and the team learned many valuable lessons, some through trial and error. With this hindsight, we are able to analyze our work and discuss what we could have done better or what we could have changed to be more productive.

## 7.1 Dillon DeSimone

My personal role in the project was as texture artist and animator. These two roles also required that I UV unwrap the game models, as well as rig them (if necessary). Francesca would create models and pass them off to me. I would give them a quick glance to make sure the topology was acceptable for animating purposes, then would start UV unwrapping. After UV unwrapping, I would texture them and pass them back to Francesca. Some models, like the protagonist and enemies, needed to be rigged and animated, which I would do before passing them to Robert, who implemented the characters. I really enjoyed working with this pipeline- distinct responsibilities and steps made sure that everyone kept busy and knew what they had to do.

Producing the textures for numerous assets across *Chinmoku's* creation was an excellent experience. The factory-line approach allowed me to pump out several high-quality textures a day, and really gain a momentum that would aid me in production. Animation was the same way, rigging became easier, and I could effectively rig characters in short time. Animation was by far the most enjoyable portion of the project, but always time-consuming. Experimenting with each character's movements and fine-tuning them took lots of my time, but was always worth the result. Thankfully, we never had any massive issues while exporting animated assets into Unreal. A few hiccups occurred here and there, but they were obvious and easy to fix on my end.

My only regret is that I did not start asset production sooner! Many of our early weeks were spent designing and deliberating on what *Chinmoku* would be and how it would play. This time was necessary, of course, but I absolutely wish that I had started making rudimentary assets and greyboxes earlier. When it came time to actually start production, we were nearly a third done with our time in Japan, and the asset list looked tremendous. Near the end, when we were pressed for time, I felt rushed and worried that it would show in my assets

Perhaps this was unavoidable, since it was such a short project time. *Chinmoku* needed lots of design and layout, and the time we spent concepting early on did not go to waste. Overall, I learned a ton while working in Japan, and became confident in my level of contribution to a big project.

## 7.2 Robert McKenna

Throughout the course of this project I planned, managed, and iterated over the high level architecture of our extensions to the Unreal Engine 4, i.e. gameplay components and scripting. In addition, I also planned, designed, and implemented much of the low level architecture in Unreal, including all Bluetooth integrated components, as well as the Bluetooth system itself. Many of the zones were also created by me, such as the combat, interact, and puzzle zones, a number of which integrated directly with Sam's work. In regards to the controller, all of the framework, and the app itself were planned and implemented by me. These include the writing recognition system, joystick, and anything involving Bluetooth on the phone's end. I also ported all of the puzzles that Sam completed to the android phone, for easy use with the touch screen.

In recent years, as a Computer Scientist, I have begun to take an enthusiastic interest in software architecture and programming patterns. It was over the course of this project that I really got to exercise my knowledge in these fields due to the project's size. I felt as though the application of my skills was both appropriate to this project and effective. The modular gameplay system proved to be rigid enough for artists to make use of without introducing errors, while flexible enough to provide unique game play. The Bluetooth system also proved to be fast and easy to integrate, while also remaining nearly bug free. Given the time constraints, I believe my focus on extremely modular code, which made use of a number of design patterns, ensured that we had the tools to rapidly build content.

Although the tech of *Chinmoku* has proven to be a success, the experience has not been without problems. But these problems also presented invaluable learning experiences. Because much of my own time was spent working on tools or extensions to the engine for use in our game less gameplay was actually implemented than should have been possible. This is unfortunate because it became a tug-of-war between new/ improved features or new content. Looking back, I may have opted to keep the code simpler in favor of building and scripting actual gameplay. I'm sure that, on a larger team and given more time my additions could have been very useful and well utilized. The error in my planning was treating the resources at our disposal as being greater than they were. So, in a sentence, I believe I scoped too big in regards to tech. This being said though, I am still very happy with the sheer amount of interesting and useful features that were implemented. An amount I do not believe would have been possible without the careful attention paid to the architecture of our code base and the strict adherence to the ideals of a pragmatic programmer.

## 7.3 Samuel Wallach

My role on the project primarily involved design and implementation of puzzles and a puzzle creation system. I also worked on the trigger boxes used in much of the interaction the player has with the game. I worked closely with Robert to make sure systems designed for the phone control properly interacted with the systems on the computer. Alongside those aspects, I did some work creating the interface for the in-game desktop that is intractable in the hub area.

Over the course of the project I learned a lot about organizing code to avoid extra work in the future. Since the code had to interact with the phone application, it required forethought to make sure everything worked. Going forward, I intend to use what I've learned to make coding in a team go more smoothly, as well as making even individual projects require less overall code through more careful planning.

Early in development I did not think ahead enough about the organization of the code and how it would interact with other systems. I especially spent too little time thinking about how my systems would interact with systems I did not write. This meant that earlier code I wrote needed to be reworked quite a bit when the time came to put it together with Robert's code. It took until decently far into the project for me to properly consider everything that a system would do before I started working on it, but once I learned to do that work went much more smoothly.

Relatedly, some systems I worked on had to go through multiple iterations, simply because I hadn't considered the potential issues they might have thoroughly enough. As the project went on, I got better at predicting potential issues in advance and designing the code with said issues in mind.

## 7.4 Francesca Carletto-Leon

Being abroad is always an incredible experience full of excitement, discomfort, and learning. Working on a project in Japan was definitely all of these things. Going into this project I had expected and prepared for all sorts of things, but managing a team of my peers was not one of them. While in Japan I took care of all team management aspects of the project, including creating and maintaining a Slack channel, Google Drive, Dropbox folder, and Gantt chart. It was my responsibility to schedule meetings and communicate with our advisor, who was out of country for most of the project's duration. I also prepared weekly meeting agendas and reports, including gameplay videos demonstrating updates made to the game. Though I believe I was able to manage the team well and learned a lot through doing so, at times the stress of it got to be and hindered my productivity and enjoyment of the project. In hindsight, this is something that I believe a team leader needs to be able to harness in order to not stress or become irritated with the other team members, as I did at some points in the project. I learned to manage the team through understanding and motivation, similarly to how we sought to motivate our players through positive reinforcement and rewards.

