Project Number: RL1-P009

Blinding Silence

A Sound-Based Puzzle Game

Interactive Media and Game Development

by

Ryan Bedell
Elliot Borenstein
Drew Hickcox
Lukas Wong-Achorn

Advised by

Professor Jennifer deWinter
Professor Robert W. Lindeman

Abstract

for the development of

Blinding Silence: A Sound Based Puzzle Game

By

Ryan Bedell, Elliot Borenstein, Drew Hickcox, Lukas Wong-Achorn

This is an Interactive Media and Game Development Major Qualifying Project report, focusing on the state and development of a video game based on Terathon's C4 Engine. The game, titled *Blinding Silence*, is a single player game with a unique sound-based visual aesthetic and Wiimote-based control scheme.

This document discusses the state, development, and original design of the game *Blinding Silence*. *Blinding Silence* has a unique control scheme that uses two Wiimotes and an infrared LED headset for in-game navigation. The game also has a unique visual design where every noise makes a burst of light, allowing players to "see" sound. Through these the player solves physical puzzles.

The player controls a blind man with a mysterious staff he uses as a cane. The world has been taken over by darkness, with people endlessly repeating the same task forever. The man discovers he can influence people with sound and begins uncovering the chain of events that led to the catastrophe.

Blinding Silence has a visual style indentified by its sound-based lighting. Models are viewed in silhouette, which removes the importance of textures and increases the importance of models. Humans in *Blinding Silence* have exaggerated physical characteristics to compensate for this.

Acknowledgements

Team Uncertainty would like to thank our advisors Professors deWinter and Lindeman for being critical when we really needed it. We would also like to thank Joshua Luther and Andrew Tremblay for volunteering as talent for motion capture and animatic purposes. We would like to thank all our testers, especially Andrew Tremblay, Morgan Quirk, and Adam Pastorello who lent their likenesses to our presentation.

A special thanks is extended to Graham Pentheny, whose ideas early on in the design phase were influential in the eventual design of *Blinding Silence*.

Finally we thank Team Blackfire (Joe Alea, Garret Doe, Sarah Gilkey, Chris Williams, and Francis Williams), for developing alongside us for all four terms and for being supportive throughout our friendly rivalry.

Table of Contents

List of Figures

List of Tables

1. Introduction

Blinding Silence is a computer game designed by a team of Worcester Polytechnic Institute students as a Major Qualifying Project.

1.1 Descriptions

Use the power of sound to save a world on the brink of madness!

Blinding Silence is a game where the player utilizes the ability to see sound to solve physical puzzles. With unique "sonar vision," puzzle elements come to life in a system of interacting elements. A wave of darkness has fallen over the land, and as the light wanes, so too does free will. Humans are mindlessly repeating the same motions forever. The player controls a blind man saved by chance, influencing the mindless humans through sonic interaction, altering their tasks in order to solve puzzles. The player solves puzzles, breaks the crystals keeping the world in thrall, and helps to bring light back to a land of darkness!

1.2 Audience

The intended audience of *Blinding Silence* is the subset of puzzle game players willing to explore a unique interface. As many puzzle games employ unique interfaces as game or individual puzzle mechanics, this should be a large subset of puzzle game players. A large secondary audience is game players of all sorts who are interested in exploring *Blinding Silence*'s unique visual aesthetic.

1.3 Location

Blinding Silence takes place in a forest that has been nearly clear cut by mindless drones. Enough trees still stand that bursts of light still silhouette dead branches reaching into the sky. Although there are natural elements left in the world, most of them have been transformed by humans to serve a purpose—stumps, square-cut rocks, and piles of logs.

1.4 Controls

The player physically has:

- Infrared LED Glasses
- A cane Wiimote housing with nunchuck attachment

These devices allow for head tracking and visceral interaction for creating sounds in-game, creating a uniquely intuitive game experience. It is possible to play with a mouse and keyboard as an alternative.

1.5 Innovation

The main game mechanic in *Blinding Silence* is its unique visual style, which draws attention to puzzle elements while facilitating the interaction of those elements. The physical control system of the game is also original, offset by the familiar experience of deciphering and influencing working systems.

2 Game Design

This section covers the game design that went into *Blinding Silence*, the design elements that were planned to be incorporated but were cut due to time constraints, and the rationale for those decisions.

2.1 Story

A lengthy background and story was initially developed for inclusion into *Blinding Silence*. Although time constraints led to the majority of the story being cut, a distilled version of the player's motivations and goals was incorporated in the instructions. Appendix A contains the original backstory for *Blinding Silence*, as incorporated into the initial design document.

Section 2.1.1 describes the game's story as designed to be included in the game and rationale for removing it, while Section 2.1.2 explains what is included in the game.

2.1.1 Game story

The story of *Blinding Silence* was designed to be delivered in three ways: an introductory cut-scene, personal accounts told by rescued workers at the end of levels, and various comments made by the cane.

The introductory cut-scene was initially an animatic—a video style combining still images, a few graphical elements, and a verbal narration. The player's cane was sentient, taking the name Zimri and guiding the player at various points throughout the game.

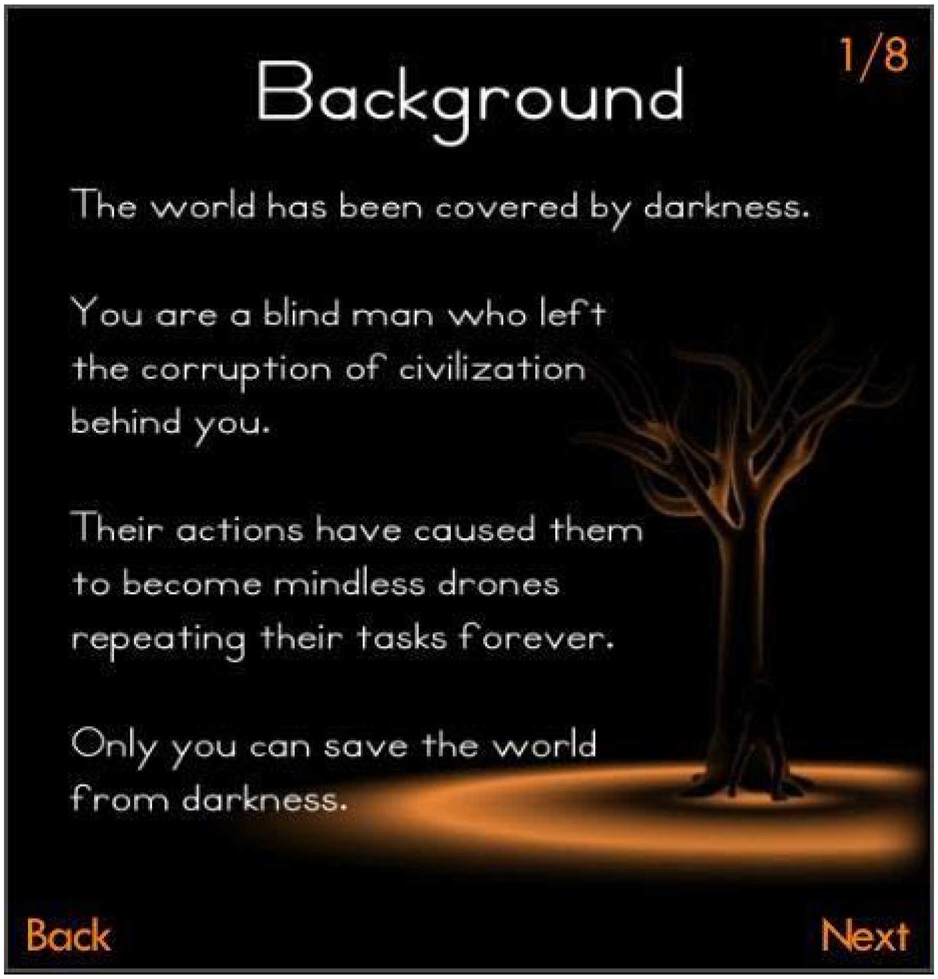
The opening animatic and Zimri's consciousness was cut due to a lack of time. Although player direction and sense of place had been rooted in the player's interaction with Zimri, the team decided to replace the story and introduction elements with a comprehensive instructions screen to maximize time spent on more crucial art elements.

Personal accounts were removed from *Blinding Silence* when the complicated and confusing soul stealing mechanic was removed from the design, and relative levels of success were no longer present in gameplay.

For further explanation of animatics see Section 3.5.1 and Appendix B

2.1.2 In-game story elements

Development of *Blinding Silence* took a lot longer than initially planned. The game did not end up reaching feature freeze until halfway through the third term of development. As a result, the non-critical animatics were trimmed down until they were summarized as a few short sentences that make up the first page of instructions, as shown in Figure 1. Although this is not ideal, the story of *Blinding Silence* is secondary to gameplay. Further expansion of *Blinding Silence* is unlikely to increase the amount of story exposition.


1/8

Background

The world has been covered by darkness.

You are a blind man who left the corruption of civilization behind you.

Their actions have caused them to become mindless drones repeating their tasks forever.

Only you can save the world from darkness.

Figure 1: In-game story

2.2 Gameplay

This section describes the interaction the player has with *Blinding Silence*, as well as the elements in the game that are interacted with, and the rationale for the elements' inclusion.

2.2.1 Overview

Each puzzle in *Blinding Silence* is a miniature machine. The player's goal in each level is to alter the machine such that a certain action is repeated. Once the repeated action is in place, the player can strike the level's crystal and move on to the next level.

As shown in Figure 2, to solve puzzles the player must reverse engineer the system starting from the operator or operators that block off the level's crystal. This requires the player to make a mental map of every level, figuring out how to maneuver actors to achieve a winning state. This process is well supported by the visual style of *Blinding Silence*, since all actors that would need to be moved are making the only lights the player can see.

To facilitate solutions, *Blinding Silence* has controls in place to help the player orient in levels, and its very levels are designed in such a way as to focus the player into paying attention to relevant game objects.

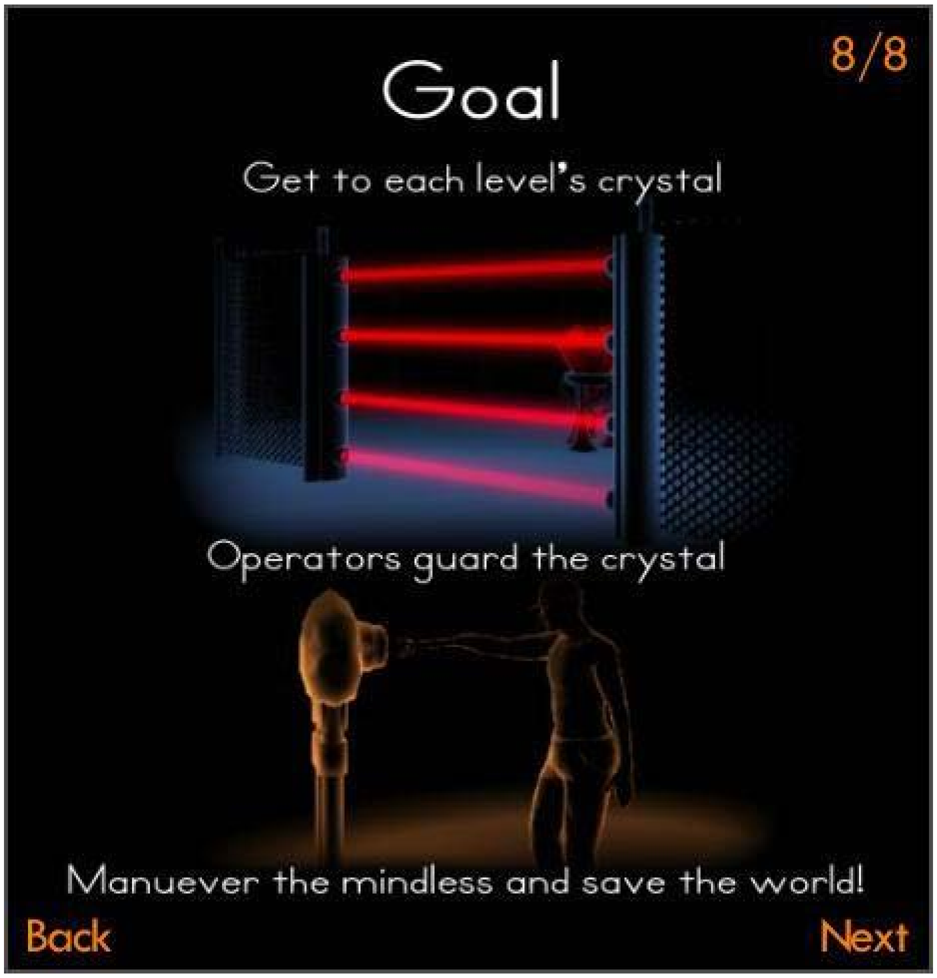

8/8
Goal
Get to each level's crystal
Operators guard the crystal
Manuever the mindless and save the world!
Back
Next

Figure 2: In-game objective summary

2.2.2 Interface

The controls of *Blinding Silence* were some of the earliest established aspects of the game. Having direct control over the in-game cane as well as head tracking was included in the earliest design documents as a unique feature of the game. The in-game page explaining the controls can be seen in Figure 3.

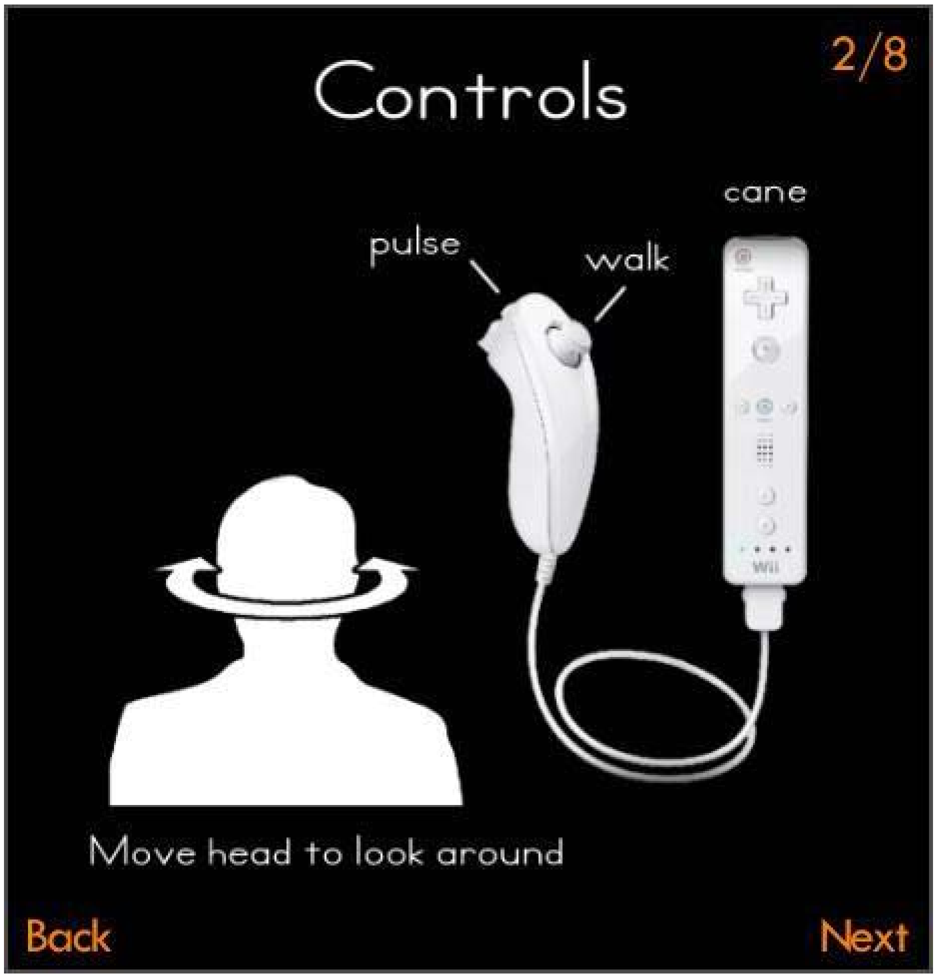


Figure 3: In-game control instructions

2.2.2.1 Wiimote cane

The Wiimote cane consists of two parts. The Wiimote is used to control the in-game cane, which the player uses to make sounds that affect level elements and allow the player to see. The nunchuck attachment allows the player to move, and also has the ability to send out loud pulses of light that illuminate the world for the player to get his or her bearings.

The nunchuck attachment's analog stick provides "tank control" over movement—the player can move forward, backward, and turn. While this is somewhat limiting, it is augmented with control over the first person camera.

Pulsing functionality was added after the first round of developer testing of levels, as establishing the character's place in the world was immediately apparent as needed for understanding where a level's elements had been placed.

2.2.2.2 Head tracking

A second Wiimote pointed at the player's head combined with infrared LED-adorned glasses allows for simple head tracking to be accomplished. The player controls the first person camera by moving his or her head vertically and horizontally. Smaller movements can be accomplished by tilting up, down, or to the sides.

This head tracking simplifies the process of orientating in a level. The player can look left and right while walking, which helps the player identify objects and their placement in the world. This added support is necessary due to the pulsing nature of lighting in *Blinding Silence*—every bit of help in identifying where objects are speeds up the crucial step of figuring out how the levels are set up.

2.2.3 Sounds

Sounds in *Blinding Silence* are the light by which the player sees. This mixture of senses was inspired in part by synesthesia, and in part to combine sight and sound in order to make a unique gameplay experience.

As sounds create bursts of light, player attention is drawn to any game element that makes sounds. These are limited to actors and switches, which are the elements that players actually need to interact with. In this way, play attention is drawn directly to puzzle elements in every level.