I was also responsible for creating the majority of the 3D models for the game, including all character/enemy models and environmental clutter items. While we were able to create an extraordinary number of assets in our short time in Japan, I believe we could have been even more productive had we first implemented standards for 3D modeling procedures and done a bit of research in regards to importing assets into Unreal. We had a few hiccups where some minimalistic error was made (a box wasn’t checked, etc.) and all assets had to be passed back to the artists to correct. This included mistakes with sizing the models for importation and flipped normals, which all had to be passed back for adjusting, which cost us ample time.

I’m extremely proud of our project and believe we handled the stress and tight schedule extremely well for never having been exposed to this type of work environment/situation previously and I can’t wait to show *Chinmoku* to the world!

## REFERENCES

(2015). Retrieved from Fun 4 The Brain: www.fun4thebrain.com

Blair, M. (2012, December 16). *When Creating Educational Video Games, Make Education a Second Priority*. Retrieved from BostInno: www.bostinno.streetwise.co

Chatfield, T. (2010). *7 Ways Games Reward the Brain.* TED Talks.

Farrington, J. (2012, July 20). *Wii U: Can the GamePad Change Console Game Design?* Retrieved from DorkShelf: www.dorkshelf.com

Griffiths, D. M. (2002). *Educational Benefits of Video Games.* Nottingham Trent University.

Harris, C. (2009, February 27). *The Evolution of Nintendo Dual Screen*. Retrieved from IGN: www.ign.com

Klein, D. J., Freeman, D. J., Harding, D., & Teffahi, A. (2014). *Assessing the Impact of Second Screen.* Technologia.

McGonigal, J. (2010). *Gaming Can Make a Better World.* TED Talks.

Programme, T. N. (2013). *Game-based Learning: Latest Evidence and Future Directions.*

Resident Evil. (1996). Video game: Capcom.

Romero, B. (2011). Gaming for Understanding. TED Talks.

Silent Hill. (2006). Video game: Konami, Sony Computer Entertainment.

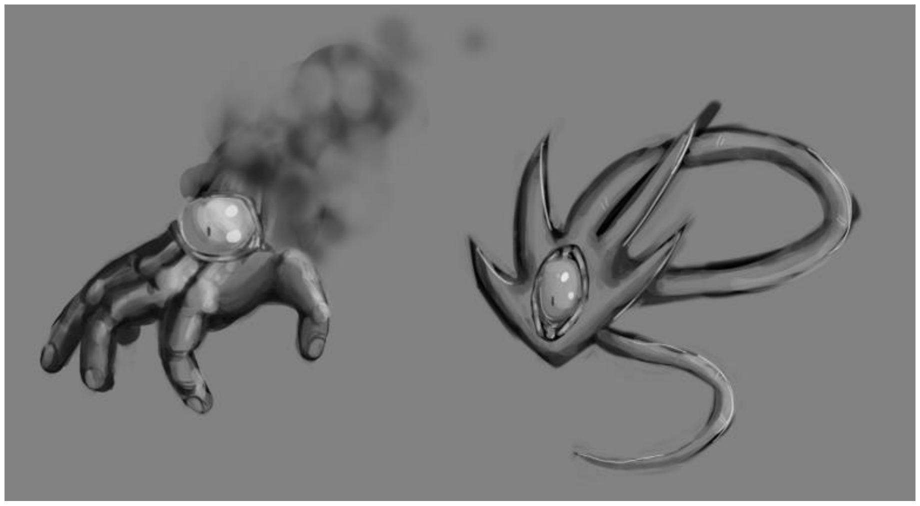
Steinhardt School of Culture, Education, and Human Development. (2013). *Educational Video Games Can Boost Motivation To Learn.* New York University.

Tekofsky, S. (2010). *Theory of Gaming Motivation*. Retrieved from Think Feel Play: www.thinkfeelplay.com

*Zinnia: Online hand recognition system with machine learning*. (n.d.). Retrieved from http://taku910.github.io/zinnia/

Zytnik, M. (2014, Sept. 29). *School Report for Apple App Store and Google Play*. Retrieved from Adjust: www.adjust.com

## Appendix A: Art Processes and Concepts



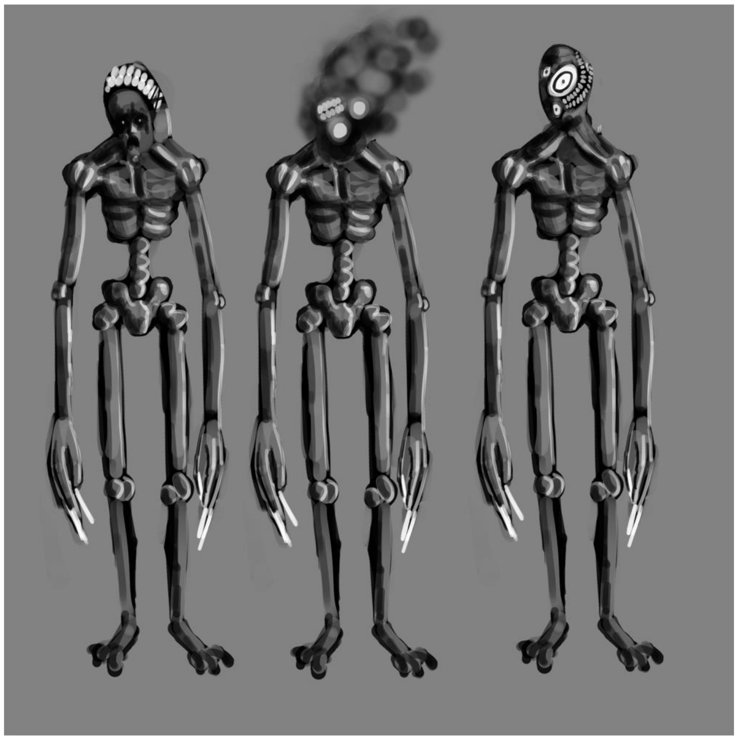
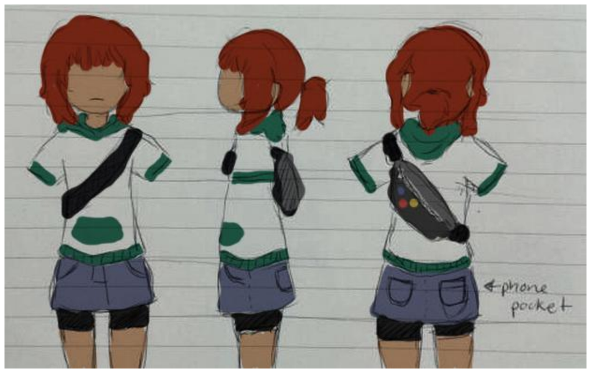


*Figure 19: Concept art and color tests for enemies (Some unused).*



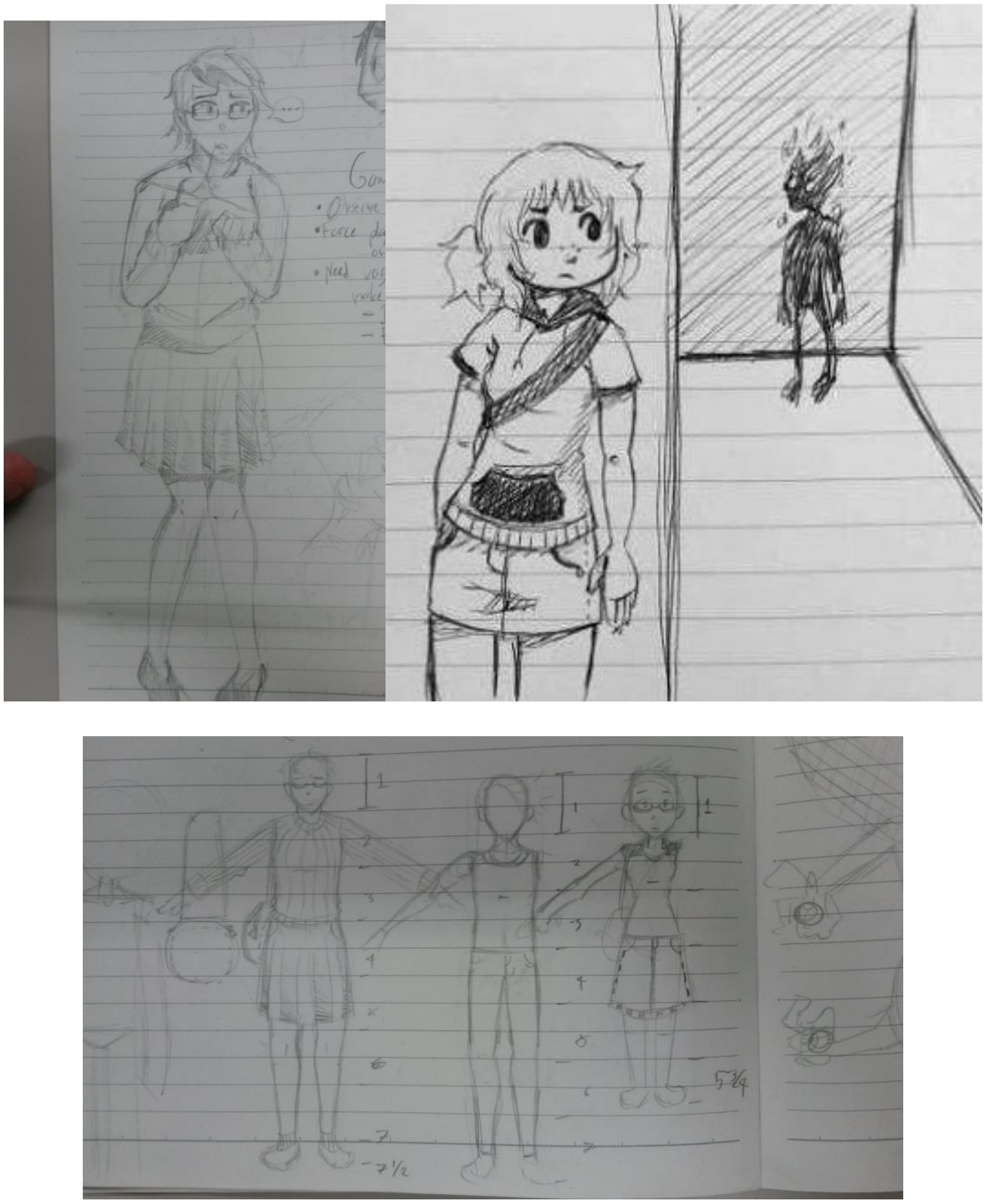


*Figure 20: Concept art, color tests, and proportions for the player character.*

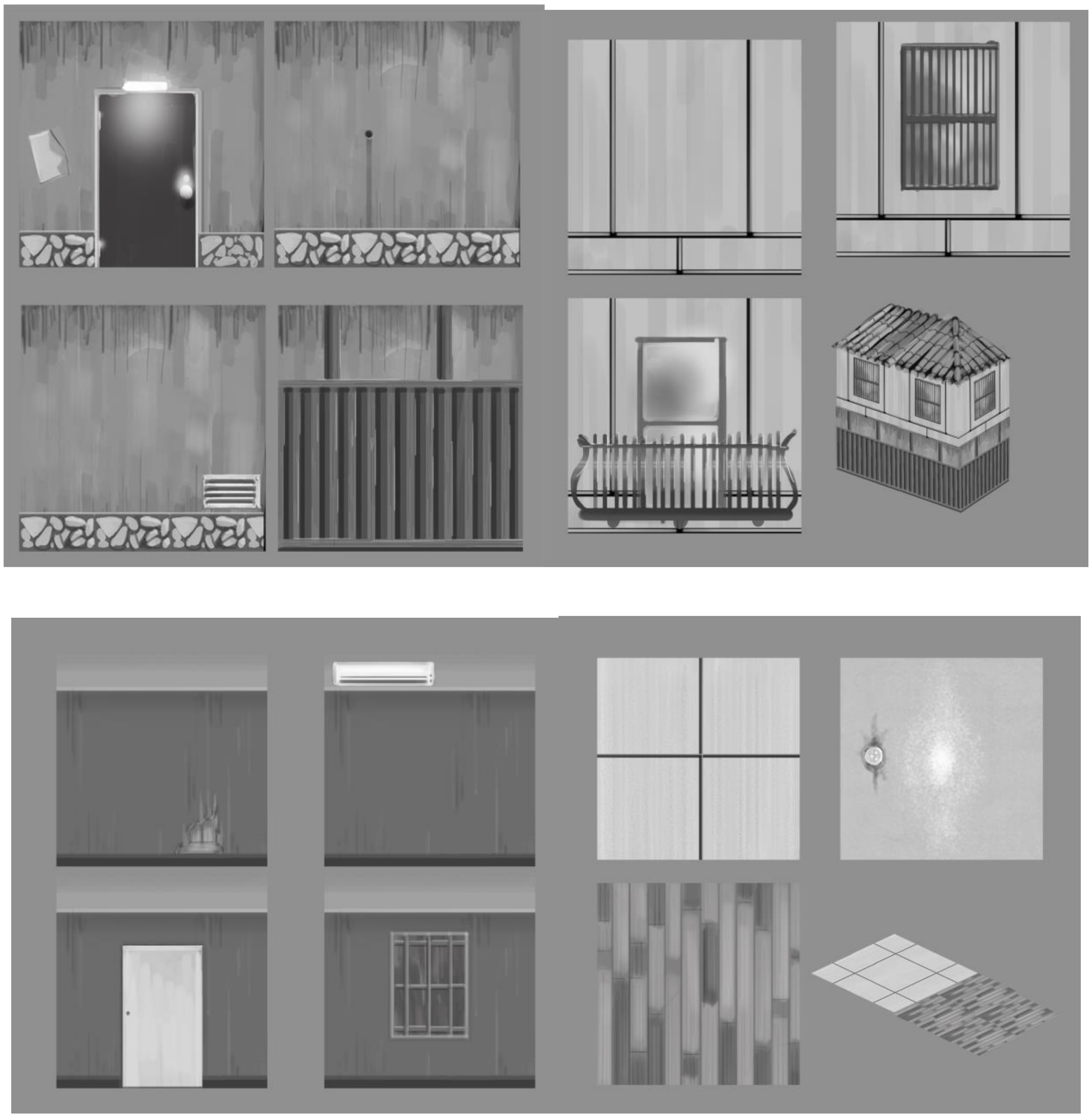
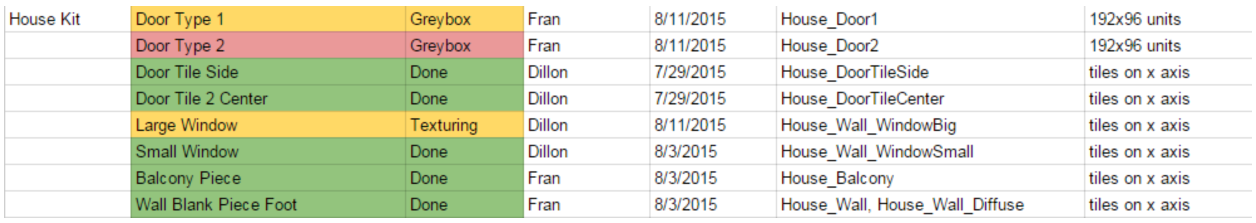


*Figure 21: Environmental concept for indoors and outdoors*



| | | | | | | |
|---|---|---|---|---|---|---|
| House Kit | Door Type 1 | Greybox | Fran | 8/11/2015 | House_Door1 | 192x96 units |
| | Door Type 2 | Greybox | Fran | 8/11/2015 | House_Door2 | 192x96 units |
| | Door Tile Side | Done | Dillon | 7/29/2015 | House_DoorTileSide | tiles on x axis |
| | Door Tile 2 Center | Done | Dillon | 7/29/2015 | House_DoorTileCenter | tiles on x axis |