As shown in Table 1: Sounds and associated colors, sounds are based on the underlying material properties of the object being struck.

Table 1: Sounds and associated colors

Material	Color
Wood	🟨
Stone	🟪
Metal	⬜
Dirt	🟫
Crystal	🟥

Every object in the game either corresponds to one of these sounds or is assumed to be dirt. Categorizing sounds in such a way simplifies interactions for the player, allowing for a range of sound creation to be associated with a limited set of interactions. While the actual sound of chopping wood is very different from that of hammering wood, the light created is the same.

The first two sounds are the sounds that actors—the mindless drones that serve as puzzle elements—respond to. All actors have a similar but separate response to wood and stone sounds, generally moving to a corresponding resource. This was decided early on, as providing two inputs for every actor allows puzzles to become complicated in design while remaining simple in theory.

Metal and dirt sounds have similar functionality, which is very little. They serve to provide illumination and concept reinforcement. Actor footsteps create dirt noises, which helps players track where a moving actor is without affecting the system.

Crystals have their own special sound, which separates them from the rest of the sounds in the game.

Player interaction is performed entirely through using the cane to create sounds. The cane creates a sound based on the material of the object struck.

2.3 Level components

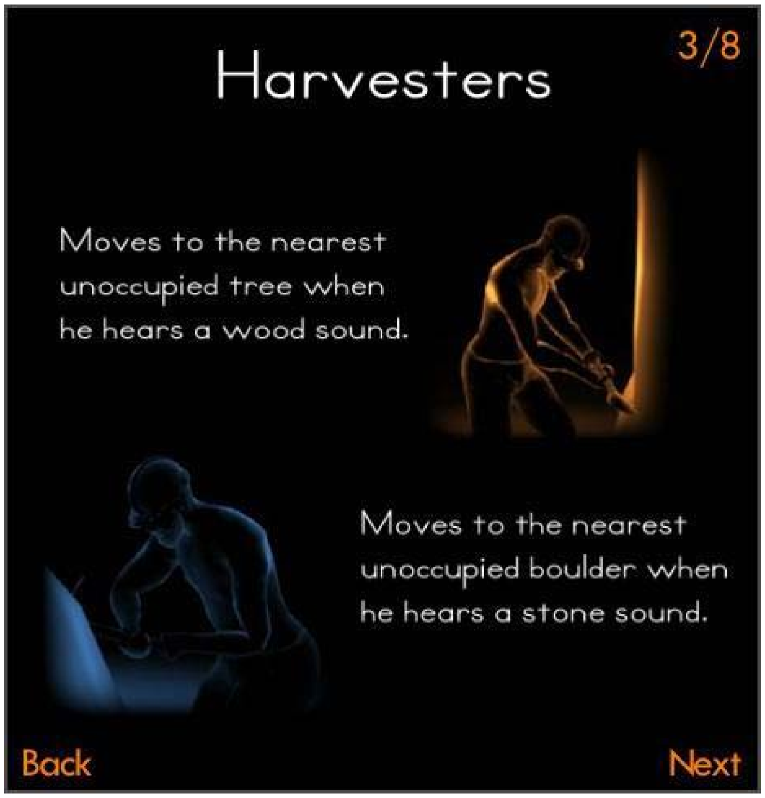
This section describes the various components that can be found in the levels of *Blinding Silence*. Every component is placed using custom tools created to speed up the art asset pipeline.

2.3.1 Actors

Actors are the actual puzzle elements of *Blinding Silence*. Actors endlessly repeat an action until they hear a sound and switch where they are acting, only to mindlessly repeat an action once again. The main differences between actors are what places they perform their actions, what actions they perform when they get there, and how they respond to different sounds.

2.3.1.1 Harvesters

Harvesters are the most straightforward actor in *Blinding Silence*, and as such they are the first actor that the player interacts with. Harvesters interact with trees and boulders, which they move to when they hear wood and stone sounds, respectively. As shown in Figure 4, harvesters replace their left hand with the appropriate tool as they work. This theme is repeated in all other actors, and was intended to show that over time the mindless drones have become one with their tools.



Moves to the nearest unoccupied tree when he hears a wood sound.

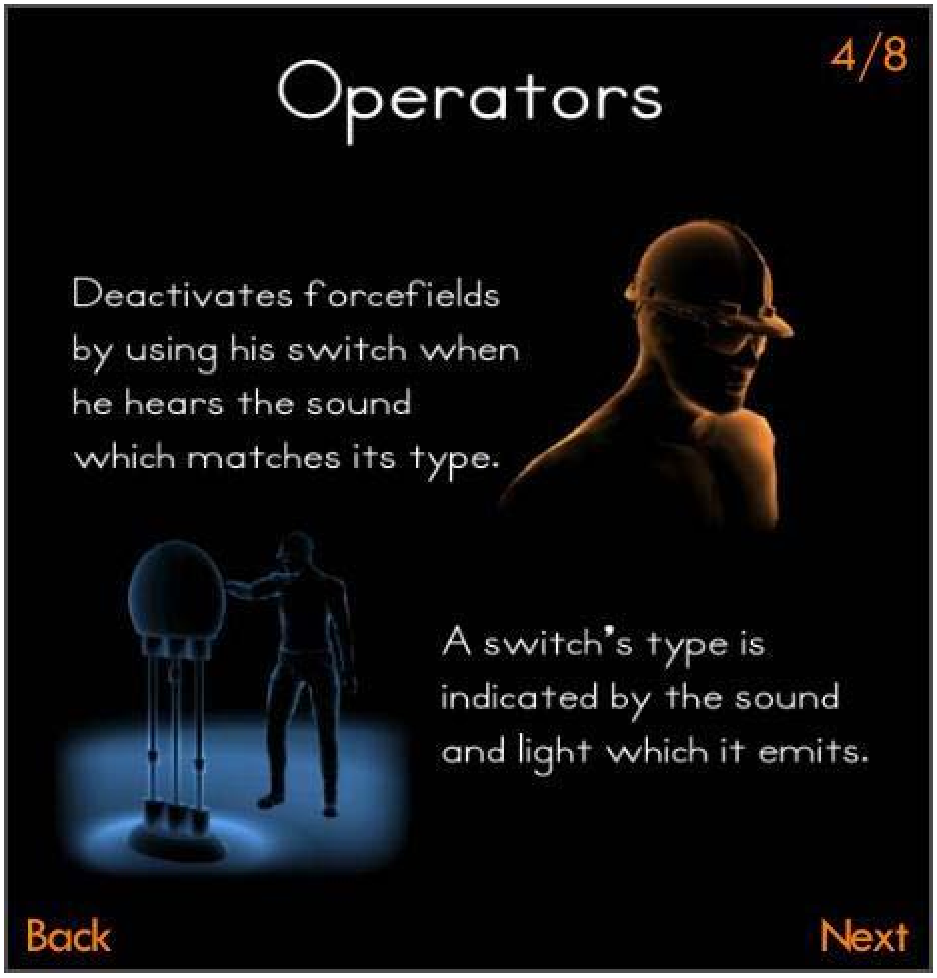
Moves to the nearest unoccupied boulder when he hears a stone sound.

Figure 4: In-game harvester screen

2.3.1.2 Operators

Operators are the focal actors of every level. Operators stand by a switch, and are unique in that they have an idle period during which they do nothing. As a balancing factor in this, the operator's switch emits the sound that the operator needs to hear, as shown in Figure 5.

Once the operators hear the appropriate sound, they interact with the lever in front of them. This opens a gate in the level, although the gate will close again. Only by having another actor create a recurring sound near the operator can the player pass through the gate.



Operators

Deactivates forcefields by using his switch when he hears the sound which matches its type.

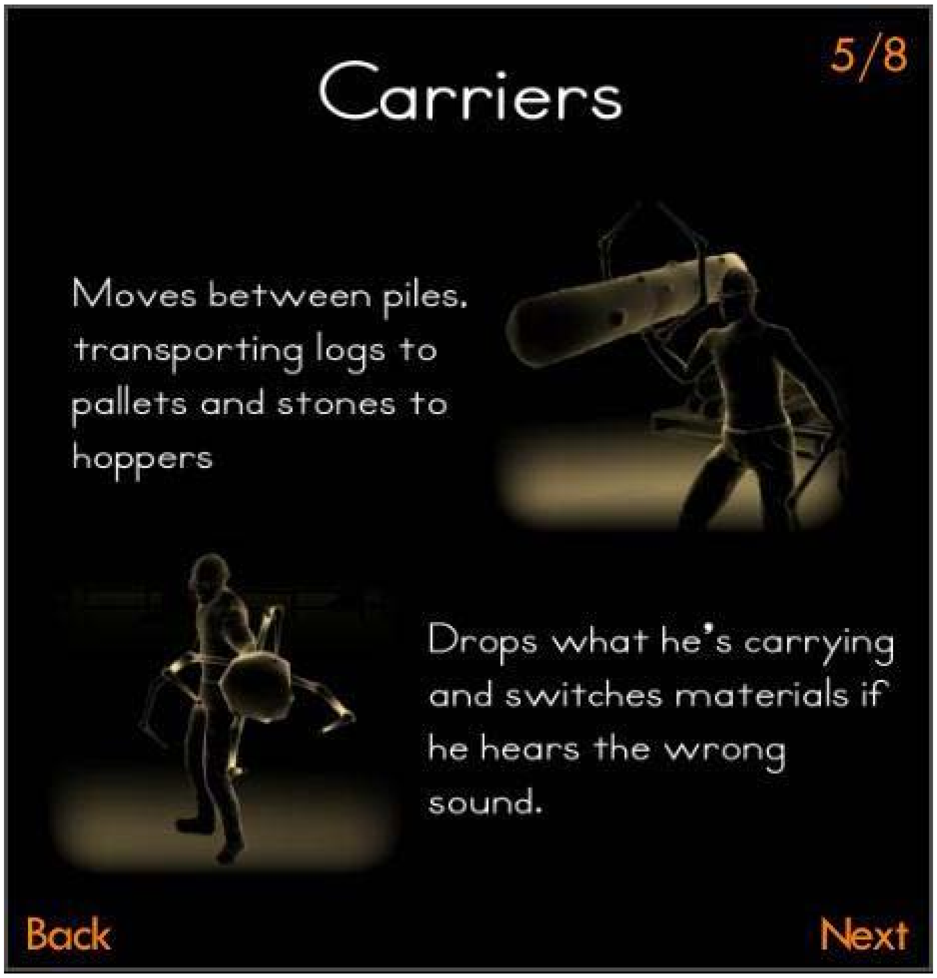
A switch's type is indicated by the sound and light which it emits.

Figure 5: In-game operator screen

2.3.1.3 Carriers

Carriers move between two different resources, which makes their responses to wood and stone sounds slightly more complicated. The carrier moves to a log or stone pile, and moves to a wood or stone hopper. If the carrier hears the opposite sound from what he is working on, he will drop whatever he is carrying and switch targets. However if he is carrying wood or stone and hears the same sound, the carrier will switch between corresponding hoppers.

As shown in Figure 6, the carriers have both hands replace by oversized claws. While in constant light this makes a carrier look cartoonish while carrying a log or stone, in the pulsing darkness of *Blinding Silence* it has the effect of making carriers seem almost spidery.



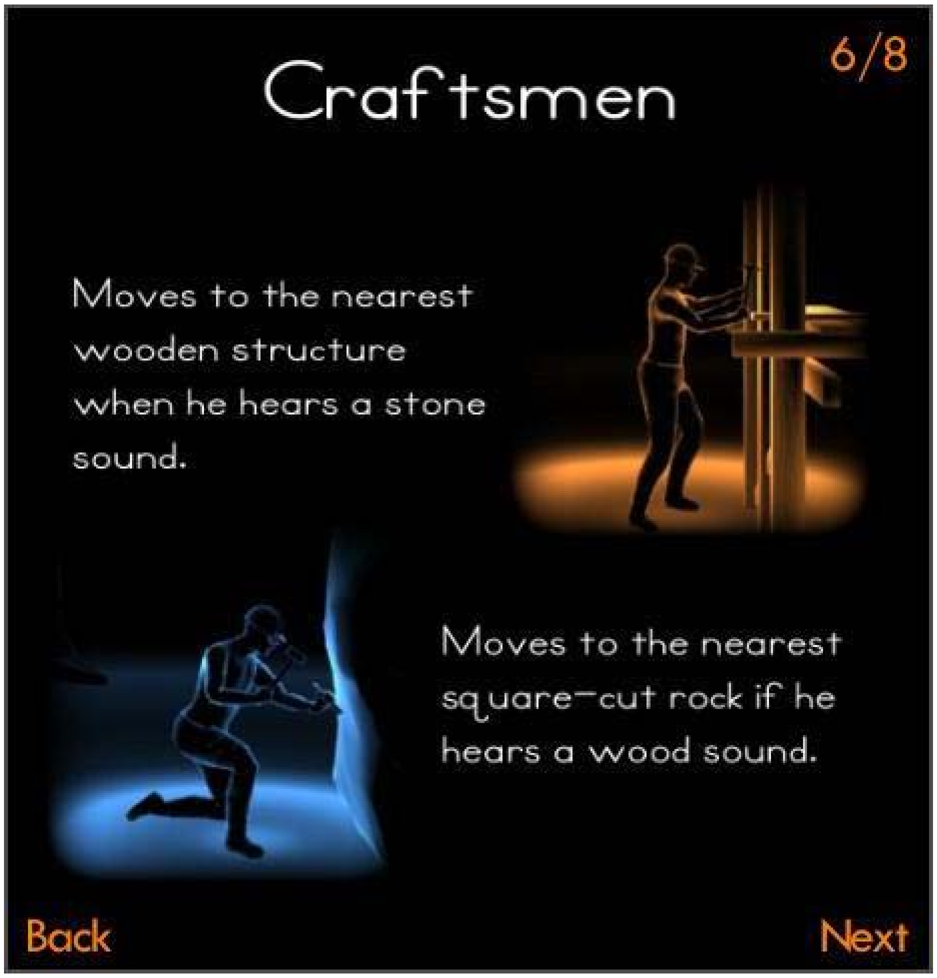
Moves between piles, transporting logs to pallets and stones to hoppers

Drops what he's carrying and switches materials if he hears the wrong sound.

Figure 6: In-game carrier screen

2.3.1.4 Craftsmen

Craftsmen are similar to harvesters in that they move to one type of resource and stay there. The wood resource they work on is a scaffold, and the stone resource is a large square-cut stone. However, the craftsman goes to the opposite resource type to the sound he hears. As explained in Figure 7, he goes to work on the stone material when he hears wood, and wood material when he hears stone.



Craftsmen

Moves to the nearest wooden structure when he hears a stone sound.

Moves to the nearest square-cut rock if he hears a wood sound.

Figure 7: In-game craftsman screen

2.3.2 Objects

There are several other elements present in levels besides actors. These serve to direct actors and players as barriers, focal points of work, and as objects the player must strike in order to create sounds.

2.3.2.1 Actor resources

There are five wood and five stone resources, shown in Figure 8, that correspond with various actors. These each make the corresponding sound when struck, allowing the player to move actors to similar points simply by striking the resource they are working on. This works unless there is no other viable point for the actor to go to, in which case the actor will ignore the sound.

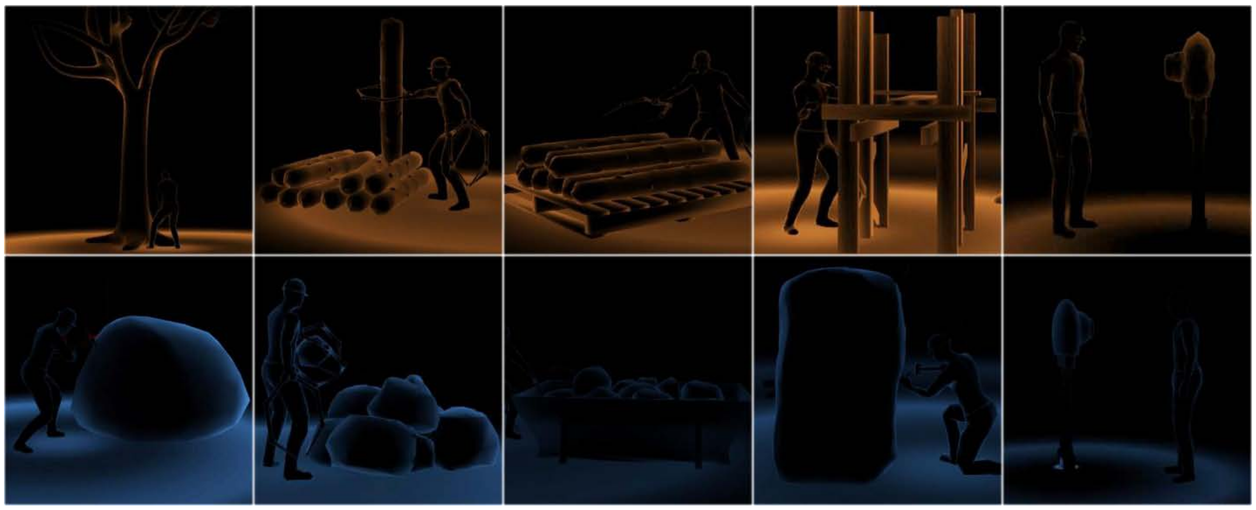


Figure 8: Actor resources

2.3.2.2 Barriers

Various objects block the path of actors and the player, and examples of these can be seen in Figure 9. Metal and stone fences block off the level-ending crystal, mark the outside of the level, and divert actors over long spaces. Chopped up tree trunks also litter the area. These not only cause the levels to be more interesting to traverse, but allow the player more opportunities to create sounds they want. Actors will not interact with stumps or stone walls, but they will respond to sounds created by hitting them.