| | Large Window | Texturing | Dillon | 8/11/2015 | House_Wall_WindowBig | tiles on x axis |
| | Small Window | Done | Dillon | 8/3/2015 | House_Wall_WindowSmall | tiles on x axis |
| | Balcony Piece | Done | Fran | 8/3/2015 | House_Balcony | tiles on x axis |
| | Wall Blank Piece Foot | Done | Fran | 8/3/2015 | House_Wall, House_Wall_Diffuse | tiles on x axis |

*Figure 22: Excerpt from the asset list.*

# APPENDIX B: ADVISOR MEETING AGENDAS

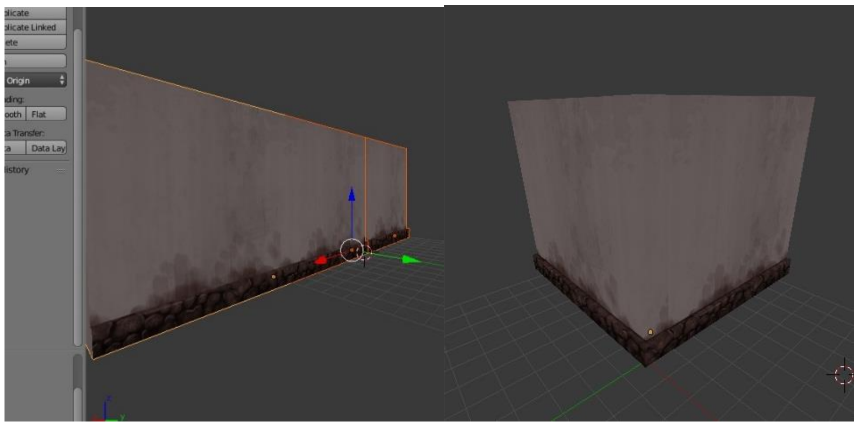
**8/18 Meeting Agenda**
What we've been up to:

- Hanabi with Japanese lab students!
- Presented project idea and progress to lab students and professors
- Vacation in Tokyo!
- Japanese classes start tomorrow!

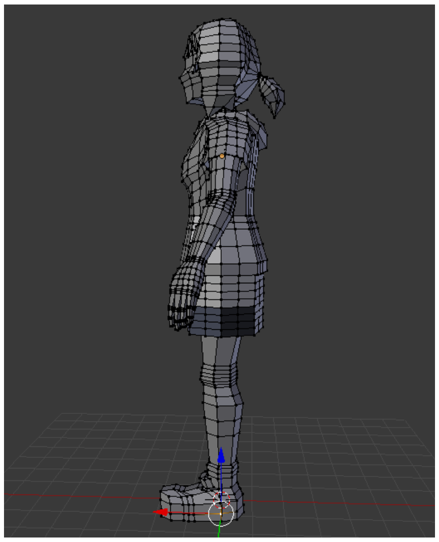
General progress report:

- Continued documentation of research and work
    - Google Drive now has a tech writing section
    - Research was done on other writing recognition apps
    - Updated and expanded game document to currently reflect design and goals

- Environmental art assets under construction
    - Asset list fully updated
    - Created base models of modular house building kit
        - Testing texture tiling and colors



- Character protag model just about finished
    - Added more edge-loops to allow for twist bones and bending
    - Entirely redid topology on shoulders so model's arms can be lifted without too much deformation



- Tech Demos!
  - First puzzle is entirely functional
    - Was built originally with a single hard coded answer but now can be adjusted to utilize any hiragana
  - Created analogue stick for android device
    - Highly responsive
    - Functional in game, demonstrating the utility of the Bluetooth Framework
  - Bluetooth Framework is at a much more developed stage.
    - No actual Bluetooth code needs to be touched to integrate and scene, character, or app into the bluetooth system.
    - Event driven, meaning that resources aren’t wasted, ensure scalability of the game as more and more ideas are implemented.

**8/25 Meeting Agenda**

What we've been up to:

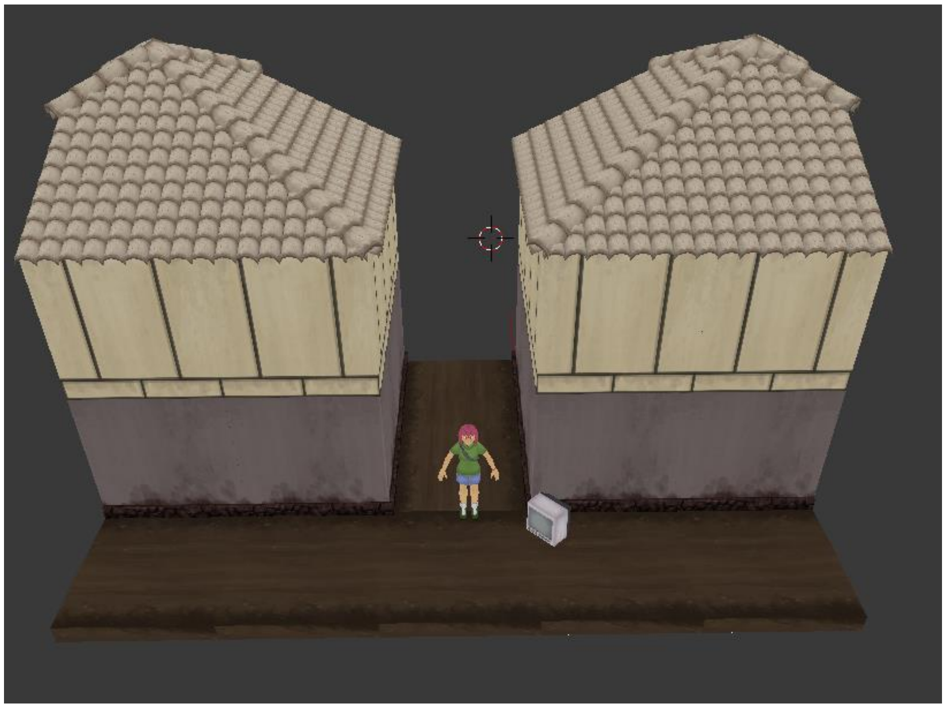
- Japanese language bootcamp complete!

Completed goals from last week:

- Character model finished and rigged, begin work on enemy/NPC models
- Begin texturing modular kits so map building utilizing final assets can begin
- Continue modeling environmental objects and 'clutter'
- Analogue joystick working on android with bluetooth
- Writing recognition prototyped

Modular Art:

- Outdoor house kit almost completed
  - Need to add small details like window and door variants
- Tile textures coming along nicely
  - Corner pieces still needed for ground textures



Character/Object Art:

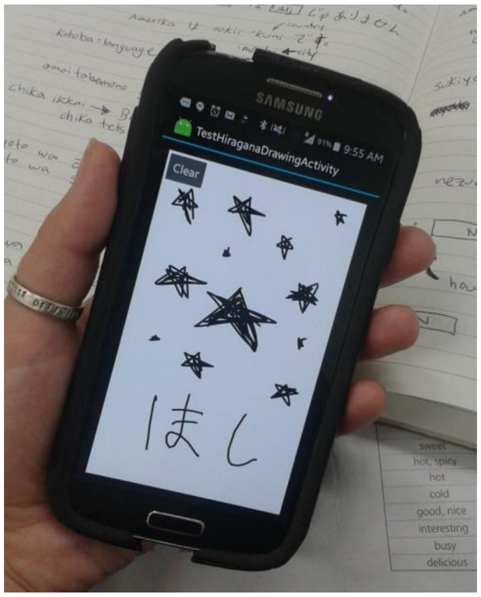
- Character model finished and rigged
- Character texture first draft completed
  - Colors may change to fit with surrounding environment's pallette (proposed color ideas in the google drive)
- Hub environment art and indoor/outdoor clutter is getting cranked out!