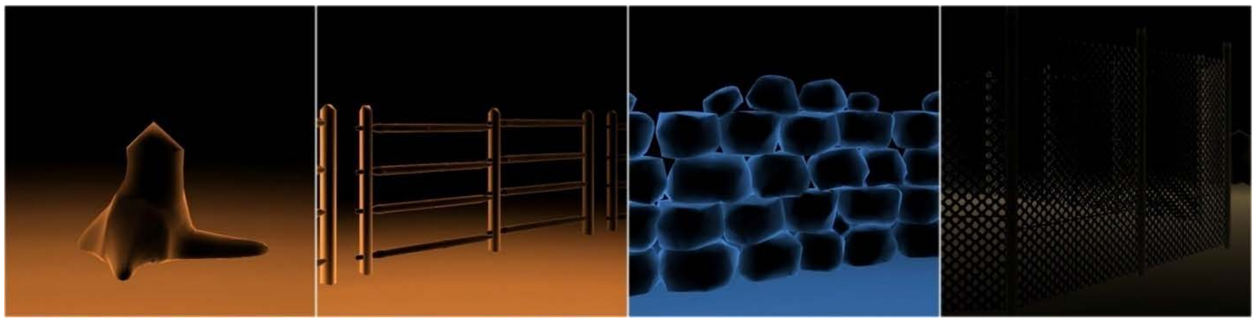


Figure 9: Non-interacting objects

2.3.2.3 Crystals

Reaching the crystal is the goal of every level. When all the operators are being correctly triggered, the player can strike the level's crystal and move on to the next level. This provides visceral satisfaction as reward for completing a level. To enforce the reward, some levels have a large crystal at the end, shown in Figure 10, which dwarf both the smaller crystals and the player.

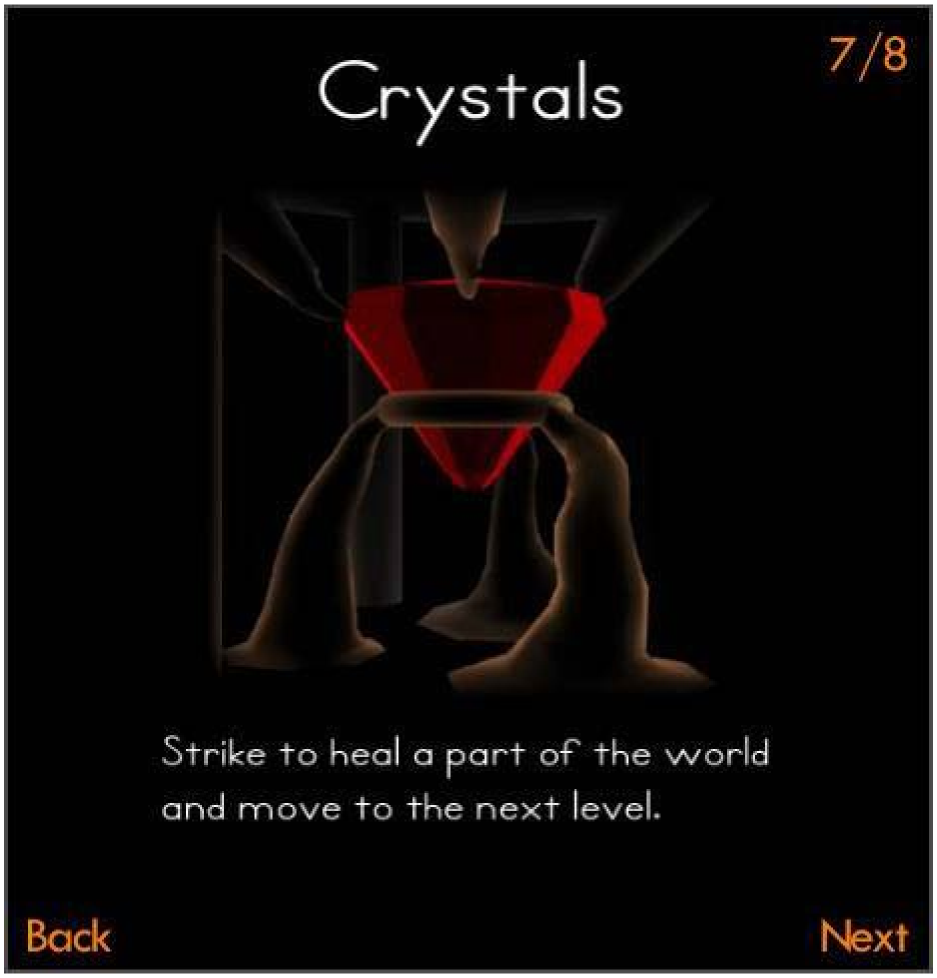

7/8
Crystals
Strike to heal a part of the world
and move to the next level.
Back
Next

Figure 10: In-game crystal screen

2.4 Levels

Levels in *Blinding Silence* are set up to slowly expose players to the different kinds of actors found in the game. Over the game's eight levels a maximum of two different types of moving actors can be found at a time. This was planned due to time constraints; around halfway through development it became apparent that there would not be much time to spend developing levels. Rather than further confuse the player with complicated puzzles on top of a unique lighting system and original control system, levels are designed to ease players into the game.

If any future development would be done on *Blinding Silence,* it would focus mainly on expanding the levels.

3. Artistic Design

Blinding Silence puts the player inside the mind and perspective of a blind man who can “see” sound. To convey this sensation of synesthesia, sounds which occur in the game world are coupled with a visual representation of the “sound wave” as a form of illumination, as shown in Figure 11. As such, the artistic style favors the significance of shape and silhouette over that of surface and texture. Color in Blinding Silence is uniform and simple; objects do not have their own color, but instead are tinted based on the sounds which the player is hearing. This strengthens the bond between the visual and auditory elements of the game, immersing the player in a unified aesthetic.

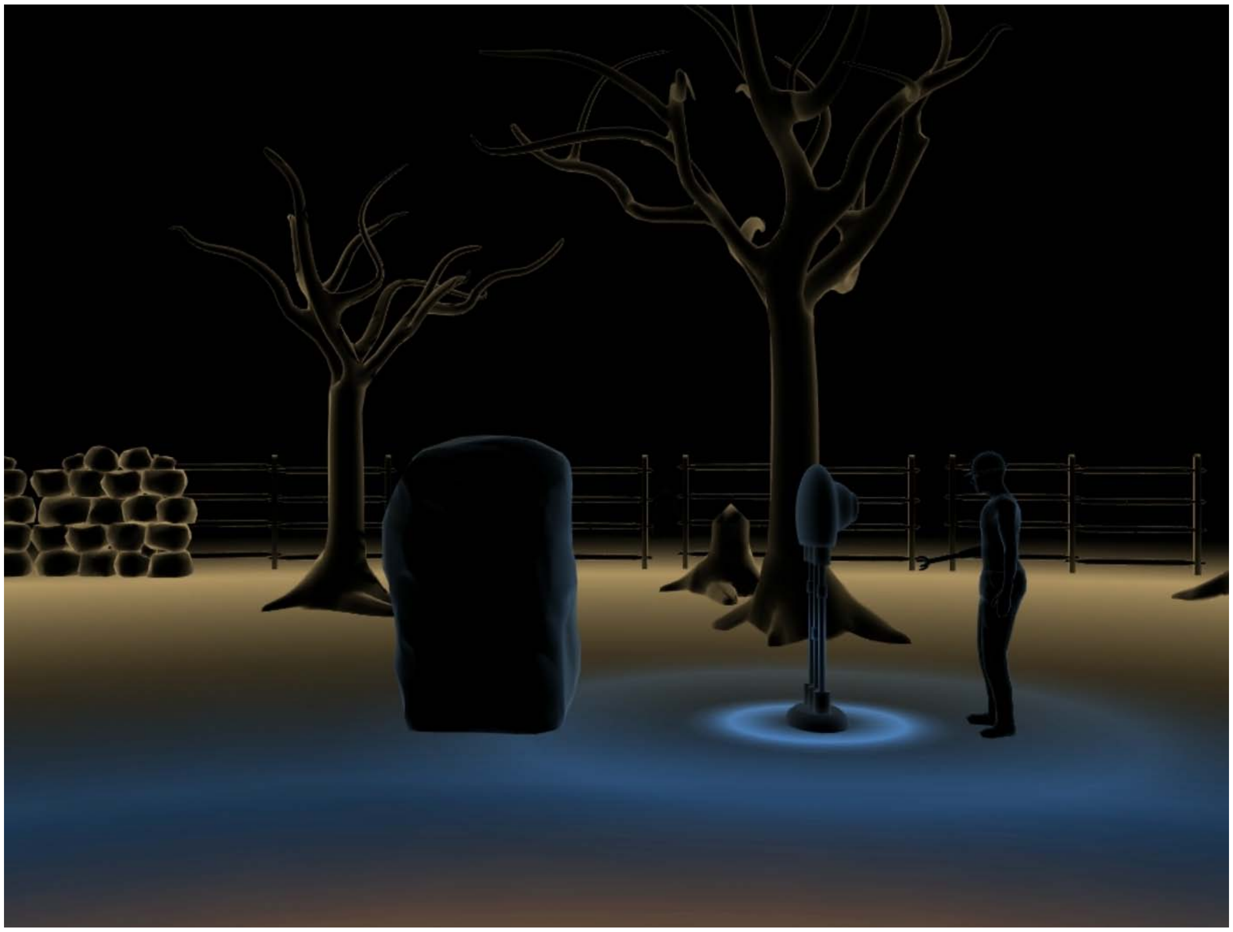


Figure 11: The World as Seen By the Player

A full asset list is available in Appendix C. Reference and concept art can be seen in Appendix D.

3.1 Visual Style

The player is blind but can see the world through the use of Sonar Vision. All sound in the world is visible, and the louder the sound, the brighter the visibility of the object to the player. Ambient sound sources and background noise create dim lighting of the surrounding area while individual and distinct sounds create visible waves that illuminate anything they hit. The player sees objects hit by these waves mainly by illumination of object edges. Objects tend to have a silhouetted look with little detail visible inside of the visible edges, as can be seen in Figure 12.

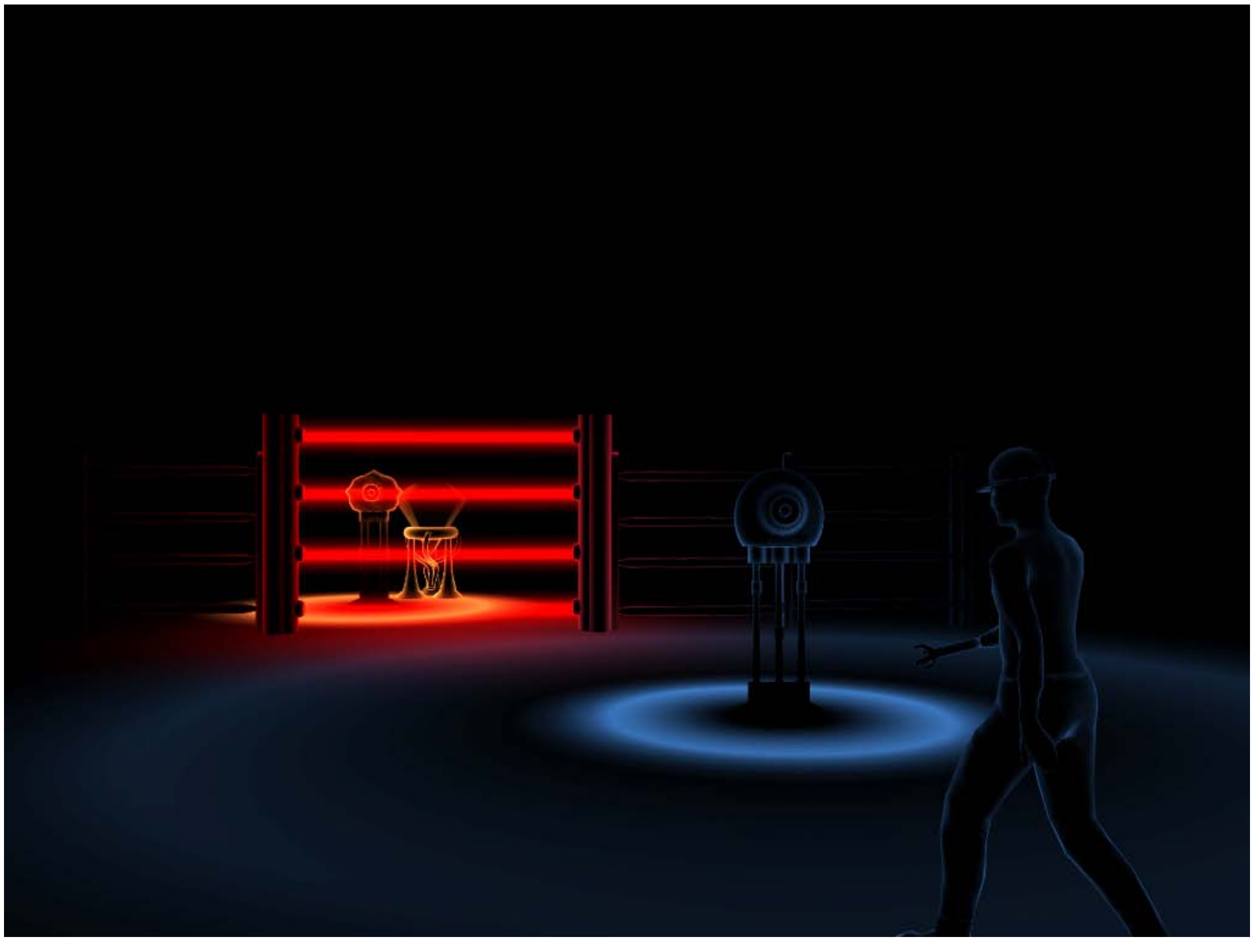


Figure 12: Simple Example of Visual Style

Each type of material is associated with a particular color; objects made of wood produce one color, impacts on dirt another, ambient noise a third, and the like. When placed against a dark background, the distinct and bright colors are plainly visible, allowing the player to easily localize the source of the sound. Also, by learning to recognize the different colors and their meanings, the player can quickly assess the world around him.

As an example, the player moves within audible range of a harvester who is repetitively striking his axe against a tree. The area around the harvester has a level of orange ambient illumination as his constant hammering has created a resonance in the area. However, this is a very dim illumination. Each time the harvester strikes the tree, the area immediately around him brightens considerably and slowly fades to its ambient level until the next strike. At the same time, when the tree is struck, it creates a wave of sound that moves away from it, illuminating in orange everything that it touches but with decreasing intensity the farther it moves from the tree. Shortly after becoming illuminated, the objects start to fade back to darkness. In this way, the player is able to localize the source of the sound (the harvester will always have some level of illumination, brighter than the edges of the waves he creates), while also being able to see his surroundings as the pulses illuminate the world.

In addition to loud sounds that illuminate large swaths of the world to the player, the world will also contain objects that produce sound constantly, but on a much smaller scale. For instance, active power crystals vibrate, and energized force fields hum. These smaller objects will not produce enough sound to illuminate the area around them, but will instead have constant ambient illumination allowing the player to have static reference points within the world, while he influences and moves the larger sound producing entities. For added visual interest and realism, the footsteps of actors in the world also create visible sound. While the illumination generated is not strong enough to light up a large area, it can help the player to determine where actors are, and what paths they are taking at a given time.

3.1.1 Technical Implementation of Sonar Vision

To represent sound visibly, *Blinding Silence* uses and augmentation to the standard C4 lighting and rendering engine. In addition, every object in the world uses a specialized shader, similar to a Fresnel shader.

3.1.1.1 Augmented Fresnel Shader

The Fresnel shader illuminates the edges of the object, while providing less illumination to details on the portions of the object facing the player. It does this primarily by using the inverted tangent view direction between each world point and the player camera, causing those points that face toward the camera to be darkest and those perpendicular to the camera's view to be brightest. This helps to provide the silhouetted art style for the game. The standard implementation of this type of shader has been enhanced to include normal mapping, allowing for added model complexity and detail in the silhouettes. A detailed description of the standard Fresnel shader, and its implementation, can be found on the C4 Engine wiki[1].

To help add increased detail to the world, we augmented the standard Fresnel implementation with a normal map. Figure 13, on the next page, shows how this was achieved. On the left side is the standard implementation, sending the z value of the tangent view direction into an invert node. On the right, a normal map is sampled and the dot product is calculated between it and the tangent view direction. The normal map is multiplied by a constant prior to the dot product as a method of controlling the intensity of the effect.