Analogue joystick on Android device:

- Works directionally with minimal delay
    - Reworked initial prototype to shorter delay considerably
- Next step is to implement 'analogue' style movement with speed variation

Writing recognition:

- Rudimentary writing and 'clear' button functional



Puzzle implementation:

- Reworked entire puzzle system for more efficient implementation in the future
    - Previous version
        - Utilized UMG (built-in Unreal system for interfaces) and has to be edited through blueprints exclusively

        - Switching to a hiragana ‘object’ instead of hard-coding in puzzle questions and answers complicated things
    - New version
        - Code-based and easier to edit
        - More generic and less work has to be redone to implement each puzzle

**9/9 Meeting Agenda**
**Goals for this week:**

- Character model
  - Rework texture
  - Finish all animations
- Enemy models
  - Finish modeling the 2 small monster variants (1 finished!)
  - Texture and rig
- Intro area
  - Continue to fill with clutter and add detail
  - Create vowel 'hints' to be placed in level (designed but not added)
- Hub environment
  - Blocked out and mostly filled with final assets
- SUPER MEGA COMBAT WEEK
  - Start and finish combat 'zones' so that they can be placed in levels
  - Enemy movement / health system / writing attacks
- Puzzles
  - Puzzle implementation has been made faster and easier
  - Crank out 2 new puzzles (this includes both pc and android versions)
  - Polish previously made puzzles
  - Create list of all assets needed for phone/computer puzzle UI

**Misc.**

- Sent model/Texture work to Prof. Sutter for feedback but he forgot to reply. Follow-up e-mail sent.
- Scheduled a Skype meeting with Berklee students for later this week
  - They've said they can provide us with a minimum of three unique tracks and possibly sound design

**Art!**

- Finished hub environment building kit
  - Walls, wall variants (doors, windows), floor, and staircase finished
- Majority of Hub furniture completely done
  - Remaining: Computer (hero object), kitchen cabinetry and clutter, doors)
- Created more clutter items for outdoor/indoor environments
- Adjusted protagonist texture to better fit environment color scheme

**Tech!**
Super Mega Combat Week!

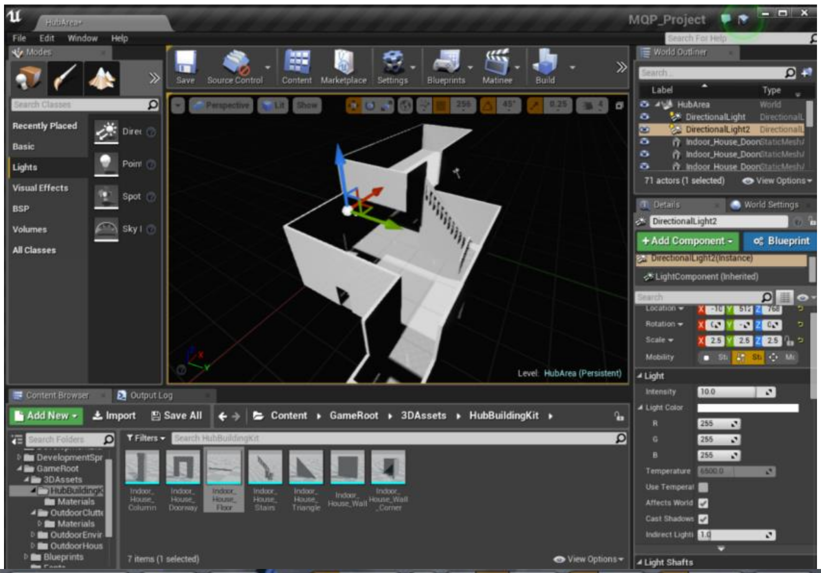
- Combat is functional and extremely easy to implement!
  - Combat editor finished!
- Enemy spawn/movement/destruction working.
  - Next up is putting in enemy/protagonist models and animations and creating health system for player

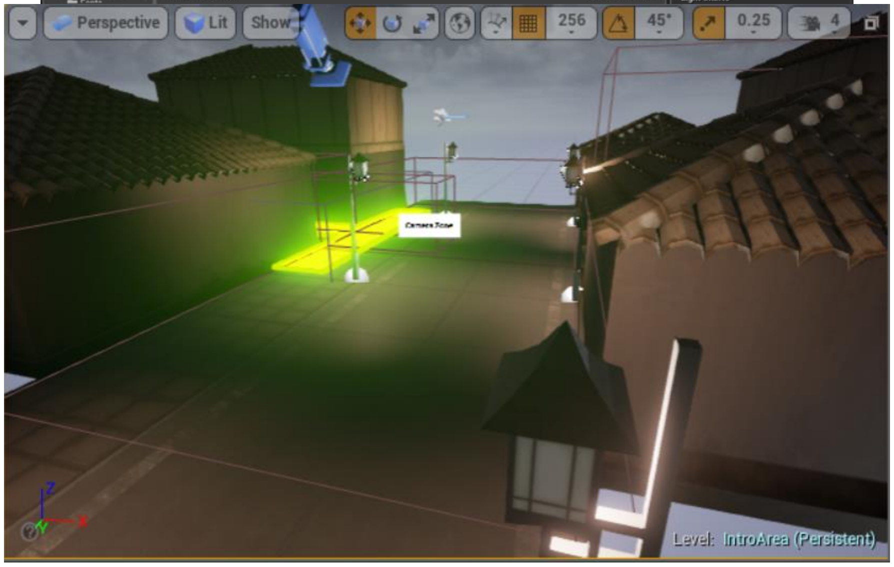
**Puzzles!**

- Two more puzzles created
  - Lock-pick puzzle: Players must follow a sequence and turn the lock-pick to the correct characters to unlock a door/chest
  - Sliding tile puzzle: Players must arrange the characters or a column in the correct order by moving misplaced tiles to the top of the stack

**Map Creation**

- Hub area is blocked out
  - Ran into some tiling issues because of the stairs but it's been resolved!
- Worked on lighting in Intro area
  - Street lamps added





Building the hub in engine and placing zones in Intro area.