[1] http://www.terathon.com/wiki/index.php/Building_a_Fresnel_shader

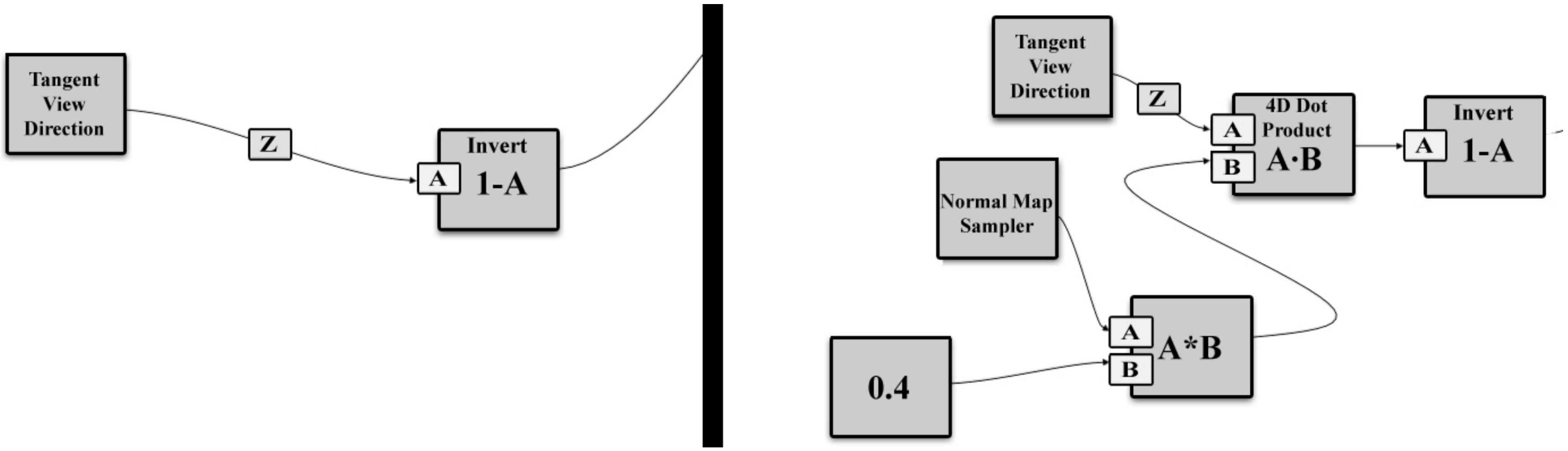


Figure 13: Fresnel augmentation; Standard on the left, changes on the right

3.1.1.2 Pulsing Shader

Some objects in the world have a form of constant illumination. In particular, the crystals that are the goal of every level needed to be visible for the player to know where his goal was. However, we wanted these ambient sources to still have a dynamic look to them, so we created another offshoot of the Fresnel shader. This pulsing portion of the shader, seen in Figure 14, on the next page, smoothly ramps up to full brightness, and then smoothly ramps back down to a set intensity. C4's shader editor provides a node called Shader Time, which generates a ramp from 0 to 1 linearly every 120 seconds. The portion on the left of the figure takes this value, and uses it to calculate the same 0 to 1 ramp over a different period of time (in this case every three seconds). This doesn't create a smooth pulse in intensity, as the value immediately jumps back to 0 once it reaches 1. The right side of the figure shows how the shader adjusts to smoothly ramp up to 1 and back down to 0. The Set If Greater Equal node will output a 1 if the input value is greater than 0.5, and a 0 otherwise. If the value is between 0 and 0.5, the lower portion of the figure will output the value directly, ramping from 0 to 0.5. If the value is greater than 0.5, then the upper portion will create an output, subtracting the input value from 1 to provide an output that ramps down from 0.5 to 0. This produces a value that smoothly ramps between 0 and 0.5, and then back down. The last step is to multiply this output by a chosen value to produce the desired range. For example, to ramp from 0 to 1 and back, the output would be multiplied by 2.

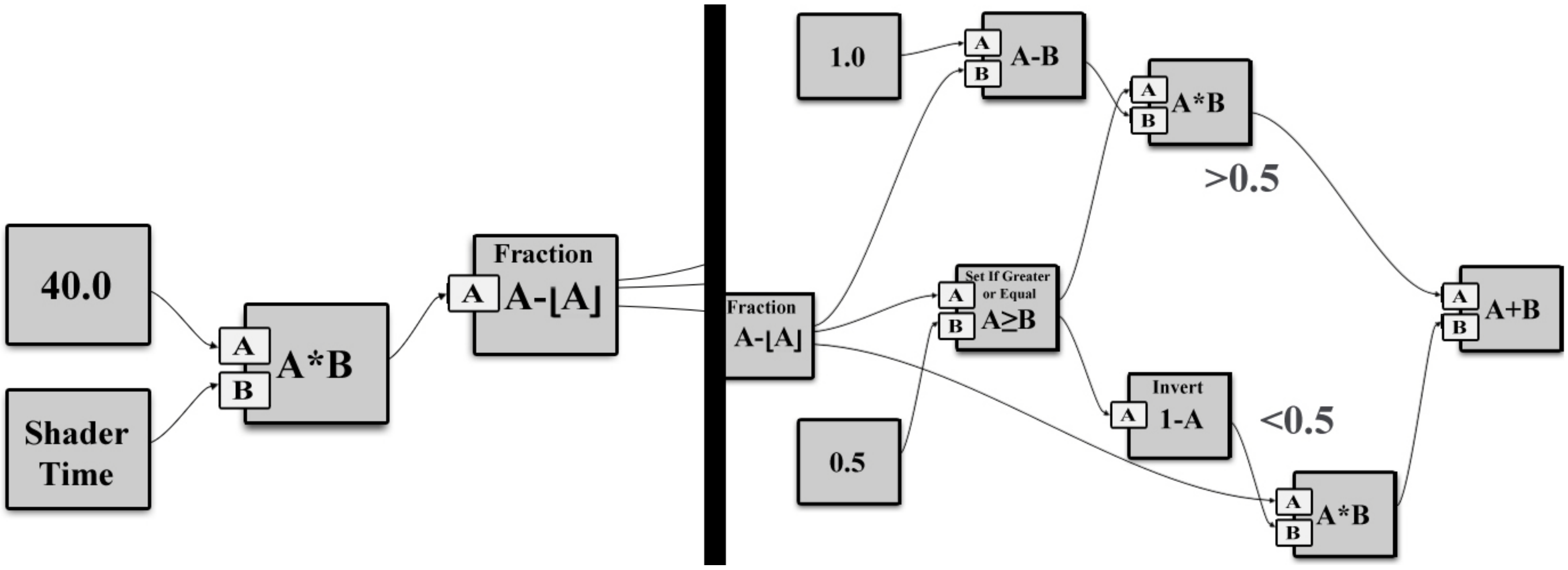


Figure 14: Pulsing Shader

3.1.1.3 Visible Sound Emitters

The most difficult effect to create was the sound waves illuminating the world. In the design phase, it was suggested that the ideal method of implementation was to have sound-emitting entities spawn spherical lights that grow outward. As they grow, the intensity of the light decreases, and they are removed from the world once they reach an intensity of zero. Rather than just illuminating within the entire sphere, only the outer shell of the sphere should emit light, imitating the nature of a sound wave; only the objects within this shell would be given illumination while objects that had already passed through it would return to darkness. This could either be implemented using a spherical volumetric light, or by using a texture map on the light. We determined that the above method was not possible using the stock capabilities of C4's lighting system, and the developer of the engine cautioned against implementation of the volumetric light shells described above, citing excessive computational expense.

We continued to explore other options, and an idea was proposed to create the effect using a custom shader. Under this design, the engine would pass in the location of the sound emitter, along with its inner and outer radii. The shader would then interpolate the world-space position of each vertex, and perform a distance calculation between each fragment and the emission source. If the fragment was within the source's radii, it would calculate some amount of self illumination to multiply by the Fresnel portion of the shader. This design would also allow the player to see through un-illuminated objects by killing any fragment not within illumination range, which could help add to the visual complexity of the game. Unfortunately, though this method is possible in general, it came with two problems. The first was the simple fact that C4 does not allow for the creation of custom vertex shaders and does not publish the fragment location to the shader editor. The second problem was that this solution only works for a single emission source. It could be extended to use more than one source, but this would always be a fixed value and could cause the shader to quickly become too complex for real-time rendering. After some discussion, we eventually decided to stop pursuing this method.

Through early prototyping, we discovered that using two point lights per emission, one large and one small, created an effect similar to the one described. As such, the initial builds of *Blinding Silence* spawned standard point lights that increased in radius as they decreased in intensity until being removed from the world, as seen in Figure 15. This solution provided an acceptable compromise between the artistic vision and the real world technical limitations, but still did not fully generate the desired effect.

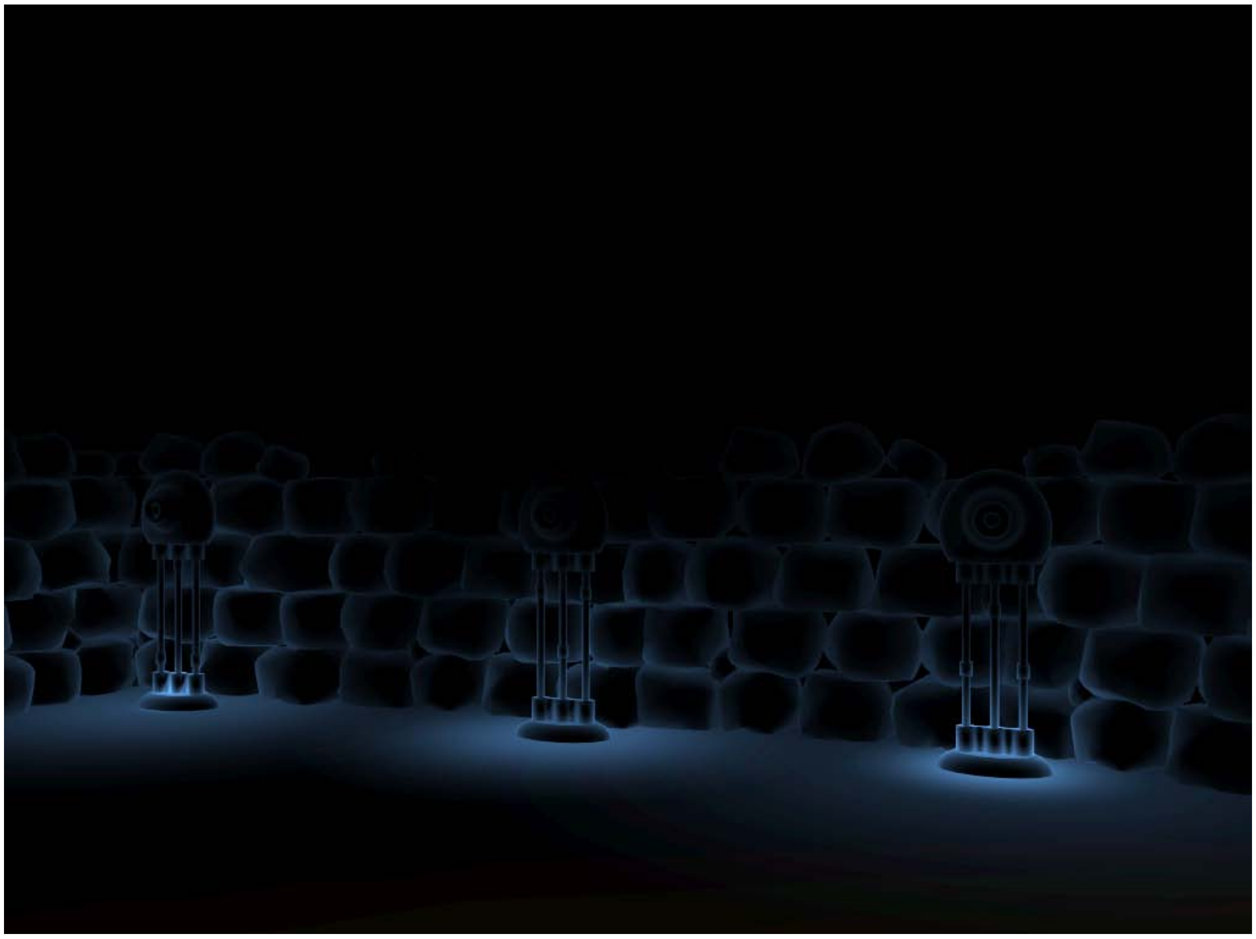


Figure 15: Sonar Vision with Standard Point Lights

Late in production, between C and D terms, we decided to revisit the effect. After researching the intricacies of C4’s lighting system, it was discovered that the attenuation for lights in the engine is controlled by a simple 3D texture that is pre-computed at runtime and then used by every light. Eventually, we were able to modify the attenuation texture to generate an effect that resembled the volumetric light shells described in the original design. The original texture was normalized to falloff from an intensity of 1 at the center to an intensity of 0 at the edge of a circular radius. We used this normalized value as input to a pair of linear functions that ramped the texture up to 1 at an arbitrary distance from the center, and then ramped it back down to 0 to create the shell. This worked quite well and generated the effect we wanted, seen in Figure 16.

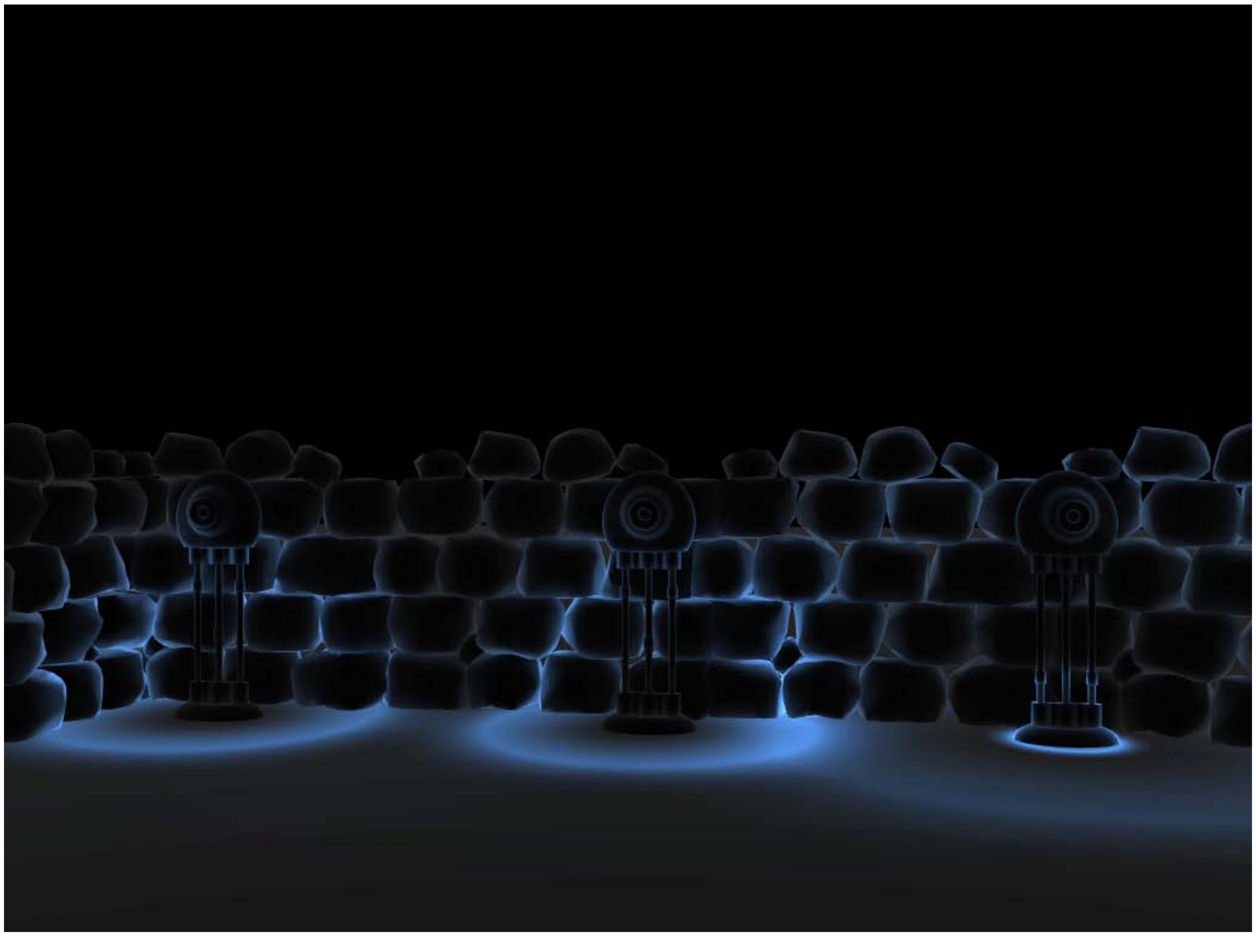


Figure 16: Sonar Vision with Custom Attenuation Texture

However, this attenuation texture is global to all point lights, and we still wanted the use of a standard point light for certain effects. We decided to create a second texture that would use the original falloff calculation, and then modified the internal shader code of C4's cube lights, allowing us to use both the original and custom attenuation textures at the same time. In the end, we were able to create the desired volumetric effect with zero additional rendering cost compared to using standard point lights.

3.1.1.4 Visible Sound Emitter Limitations

There are some drawbacks to using the method we finally settled on. One such drawback is that using this method removes the ability to use shadow-mapped cube lights. This didn't affect our game since we don't use any shadows, but others using this method would need to further augment the engine to accept a third type of point light.

The other main drawback is that the attenuation textures generated are global to all lights of a given type. This means that the falloff of each light's shell is not directly controllable. The biggest result of this fact is that the larger the light's radius gets, the larger the size of the light shell. Figure 17 shows two lights, one with a small radius and one with a large radius. Ideally, the thickness of the lit area should remain constant, but as the light grows the attenuation texture is stretched and begins to create a thicker and less distinct ring. Given the nature of our lighting effect, including how fast the lights fade out and the limited radii to which they grow, the problem is not sufficiently noticeable though. It should be possible to improve the visibility, either by having several different pre-computed textures or by somehow dynamically changing the attenuation texture for a given light as it grows.

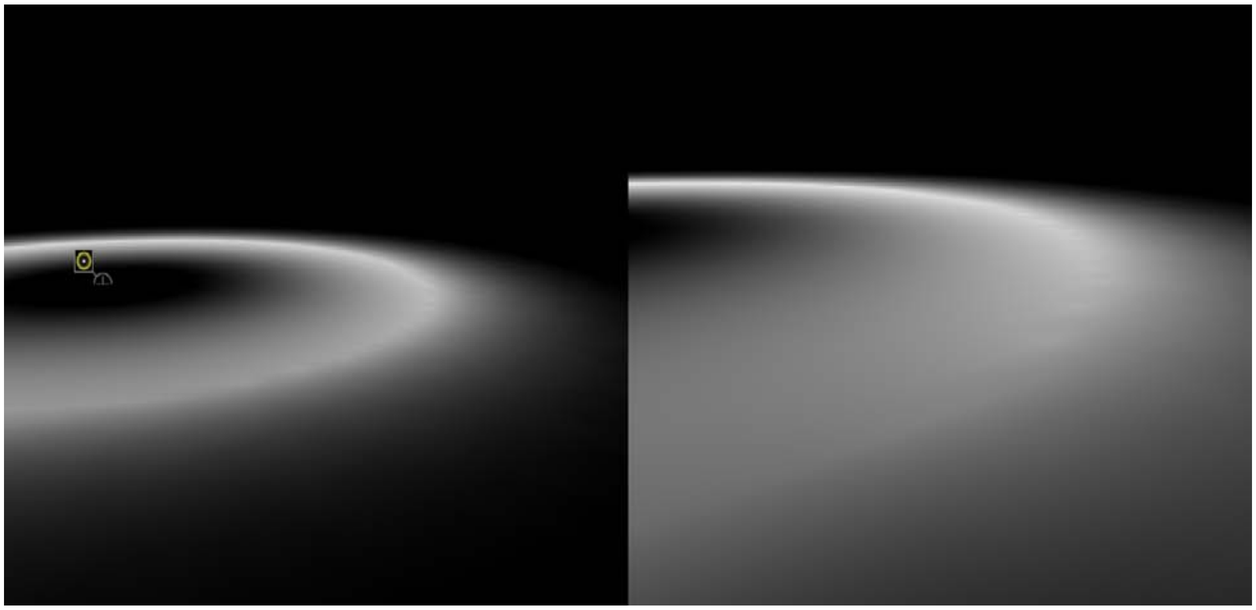


Figure 17: Attenuation Texture Stretching

3.2 Level Design

Blinding Silence takes place within a forest on the outskirts of a city. The areas surrounding the city are being industrialized, so the levels are populated with workers and rudimentary structures. Each level also has a crystal in it which the player must reach to complete the level. This crystal is enclosed by fences, accessible only via switch-controlled force fields.

3.2.1 Layout

Because the sound interactions are dependent upon distances, the placement of objects and actors in the levels is very important. A view of a completed level layout can be seen in Figure 18. Level design begins with the inclusion of puzzle-critical elements. Supplementary objects are then placed to populate the environment, and much care must be taken to preserve the solution to each puzzle when doing so. Each level has a boundary consisting of a series of fences which keep the player within the confines of the puzzle.

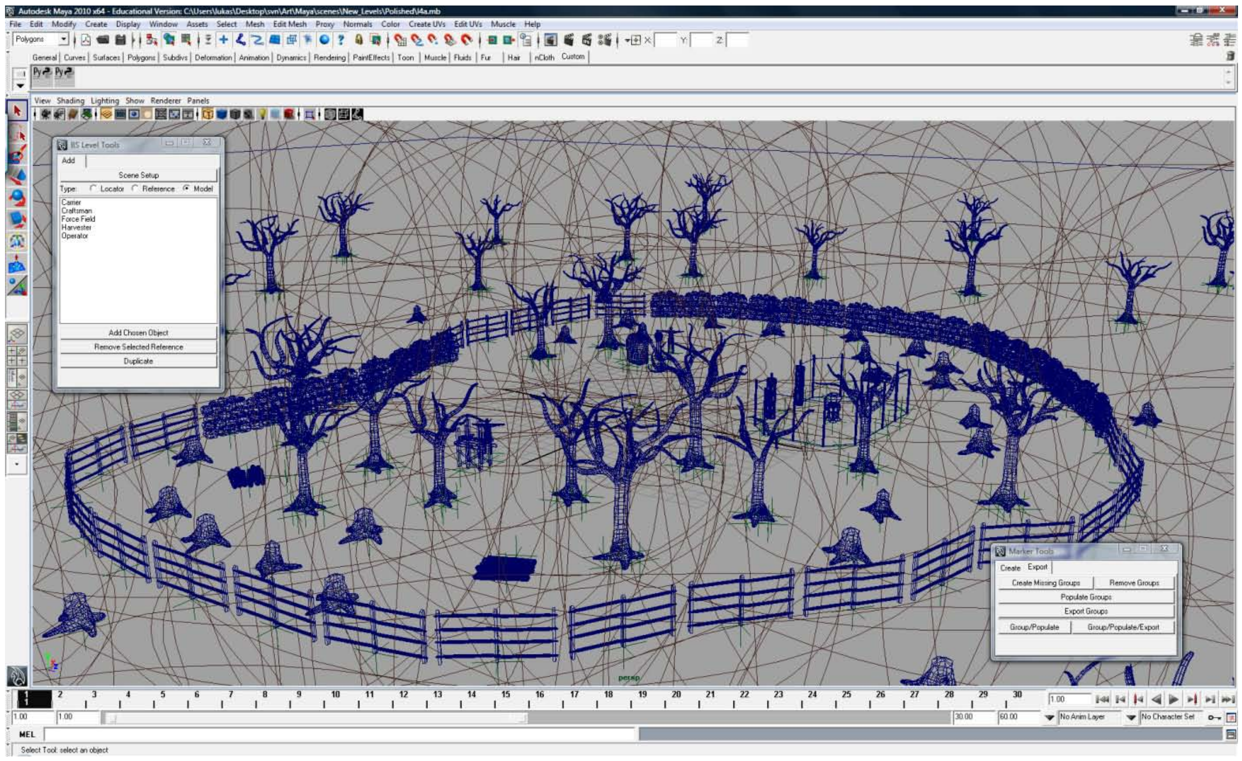


Figure 18: A completed level

Supplementary actors can be placed outside this boundary without interfering with the puzzle. So long as they are placed out of range of any potential sounds created by the player or puzzle-critical actors, they will not effect the gameplay. However, when placing external interactibles, care must be taken to ensure that actors inside the puzzle area will not attempt to walk through the fences to use them. These supplementary actors are used to provide ambient noise and additional lights.

3.2.2 Resources

Wood and stone exist in several forms within the game world. There are two types of trees throughout the forest, as well as their harvested remains. Logs are stacked in piles near trees and stumps, and also on top of wooden pallets near structures. Stone in its raw form are large boulders, and harvested stone can be found in piles of smaller rocks or in large metal hoppers. Resource deposits are infinite; the number of visible logs or stones does not change when one is added to or removed from a pile. Figure 19 shows a selection of resources in the game.

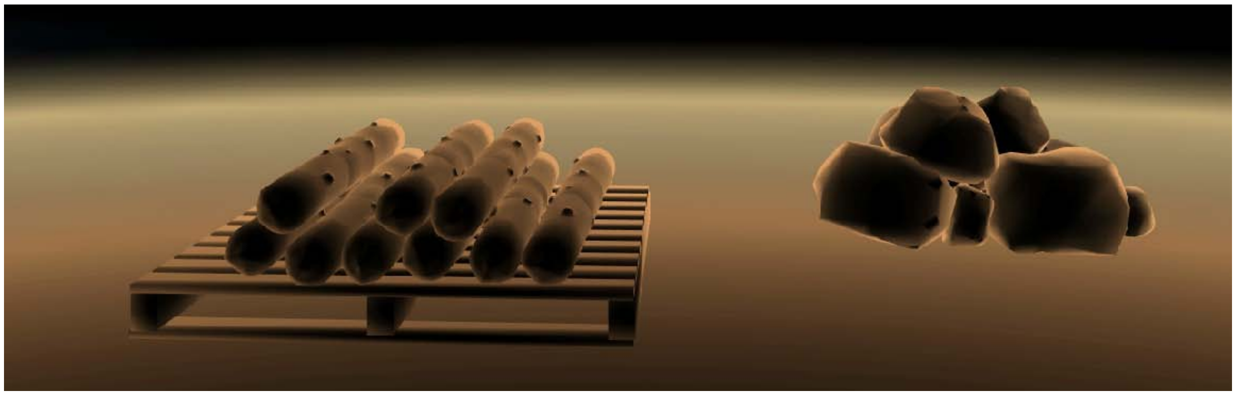


Figure 19: An example of resources

3.2.3 Structures

The two manmade wooden structures in the game world are the scaffold and the wooden fence. The scaffold is a simple construction of wooden planks, part of some larger structure which has not yet taken shape. The wooden fence, seen on the right of Figure 20, consists of posts with slats for horizontal wooden beams. Some large stones have been cut into more usable block shapes, and other smaller ones have been stacked to form rudimentary barriers, as seen on the left of Figure 18. There is also a metal chain-link fence which encloses the crystal in each level.

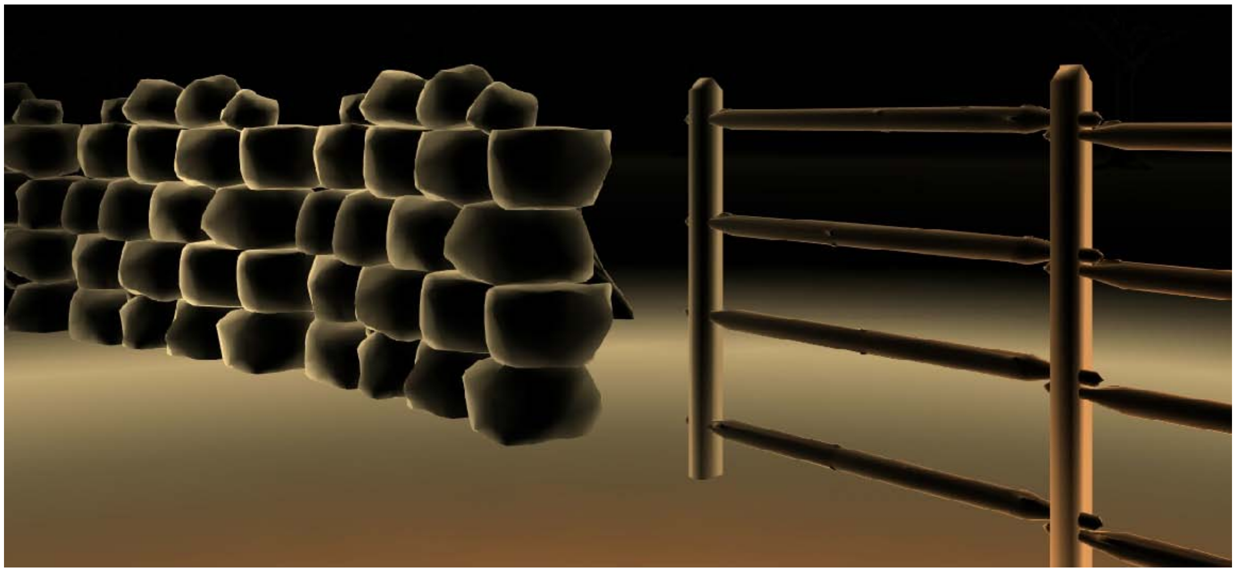


Figure 20: Stone and wood fences

3.2.4 Crystals

There are two types of the crystals which are found at the end of each level, one small and one large. They are embedded into bases which harness power from them. The bases are semi-organic and have partially grown into the ground around them. Crystals "vibrate" and emit their own noise which illuminates them somewhat. Large crystals are used in levels which end a series of like puzzles. Figure 21 shows an example of the large crystal, as seen in game.

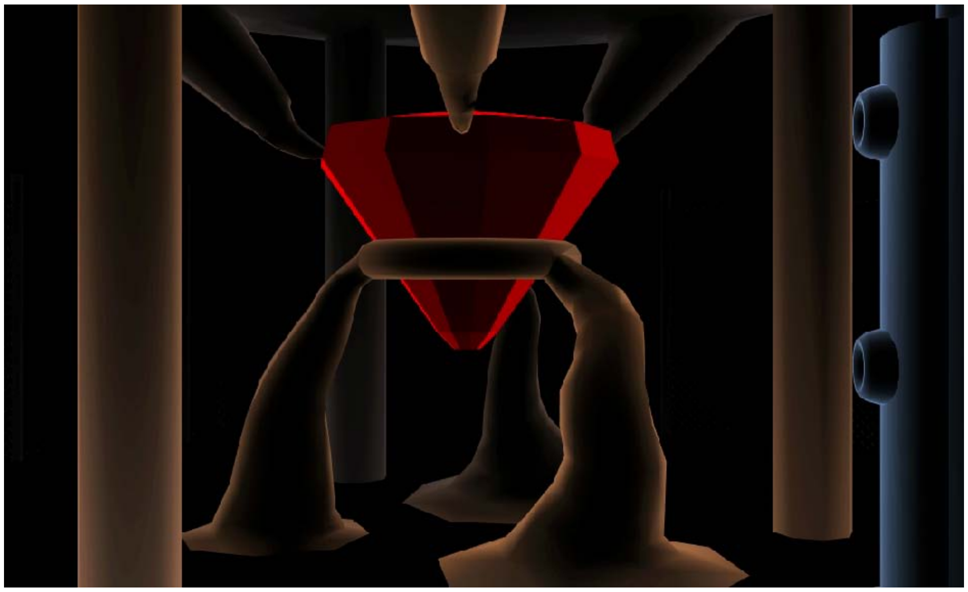


Figure 21: A large crystal

3.2.5 Force Fields

Every crystal enclosure is accessible via an energy force field. They appear as shimmering bars between a pair of posts, which impede the player and emit a dull humming noise while active. Every force field is connected to a switch controlled by an operator. Figure 22 shows an example of a force field protecting a crystal.

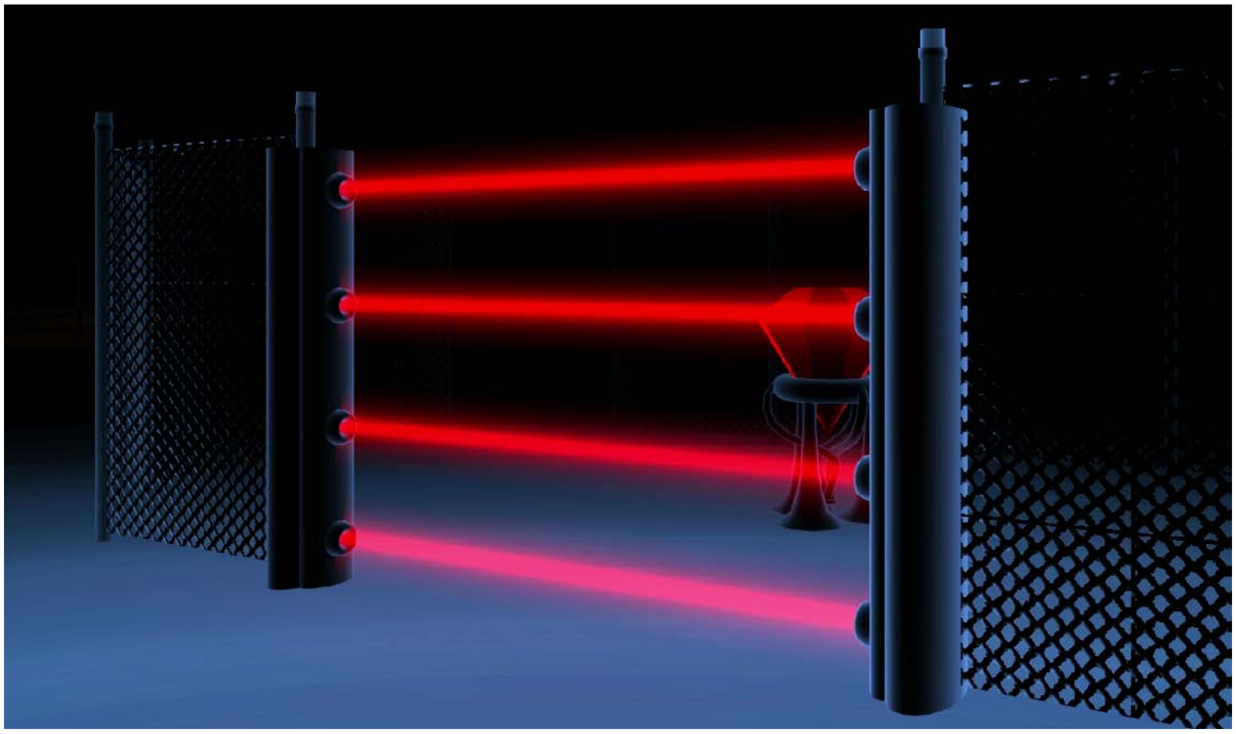


Figure 22: A force field

3.2.6 Switches
There are two types of switches, one made out of wood and one of stone. They include a receptacle into which an Operator can insert his lever attachment, not unlike a keyhole. Each lever also has a moving mechanical portion, involving pieces moving up and down the supports and generating a material appropriate noise on each fall. This makes the switch self illuminating, allowing the player to easily determine which switches are available in a given level. Figure 23 shows an example of a stone switch.