**9/16 Meeting Agenda**

**Goals for This Week:**

- Character model
  - Finish all animations, send to Sutter for feedback, revise if needed
  - Test all animations in engine
- Enemy models
  - Finish second enemy model
  - Rig and texture
- Hub area
  - Finish all Hub assets, textures, and lighting
- Level building
  - Block out first two
- Puzzles
  - 2 more puzzles finished!
  - Create android versions of last week's puzzles
  - Begin working on 2D assets for puzzle UI
- Begin planning any scripted events needed in intro of game
  - Opening cut-scene, etc.)

**Misc.**

- Received some feedback from Sutter concerning textures
  - Suggested ambient occlusion baking for some of the textures
- Wasn't able to Skype with Berklee students but have been e-mailing and they're already drafting up some samples for us to give feedback on
  - They're also doing sound design for us!
- Another Berklee student who reached out to us might be interested in recording a few voice-over lines for our protagonist
- Intro to game is storyboarded and in the Google Drive

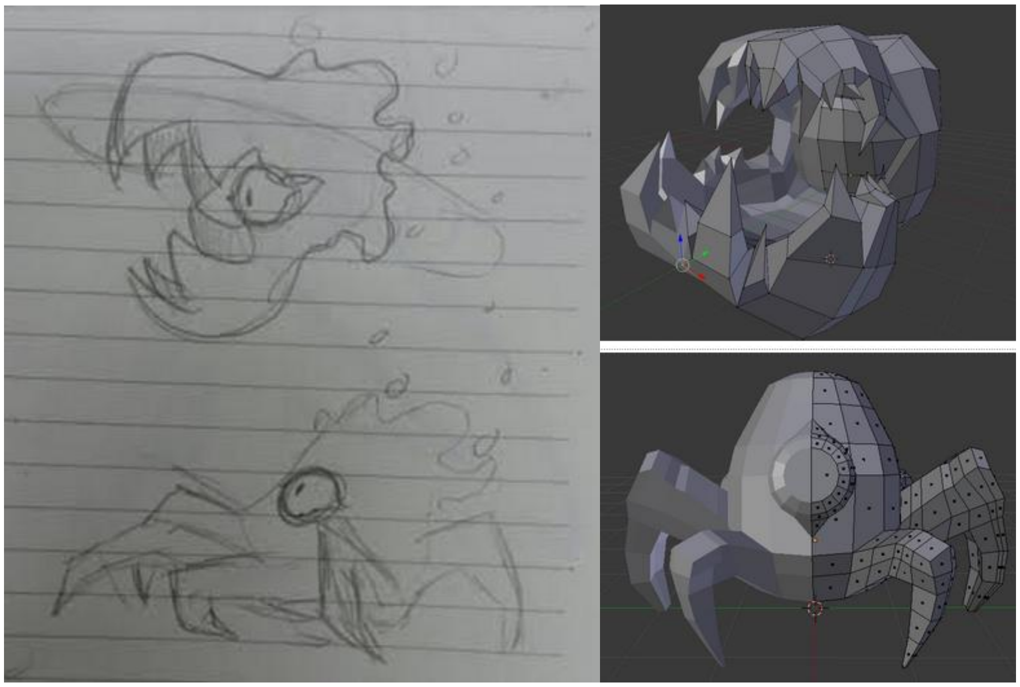
**Things from Last Meeting**

- How to interact with the computer?
  - Phone will be used as a touchpad, double click to interact
- How 'training' works?
  - After players complete a level they earn a key and return to the hub
  - The key opens a new room, in which the player receives a document/scroll that awards them a new column
  - The player must trace each character three times and then they are able to access the next level

**Art!**

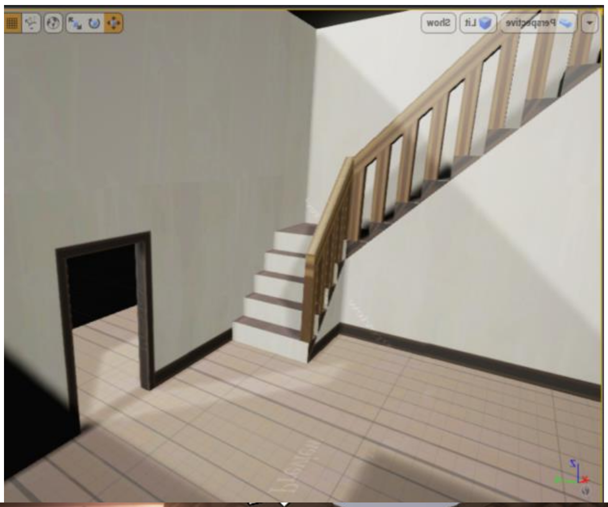
- Protagonist texture got some more touch-ups
  - Sweater pockets
  - Changed eye size (Sutter's suggestion)
- Protagonist animations almost finished

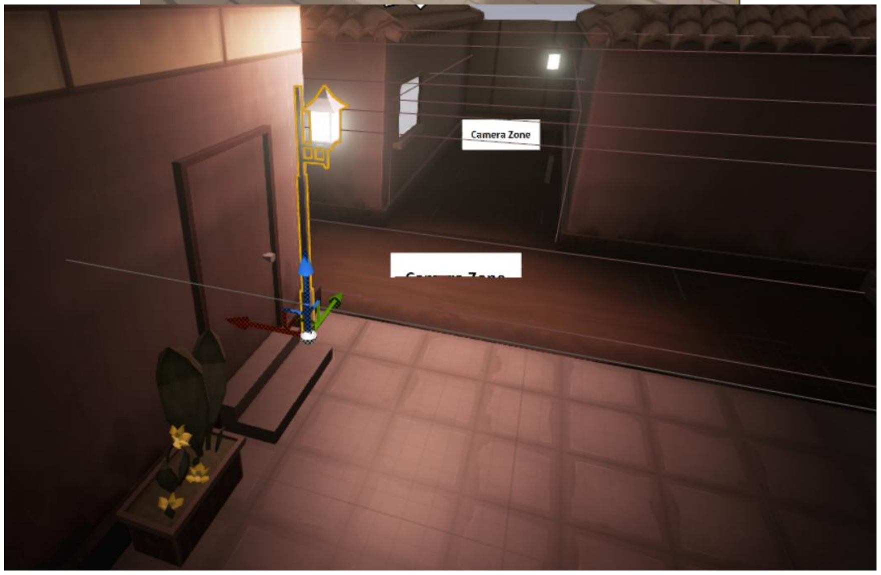
    - A few tweaks need to be made
- Small enemy models for beginning areas done!