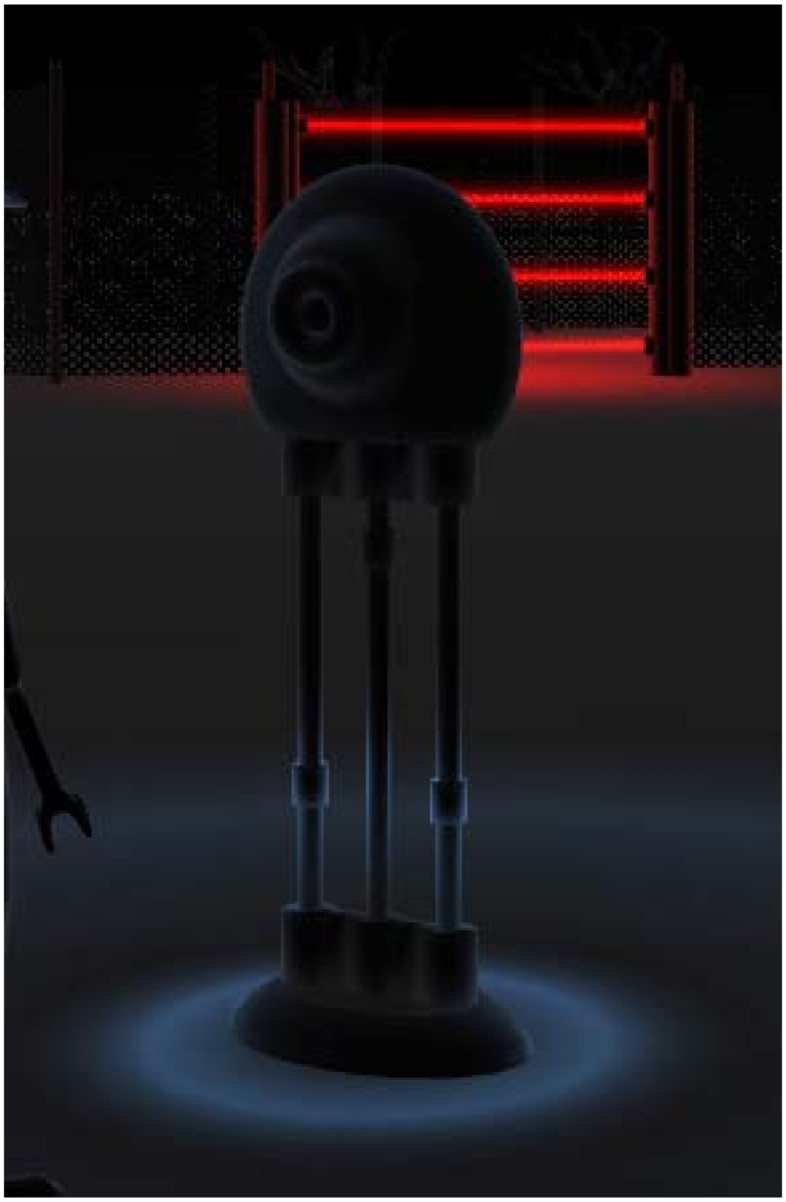


Figure 23: A stone switch

3.3 Character Design
The four characters in *Blinding Silence* are different types of workers who were once the humans responsible for the industrialization and development of the forest in which the game takes place. Their productive efforts have devolved into mindless repetition of the tasks they were given, their purpose forgotten but their muscle memory intact. The artistic style of *Blinding Silence* does not provide much

detail to the player, so the characters must be clear and recognizable. More so than in most games, it is important for the shape and actions of the characters to be obvious from their silhouette alone. The tools which were once grasped in human fingers have been fastened to their arms to expedite the drones' labor. These exaggerated elements allow the player to easily recognize the different entities around him.

Characters in *Blinding Silence* are based on generic human forms, which feature realistic proportions along with a single exaggerated element. As they have been doing their jobs repetitively for a long time, they have started to change so as to do the job better, and their form reflects this fact. Each type of actor also has distinguishing clothing and props so that the player can recognize them at a glance. Any character needing to switch between multiple props, namely the craftsman and harvester, do so by removing the attachment on their arm and replacing it with another.

All bipedal characters use a single unified skeletal rig, allowing for transfer of animation between all characters. All bipeds are based on the same basic mesh with a socket on an arm to hold props, and a hand on the other. They differ only in clothing worn and props used. The one exception is the Carrier whose mesh has two sockets instead of a hand.

3.3.1 Harvester

The Harvester's job is to reap nature's resources, providing the workers with their supply of wood and stone. With his axe attachment he chops at trees which will never fall, and his pickaxe attachment is for mining stone. He wears a hard hat and sports a large exaggerated lumberjack beard. Figure 24 shows the Harvester, as a concept on the left and as a final model on the right.

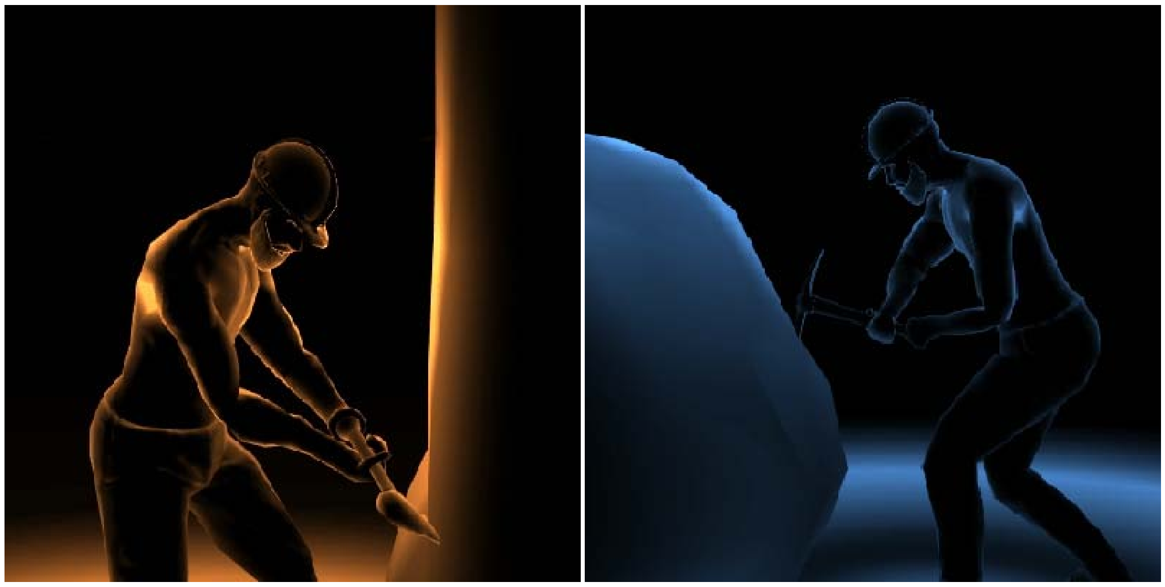


Figure 24: Harvester

3.3.2 Craftsman

The Craftsman, seen in Figure 25, is responsible for building the various manmade structures found in the game world. He relentlessly works at wooden scaffolds or stone blocks, without making any actual progress in either case. He wears a hard hat, and a pair of safety goggles. When working on wood, he uses a hammer attachment and holds the structure with his left hand. When working on stone, he uses a mallet attachment and holds a chisel in his left hand.

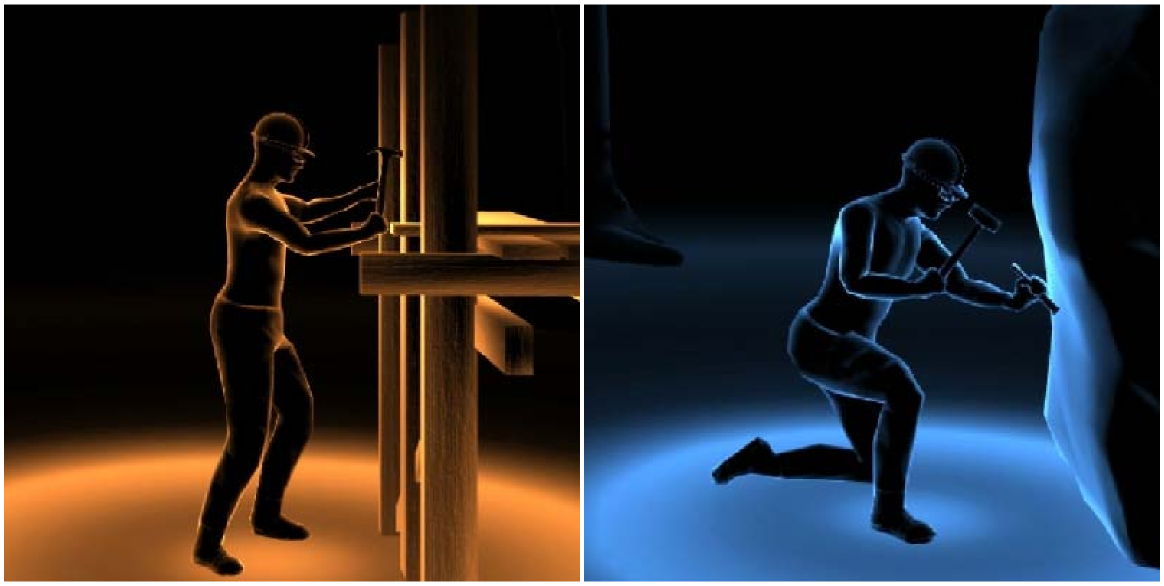


Figure 25: Craftsman

3.3.3 Operator

The Operator, Figure 26, is in charge of controlling access to the various power crystals throughout the levels by means of activating and deactivating force fields. He stands at his assigned switch and activates it by inserting his lever attachment. He wears a hard hat and sunglasses. When not actively pulling his lever, he stands idly and waits to hear the sound that will activate him.

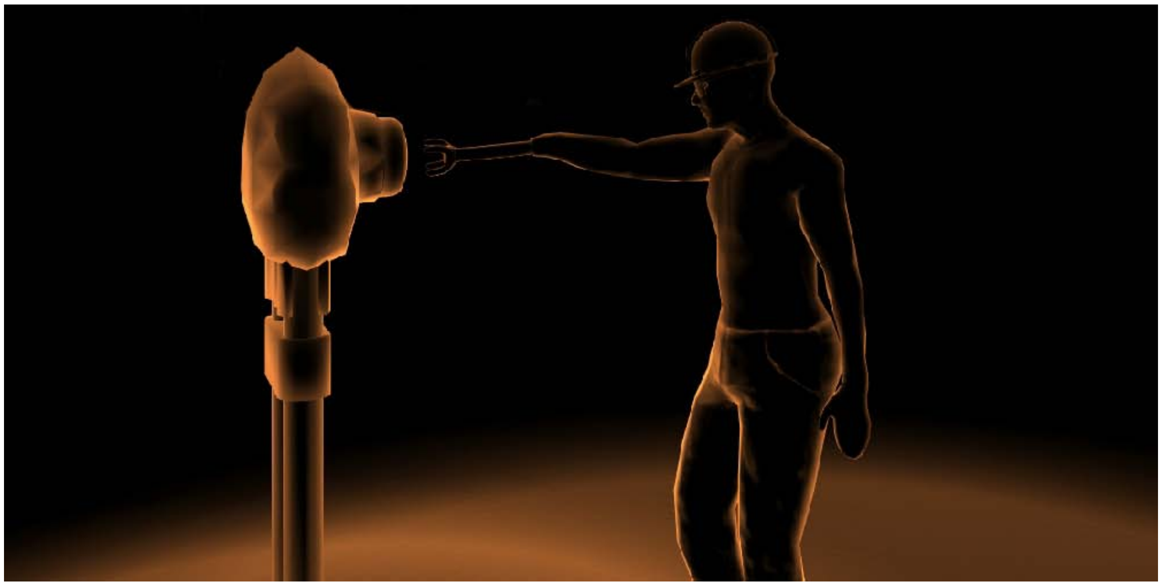


Figure 26: Operator

3.3.4 Carrier
The Carrier, Figure 27, is charged with the transportation of resources. Materials (wood and stone) collected by the Harvesters must be administered to Craftsmen building structures. The Carrier hauls logs and stones from pick-up points to drop-off points across levels. When laden with a load, he walks half as fast and his gait changes to match the weight and position of what he is carrying. In his right arm is a claw for carrying logs over his shoulder. In his left arm is a four-pronged grabber for carrying stones at his side, similar to the grabbing arm in arcade machines.

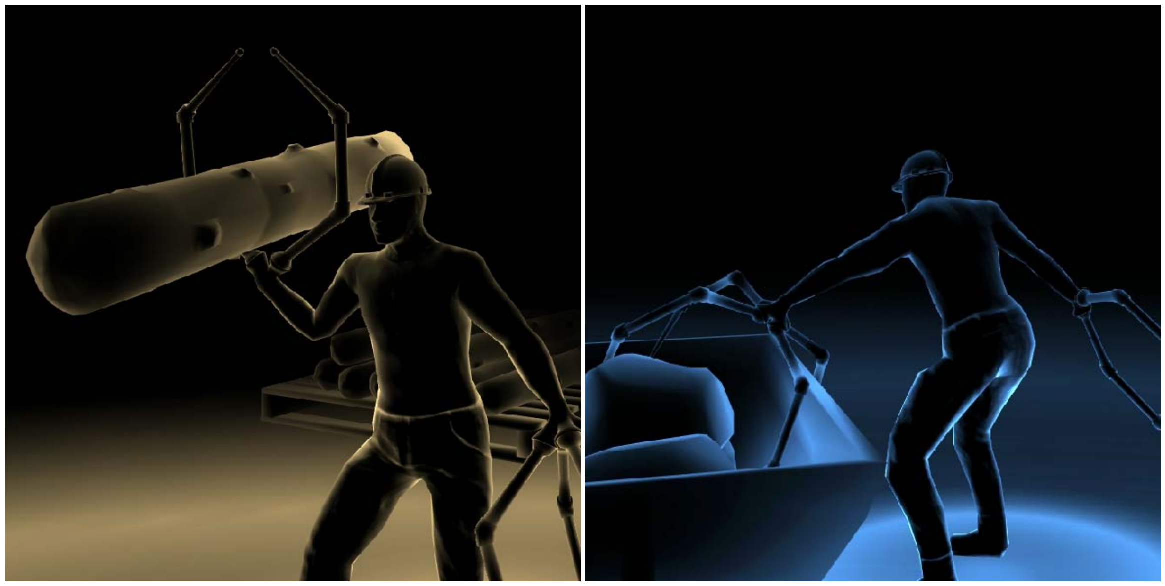


Figure 27: Carrier

3.4 Animation
In a world with little visual detail, accurate movement is important to help maintain believability. The diverse range of characters and interactions in *Blinding Silence* required large amounts of high quality animation, which would have been difficult to achieve in a short development time if created by hand. Instead, animation was created using motion capture data, allowing for both increased realism and decreased development time.

3.4.1 Stock animation
Various sources online provide access to free motion capture data. Originally, we had planned to get much of our animations from this freely available data. We were under the impression that this data would have already been cleaned and would save a great deal of time in creating the animations. Unfortunately, when we started looking through the available data, we found that there were very few animations that were on our list of requirements. Of the ones that were, they were all useless for our purposes for one reason or another. In the end, we chose to stop trying to find pre-captured animations, and chose to capture all of them ourselves.

3.4.2 Studio Setup
The WPI HIVE[2] lab owns two PhaseSpace[3] motion capture systems that were not in use at the start of the project. Over the course of B term, we tested various ways of setting up the systems to produce optimal data. Due to the physical constraints of the space available to us, we were never able to find a configuration that would produce perfect data. In the end though, using a combination of the cameras from both capture systems, we were able to capture data that was clean enough to be processed for our animations.

3.4.3 Capture and Cleanup
Near the end of B term, once the system was able to produce usable data, we had an actor from an on-campus improv group come in and perform the motions we needed. We ended up capturing almost 12Gb of data over 36 takes.

From the end of B term through to the first weeks of C term, the data was cleaned in Motionbuilder. This was very time consuming work because the data was not very clean, particularly due to an important marker that had come un-plugged in the first few takes of the capture and was not noticed until much later. This caused the right arm to often solve improperly. In addition, the setup we used for capture of the hands resulted in very dirty data that often required hours of intricate work to resolve.

The final result of the motion capture was roughly 20 animations spread over the 4 actor types.

3.4.4 Integration
Initially, we simply put each animation into the engine in its entirety. However, most of the animations required that some action be performed by the game part-way through playback. For instance, footsteps must be emitted at the proper times, and a harvester needs to emit a sound when he hits a tree. Since C4 only provides a callback for the completion of an animation, we decided to split each animation into two pieces. At the completion of the appropriate segment, the engine takes the proper action and then plays the other portion of the animation as needed.

The other concern with animations had to do with blending between them. C4 is advertised as having an animation blending mechanism that will smoothly blend from one animation to the next. For this reason, we chose not to make individual blending animations for each state change. Unfortunately, after several attempts, we found that the C4 blending did not work properly. By that time it was too late to devote further resources to creating all of the animations need for state changes. As it turned out, due to the inherently dark nature of our game, the player is rarely able to see the state change well enough to notice the lack of animation blending, so we eventually chose to leave the animations as they were rather than trying to schedule time to find a better solution to the blending problem.

3.5 Storytelling and Instructions
In the original design of the game, we wrote a detailed backstory that explained why the world was as the player saw it, along with a story arc and end-game. We planned to deliver the story through video and

[2] http://web.cs.wpi.edu/~gogo/hive/

[3] http://www.phasespace.com/

assorted stills. Due to time, we had to cut the videos and boiled the story down to its essence as stills bundled with the instructions.

3.5.1 Animatics

Instead of using live cut scenes, the story of *Blinding Silence* was to be told through an introductory animatic, stills shown between each level, and a final animatic at the end of the game. The content of these was to come from two sources: first, rendered stills of certain in-game objects would be used to help maintain the connection to the game world. Second, photographs would be used to provide the settings of the animatics, along with objects and places that do not appear in the game. This content would then be composited and animated to match narrated voice-overs of the player's cane. The images would then have various filters and effects applied to give them a distinctive visual look, similar in style to the visuals found in-game. The introductory animatic script and storyboards can be found in Appendix F and Appendix G, respectively.

Figure 28 shows an example of portions of an animatic depicting darkness and silence falling across the world. Proceeding from 'a' to 'd', the city goes from completely realistic to a look that better fits the game world as seen by the player.