- All Hub assets are done!
    - Computer textured and materials applied so we can start working on the screen HUD system
    - All assets imported at correct scale and materials made

**Level Creation**
- Blocked out first two levels in Unreal
    - Placed floor tiles and marked where interaction boxes need to be placed
- Rearranged Intro area slightly and added more variation to environment
    - Doors/windows/plants/other clutter added
    - Lighting still needs to be worked on





**Gameplay (puzzles/combat)**

- Adjusted combat so that any/all hiragana characters can be used
- Fixed scaling issue with different sized android phone screens
- Previous computer puzzles (lock-pick and stick placement) have been implemented for phone controller
- Two new puzzles
    - Slider Picture Puzzle - Tradition slider puzzle to form a picture. Makes Hiragana
    - Word Matching - Match Hiragana Word to Romaji transliteration

**9/23 Meeting Agenda**

**Goals for This Week:**

- Character models
  - Protag/2 small enemies/NPC
    - modeled, textured, rigged, animated
  - Imported into Unreal engine
- First two levels
  - Place camera zones. combat zones, puzzle zones, lighting
- Training system for learning new columns
- Implement player health for combat
  - Combat should use final enemy models and animations
- Script intro sequence (e-mails)
- Polish current puzzles and implement 2D assets

**Also Finished:**

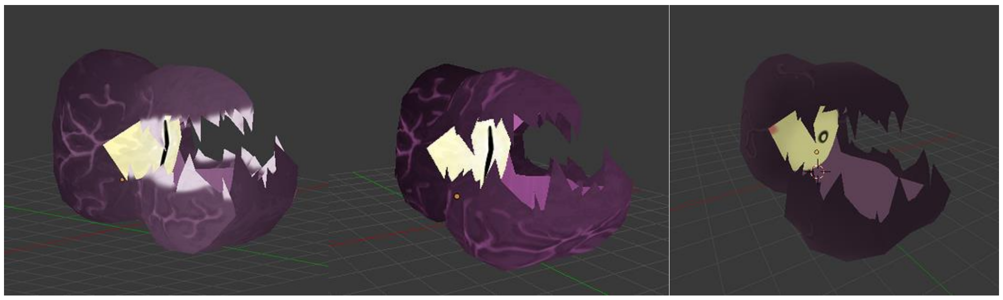
- Dialogue box implementation

**Audio:**

- Started google spreadsheet with Berklee sound team
  - Filling out audio asset list for SFX
- Wrote up voice lines for protag
  - Sticking with the minimum needed for tutorial/ambience
    - Ex. “It’s locked.”, breathing sounds

**Character Models:**

- Protagonist animations imported into enginel
  - Adjustments need to be made to walking/run speed and blending
- Simplified NPC “spirit” modeled, textured, and animated
- Enemy models
  - Tried a number of different textures with the goal of making the enemies very visible even from a distance

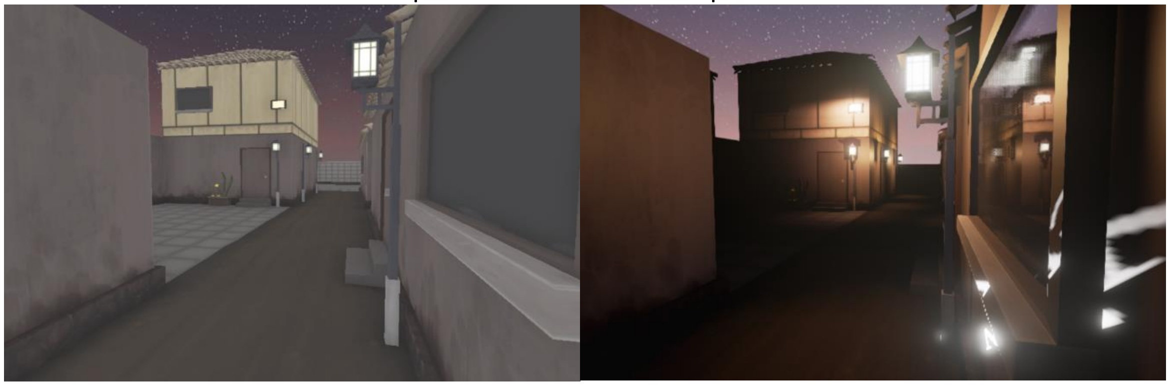


**Level Creation:**

- There are now transitions between levels! They work!
  - Bluetooth functionality carries between levels.
  - Relatively fast load but still need some sort of visual transitions (fade or other)
- Lighting and materials are almost final.

(more photos in the asset masterpost, as always)

Here's an example of an unlit scene compared to a lit scene.



**Combat:**

- Now has actualy enemy models, which attack when close to player character
- Player character now has health and is able to die

Puzzles:

- When a player enter an 'interaction zone' they can click to interact with the puzzle
    - Players are never forced into puzzles as they are with combat
- Working to polish puzzles and implement final 2d assets!

**Polish Goal:**

Our goal is to have the Intro Area, Hub, Level 1, and Level 2 finished and polished by October 9th! This goal includes a total of six puzzles and 3 hiragana columns (including the vowels).

The next two weeks will be spent polishing our existing levels. We'll be tweaking existing assets and continuing to optimize materials and lighting!