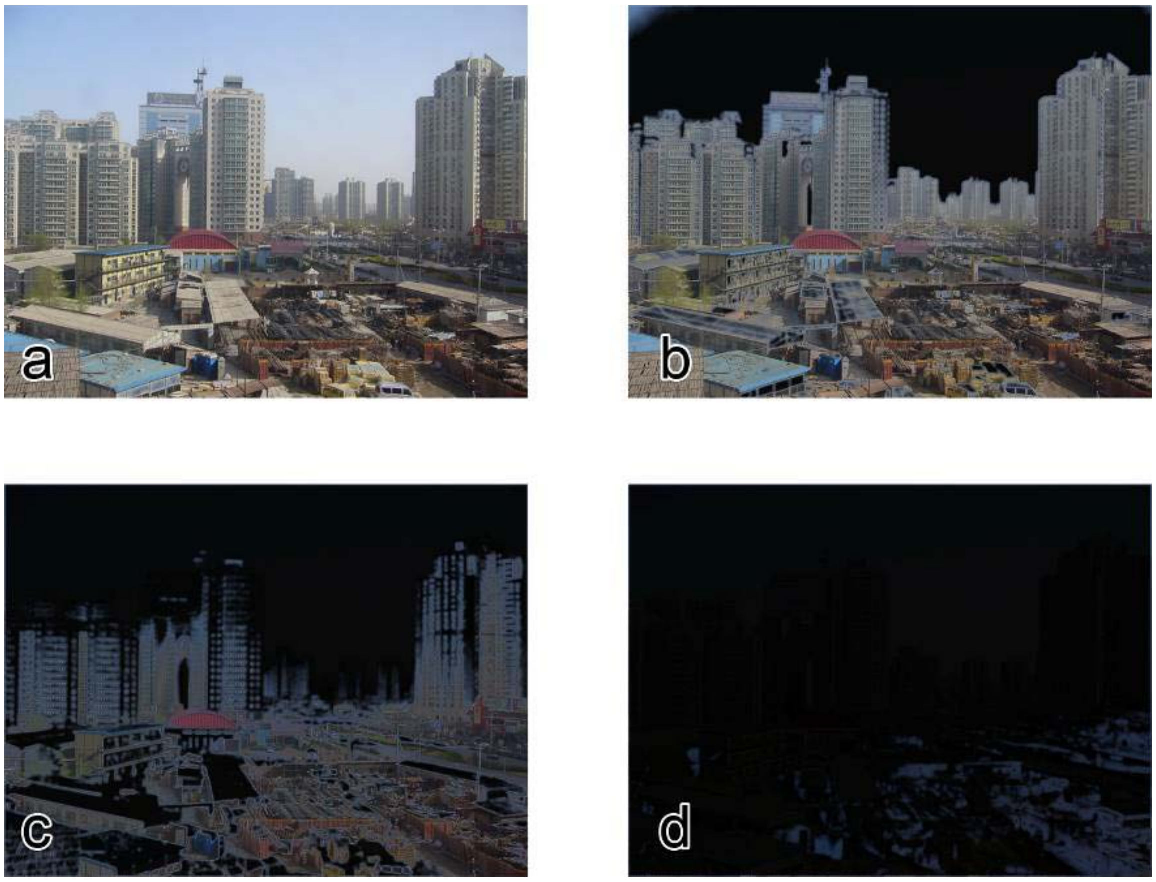


Figure 28: Animatic Progression

Figure 29 shows different variations in world style, stylizing a real image of a forest in a way that is closer to the simplified look of the in-game world. They range from very detailed and realistic in 'a', to very stylized and sparsely detailed in 'd', with images 'b' and 'c' providing more intermediate levels of detail. The amount of detail in any given shot would have varied between these styles, depending on the tone and content at that time. Animated characters would then be added to these images, and colored sound-waves overlaid to help reinforce the image and the connection to the in-game world.

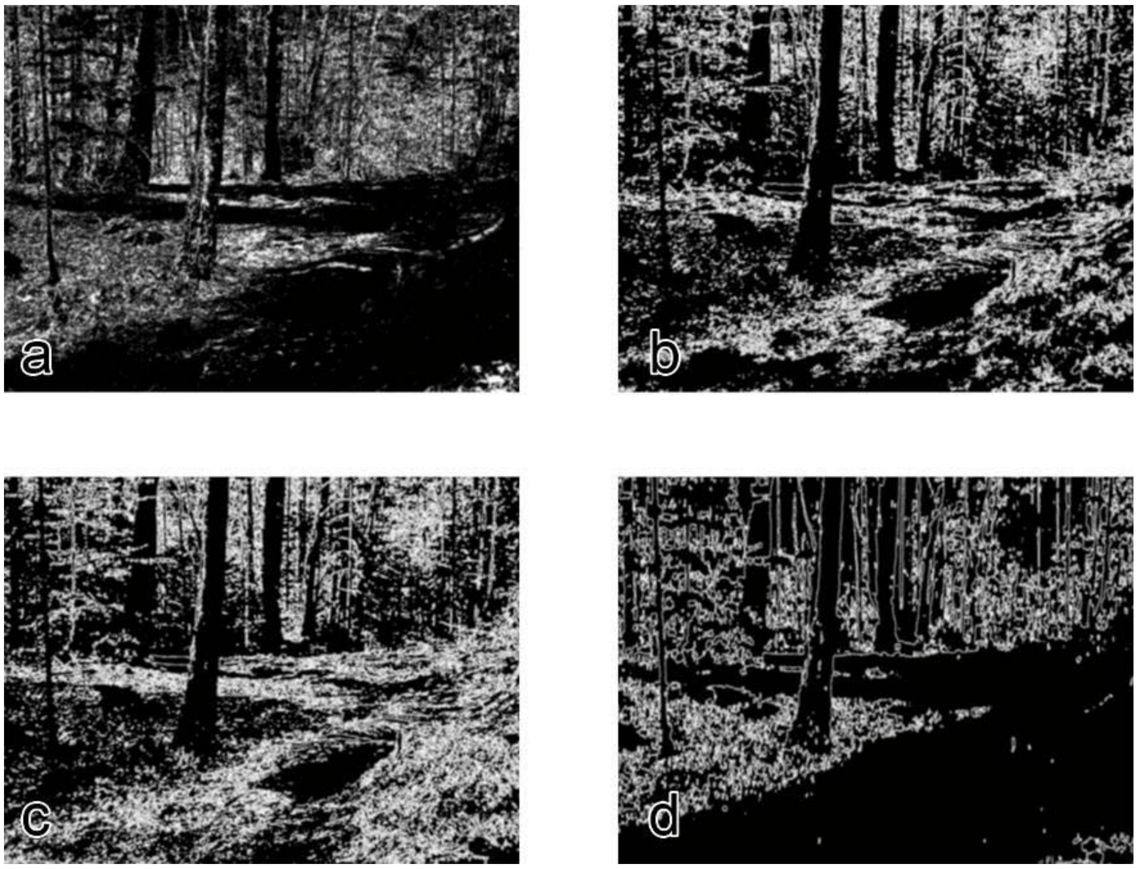


Figure 29: Example Styles

3.5.2 Stills

Because the animatics and in-game narration were both cut from the game, the story of *Blinding Silence* was abbreviated and bundled with instructions regarding the controls, rules, and mechanics. This information takes the form of a series of stills given to the player in the form of a slide show. These stills (which appear in Section 2) consist of pictures, either taken from screenshots of *Blinding Silence* or modified stock images, alongside text describing the aforementioned information. Effort was taken to maintain a coherent style among these stills and with the style of the game as a whole, particularly other UI elements. Contextual stills were created to display messages regarding events such as level completion, as well as the credits and integrated into user interface code.

3.6 Acoustic Vision

Blinding Silence does not have a traditional sound score. All sounds that the player hears, even ambient sounds, are connected to a visible cue in the game world. Through the use of a global game clock, these sound effects are constrained to a particular rhythm. This effectively creates dynamic game "music" based upon the positions and states of the player and level entities.

Sound effects were created using a mixture of audio recorded by the team and royalty free sound effects from online libraries. The manipulation of real life sounds to give them a more instrumental feel enhances the audible experience and compensates for the intentional lack of pre-composed music. The sound of a pick axe striking a stone with a constant timing becomes a percussive beat in the "song" generated by the level.

In addition, sounds are further associated with their visual counterparts through the use of color. Sound waves which illuminate the world are colored; each type of sound is represented with a distinct hue, allowing the players to identify level elements both audibly and visibly.

3.6.1 Voice Acting

We originally planned to have the player's cane act as a mentor through the game, teaching him how to interact with the world and revealing new pieces of the story to him. Even the animatics would be told from the perspective of the cane as an immortal being. To this end, we found a student to provide voice acting and began to draft the various scripts of necessary dialog. Several takes of the introduction animatic were recorded, but they were never processed and finished because more important issues arose. Once it became clear that there wasn't time, the dialog and voice-overs were cut from the game and replaced with text in the instructions.

3.7 Artistic Tools

Levels in *Blinding Silence* were created using a combination of Autodesk Maya[4] and the C4[5] level editor. Characters and props were created in Zbrush[6], and optimized for importation using Maya. Animations were created using a combination of Maya and Motion Builder[7], along with motion capture data gathered specifically for the project. Adobe Photoshop[8] was used for 2D content as needed, including the menu interface and a limited number of textures; there was little demand for textures as the look of that game was primarily achieved through the creation of custom shaders. Finally, the planned animatics were to be created using Adobe After Effects[9], but they were eventually cut due to time constraints.

3.7.1 Integration and Level Pipeline

Integration of game assets and levels was accomplished using a series of tools built in both Maya and C4. The C4 world editor can be difficult to use when trying to layout levels. Tasks that may normally take a few seconds in a standard DCC[10] application often end up requiring significantly more time and effort. Maya, on the other hand, is a very powerful tool that makes level layout very quick. However, when using

[4] http://www.autodesk.com/maya

[5] http://www.terathon.com/c4engine

[6] http://www.pixologic.com/zbrush

[7] http://www.autodesk.com/motionbuilder

[8] http://www.adobe.com/products/photoshop

[9] http://www.adobe.com/products/aftereffects/

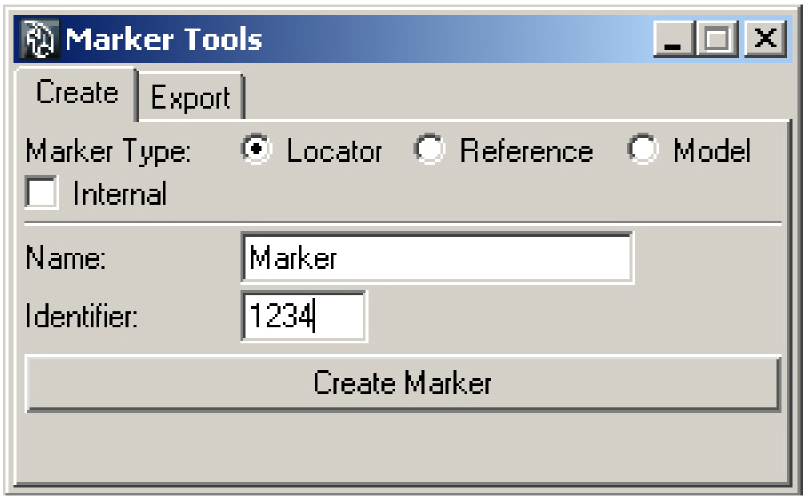
[10] DCC stands for 'Digital Content Creation' and is used here to refer to standard applications used by the gaming industry for generating visual and artistic content.

Maya for level layout, the designer has no access to important C4 structures such as markers and world referencing. The tools we created were designed to create a bridge between these two applications, allowing the designer to leverage the efficiency of Maya while retaining the ability to utilize various C4 specific constructs.

3.7.1.1 Asset Setup

One important aspect of C4 is the concept of referencing worlds into a main scene. The level designer can place markers at a series of locations and set them to all reference the same world. By creating these references, the engine can conserve resources and be more efficient compared to having discrete duplicates of a given piece of geometry. Referencing a single world also provides an added benefit in that changes to an object that appears multiple times in each level need only be made in a single file, as the changes will be propagated to any level referencing that file. We wanted to find a way to let the designer place these, and similar, sorts of markers in Maya built levels.

Maya provides a node type called a locator, which is used for the sole purpose of defining a transform in the world in the same way that C4's markers do. We decided that these locators would be the best choice for the Maya end of the marker placement pipeline. By naming them in a specific manner, we were then able to export them and let the C4 importer determine the appropriate type of marker to place in the world (see section 3.1.1.3). We created a tool in Maya, seen in Figure 30, which will generate an appropriately named locator at the scene origin, which could then be used to determine placement of that object in C4. The tool will also add special flags to the markers, allowing the exporter to easily find them later. The user chooses a C4 marker type (locators, references, or a model), gives the marker a name, and assigns it a unique four character identifier that C4 can recognize. The name output from the settings in the figure would be "LOCATOR_1234_Marker".


Marker Tools

Create | Export

Marker Type: [✓ Locator] Reference Model

Internal [ ]

Name: Marker

Identifier: 1234

Create Marker

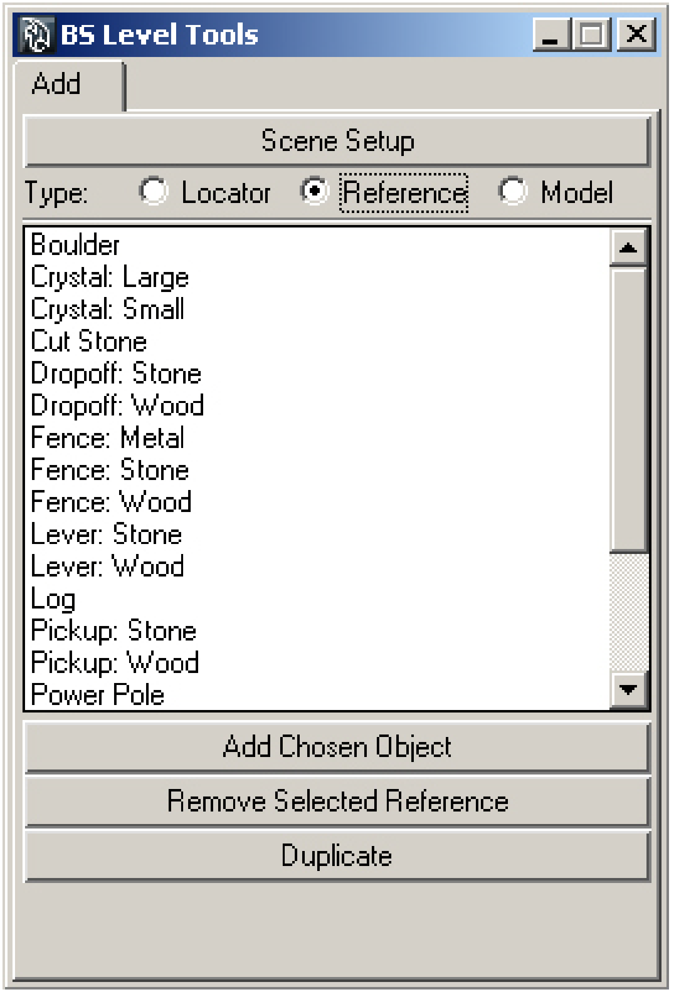
Figure 30: Marker Creation Tool

To create the actual C4 assets, the artist follows the standard C4 pipeline of exporting the desired objects from Maya and importing into a C4 world file. No special steps are needed in the editor. All of the data needed for the custom pipeline lives in the Maya locator markers, and not in the imported C4 worlds being referenced.

3.7.1.2 Level Creation and Exporting

The next step was to create some way for the level designer to place objects in the world. As in C4, Maya also includes a referencing mechanic that allows files to be referenced into a scene. Using Maya's referencing system seemed to be the best choice, but forcing the designer to manual find the right files each time would be tedious and would make the process very slow. Realizing this, a second tool was designed that would provide a list of objects that C4 is setup to import, allowing the designer to easily choose the required object without having to know what file Maya was actually referencing.

Figure 31 shows the final version of the tool that was created. The user can select from three lists (Locator, Reference, Model), corresponding to the three types of markers being imported by C4. The user then selects the desired object and presses the "Add Chosen Object" button to reference it to the scene's origin. Due to the nature of Maya's reference system, standard object deletion and duplication breaks the link between the source file and the reference. To combat this issue, the tool also includes buttons to perform helper functions, such as easily removing a reference and duplicating one or more references while maintaining their positions. Finally, it includes a button at the top named "Scene Setup" which will perform a series of operations on a default Maya scene to adjust it to match the requirements of the C4 importer.



Type: Locator [✓ Reference] Model

Boulder
Crystal: Large
Crystal: Small
Cut Stone
Dropoff: Stone
Dropoff: Wood
Fence: Metal
Fence: Stone
Fence: Wood
Lever: Stone
Lever: Wood
Log
Pickup: Stone
Pickup: Wood
Power Pole

Figure 31: Level Layout Tool

Once the designer has completed a level, he must export it in a way that C4 will be able to understand. Simply exporting the references will not work because each piece of referenced geometry would be converted into discrete duplicates. Instead, it is necessary to export the locators that were setup in the individual referenced worlds. This is where the final Maya tool, shown in Figure 32, is used. By pressing the “Group/Populate” button, the tool will search through the entire scene graph for markers with certain custom flags on them. For each one it finds, it will check another flag to determine which type of marker it corresponds to and will add a copy of that locator to a group. When the process is complete, all of the locators have been duplicated and placed into one of three groups based on their marker type. The user can then select these groups, and any custom geometry added to the level, and export using the standard Collada exporter as usual. The other buttons shown in the tool will perform the individual steps of the process, if needed.

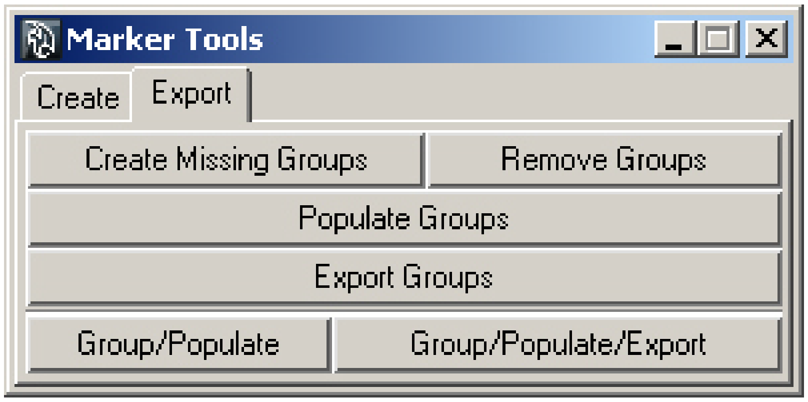


Figure 32: Level Exporting Tool

3.7.1.3 C4 Importing

The stock C4 importer is capable of recognizing Maya locators and will turn them into locator markers if their name begins with “MARKER_”. Using this basic functionality, we created three additional keywords to accept: “LOCATOR_”, “REFERENCE_”, and “MODEL_”. If any of these are found, the importer will strip out the keyword and create an appropriate marker of the given type. The process of creating each type of marker is slightly different.

3.7.1.3.1 Locator Markers

Locator markers are the easiest to import, as the name is simply divided and each part is put into a specific spot in the marker. Using the example from section 3.7.1.1, “LOCATOR_1234_Marker”, the importer will create a locator marker with a type of ‘1234’, and will give it the name “Marker”.

3.7.1.3.2 Reference Markers

Reference markers are more complicated. These markers need to store a file path to the world they are supposed to reference. However, trying to store the full file path in the name presents a number of issues in both Maya naming and in determining the length of the path name. Instead, the four character identifier is used to reference a string table. In this table is a list of identifiers and the file paths they correspond to. Once the file path is found, a reference marker is created that points to this path, and is named in the same manner as locator markers. If the identifier is not listed in the table, a reference marker is still created but it will be set to not reference any world.

3.7.1.3.3 Models
In C4, model types are four character identifiers that are registered with the engine on startup. Once again, the name used to import the marker can simply store an identifier. When importing a model locator, the importer will ask the engine to provide an instance of the model registered with the given identifier. If found, the model will be named as with the other two types and will be added to the world. If the given type is not registered with the engine, the importer will generate what is known as a “generic model”, which will initially be set to a blank model type.

3.7.1.4 Future Work
In general, the pipeline we created helped to drastically speed up the creation of levels as intended. However, the tools are not perfect and it would be useful to implement certain extensions should the tools be used for future work.

3.1.1.4.1 Hard-Coded Values
In their current state, the tools require a non-trivial amount of work if an asset changes type or a new asset is added. On the Maya side, the Markers stored in the referenced files describe how the engine should interpret them on import, and have to be manually replaced if changes need to be made. Also, the lists of available assets and their location in the placement tool are hard coded Python dictionaries that must be manually kept up to date. On the C4 end, the model registrations and the referencing string table have to be manually kept in sync with the data being output by Maya.

An ideal solution to the problem would be to keep a data base of assets, storing their identifier, marker type, and file locations for both Maya and C4. In this way, a single user interface could be used to register assets with both Maya and C4, allowing for simpler maintenance.

3.7.1.4.2 Placement and Manipulation
In the current implementation, new assets are initially placed at the scene origin. This can become aggravating as the designer must navigate back to the origin any time a new object is created. The problem is somewhat mitigated by the ability to duplicate groups of assets in place, but this feature doesn’t help new asset creation. A possible extension to solve the issue would be to allow the designer to click on the location where the object should be placed, possibly with the option of randomized rotation and automation of sinking the object into the ground. A tool like this would allow for much faster placement of objects; for instance, a forest of semi-randomized trees could be placed with a series of clicks, rather than manually duplicating and transforming each tree by hand.

Also, due to the nature of referencing in Maya, the designer can select each piece of an asset instead of only selecting the parent node that should be used for placement. It would be useful to find a way to encapsulate the referenced objects such that the designer can only see the single node used for manipulation and placement.

3.7.1.4.3 Connections
C4’s editor allows the designer to create connections between game objects that can then be accessed in code. We chose not to design this functionality into the Maya tools because it was not needed for this project. In the initial plan, any necessary connections would be made based on the proximity of certain objects to one-another. However, once we started to make levels we realized that it was important to have

control over how objects were connected. Unfortunately, it was too late to design a way of creating connection from within Maya, and so they have to be made once the level is imported into C4. In the future it would be useful to have this functionality available in Maya so that levels don't need further editing once they are imported into C4.

3.1.1.4.4 Controllers and Properties

C4's editor also allows for the ability to add controllers and properties to objects, and adjust their attributes. Initially, we had intended to not have customizable attributes on the controllers; the game would assign the controller based on model type and would then use object proximity to make any other decisions. As with connections, it soon became apparent that we needed to have control over various attributes of the controllers. We also added several custom property types to the game that needed to be set during level creation. It would be useful to create a way of registering controllers and properties with Maya and exporting them in a way that C4 can later interpret.

4. Technical Design

This section contains all relevant design challenges and solutions, as well as all relevant diagrams and charts, and justifies our technical design choices.

4.1 C4 Engine

We chose the C4 engine for four reasons:

- Every team member worked with C4 before
- C4 has a dedicated developer community
- C4 supports 3D sound
- Terathon provides full access to engine source code

4.2 Technical overview

This section will summarize the technical achievements accomplished during the development of *Blinding Silence*.

4.2.1 Interfacing with Wiimote

A way of interfacing with Wiimotes was required for this project. The Tech team decided to choose WiiYourself version 1.14 Beta, a native C++ Wiimote Library. WiiYourself allowed us to easily integrate Wiimote support into C4.

4.2.1.1 Basic Wiimote functionality

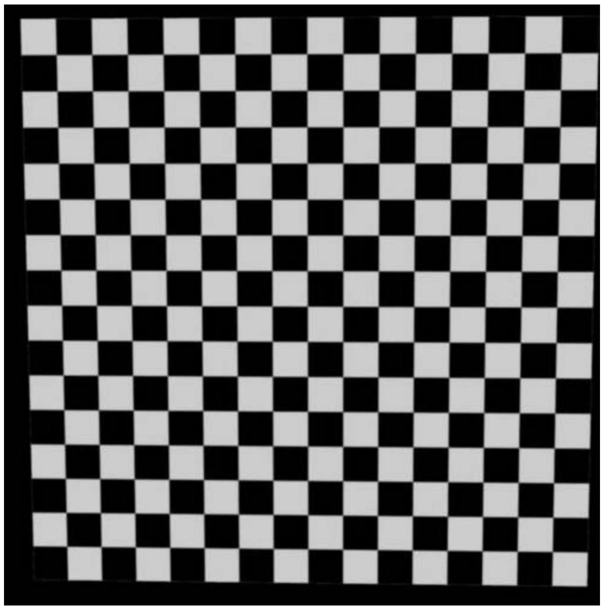
The WiiYourself library provides direct access to the data state of the Wiimote, which allowed us to directly monitor the buttons and analog stick. Every engine tick the Wiimote state is checked, and any relevant change in state is dealt with by other functions.

4.2.1.2 IR Data Receivers

IRDataReceiver is a generic class for handling IR data coming in from the Wiimotes. The receiver class has a one-to-one relationship with each Wiimote; IRDataReciever takes in up to four points and stores them for later access. This class also dynamically resizes the distance between tracked points, which helps regulate the changes of position of the points in order to smooth in-game mappings.

We used this class to both track head movement and the Wiimote held by the player. They are dealt with identically by this class, and are handled separately by the WiiCaneController and WiiCameraController.

There is a small amount of distortion effect, shown in Figure 33, as the wiimote is moved due to the effect of mapping data from a 2D plane onto a sphere. We mitigate this effect by taking the average of two points on the plane when calculating rotation, which lowers the effect of the distortion.


This checkerboard represents the data points we are getting.

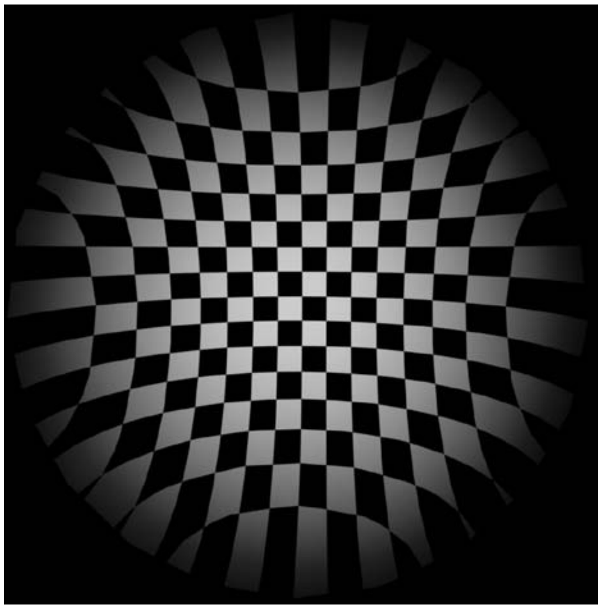

This checkerboard represents the data points when they are mapped two a sphere. As you can see, a distortion effect occurs that gets worse as you aproach the edges.

Figure 33: Wiimote distortion

4.2.2 Sound Design

Gameplay in *Blinding Silence* focuses on sounds being constantly emitted and received by both the player and actors. To do this, the property BSMaterialProperty is associated with every object in the game. There is a type of BSMaterialProperty for every class of sound in the game—wood, stone, metal, dirt, and crystal. Any object without an associated BSMaterialProperty is assumed to be dirt.

Blinding Silence contains two classifications of engines in order to facilitate sounds and actors' responses to them. The sound engine manages the sounds themselves, and the resource engine manages objects that actors will interact with.

4.2.2.1 Sound Engine

The sound engine handles sound for the game. It is responsible for notifying actors when they hear sounds. Actors register themselves with the sound engine when the game is loaded. The sound engine is responsible for keeping track of registered entities and destroying old sounds.

The sound engine tracks a list of sounds and sound receivers. The sound engine is notified every time a sound is made, and creates the appropriate sound and related light as appropriate based on the sound information passed to it. Every frame the sound engine checks how far sounds have traveled, updating the lights and notifying actors as appropriate. This concentrates sound-based responsibilities into one easily tracked class.

During development the sound engine was purposefully not made a singleton class to allow for multiple systems of sounds that do not interact to exist in the same level. For example, all of the "background" actors performing actions outside the playable area could be hooked up to a second sound engine in order to remove the possibility of accidental interaction. Although *Blinding Silence* does not currently support this functionality, keeping this as an option makes development of more complicated levels a practical possibility.

Figure 34 shows the steps *Blinding Silence* goes through to generate its sounds.

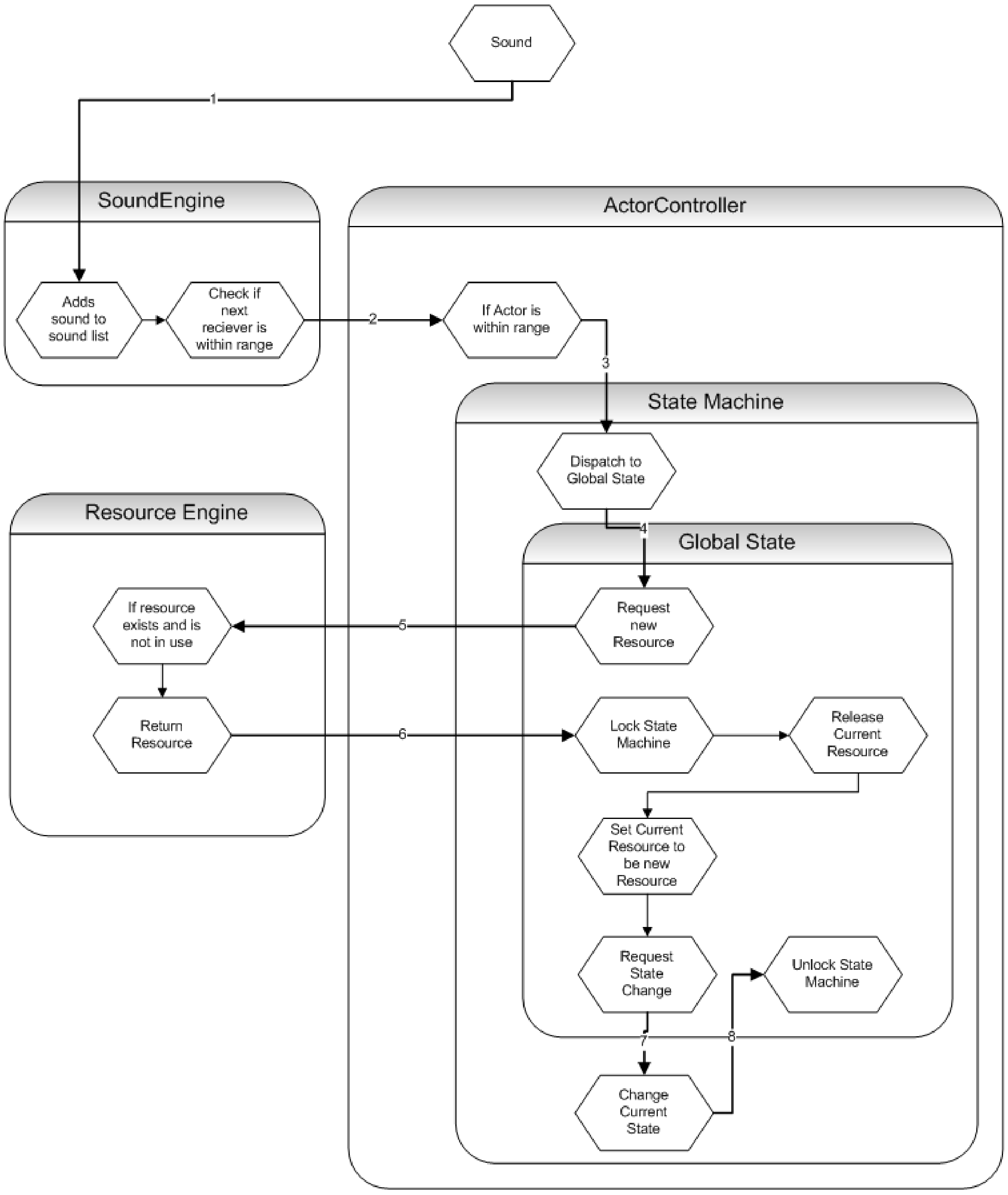


Figure 34: Sound reception flow

1. A sound message is passed to the sound engine
2. The sound engine passes the sound message to the actor controller
3. The actor controller passes the sound message its state machine
4. The state machine passes the sound message to the current global state
5. The global state queries the resource engine for a new resource
6. The resource engine returns a valid resource, and the global state locks the state machine
7. The old global state unlocks the state machine.

4.2.2.2 Resource Engine

The resource engine is responsible for keeping track of all in game resources. When an actor needs a new resource, it queries the resource engine with a resource type and a material type. If there is a free resource available, the engine returns the resource. If there isn't a resource, it returns null.

In game resources automatically register themselves with the resource engine when a level is loaded. The resource engine tracks types of resources as separate lists in order to expedite access. Lists are split based on the associated resources' BSMaterialProperty as well as the actors that check with them.

The resource engine is also not a singleton, and as with the sound engine this allows for situations where actors are set up on different systems of resources. While this might only be used for unrelated actors in the case of the sound engine, discrete resource engines could be used to create complicated puzzles where actors are separated by barriers.

4.2.4 AI

This section describes the logic behind the non-controlled characters present in *Blinding Silence* called 'actors.'

4.2.4.1 Overview

Puzzle elements in *Blinding Silence* are "mindless" workers that wander the environment performing actions repeatedly. These actors perform actions in response to sound stimuli. *Blinding Silence* implements a state machine to handle the actors, functionality in place to deal with both the sound and resource engines in order to change what they are doing.

4.2.4.2 State Machine

Each actor in the game has a State Machine, which stores the actor's current state and handles the process of transitioning from one state to another. Each State Machine has two states, a global state and a local state. The global state is executed before the normal state is executed, once per frame.

The State Machine also supports message passing. When the State Machine receives a message it passes it to the current global state to be handled. If the global state is unable to handle the message, the message is then passed to the local state to be handled.

The State Machine supports a form of locking. When locked, a state machine is unable to dispatch SoundMessages to the states. This prevents the changing of global states during critical animations, such as when harvesters change between an axe and a pickaxe, or when carriers drop logs.

4.2.4.2 Messaging System

The implementation of the state machine includes a message passing system. Messages are dispatched to the actor's global state first. If the global state is unable to handle the message received, the state machine dispatches it to the local state.

Setting up the message system this way interfaces with increasingly complicated layers of states to provide for specialized states that handle special instances. During development this was used to solve a variety of challenges; for example, operators do not transition back after their DoAction state. A special Operator::Wait state was created for the operator to switch to, which required a specialized DoAction state as well. Messages are simply passed down until they either find the state that deals with the message or returns that the message was not handled.

4.2.4.3 Global States

Global states store state transition logic that would otherwise have to be in every state. Including global states cuts the number of states we need in half. Consider Figure 35.

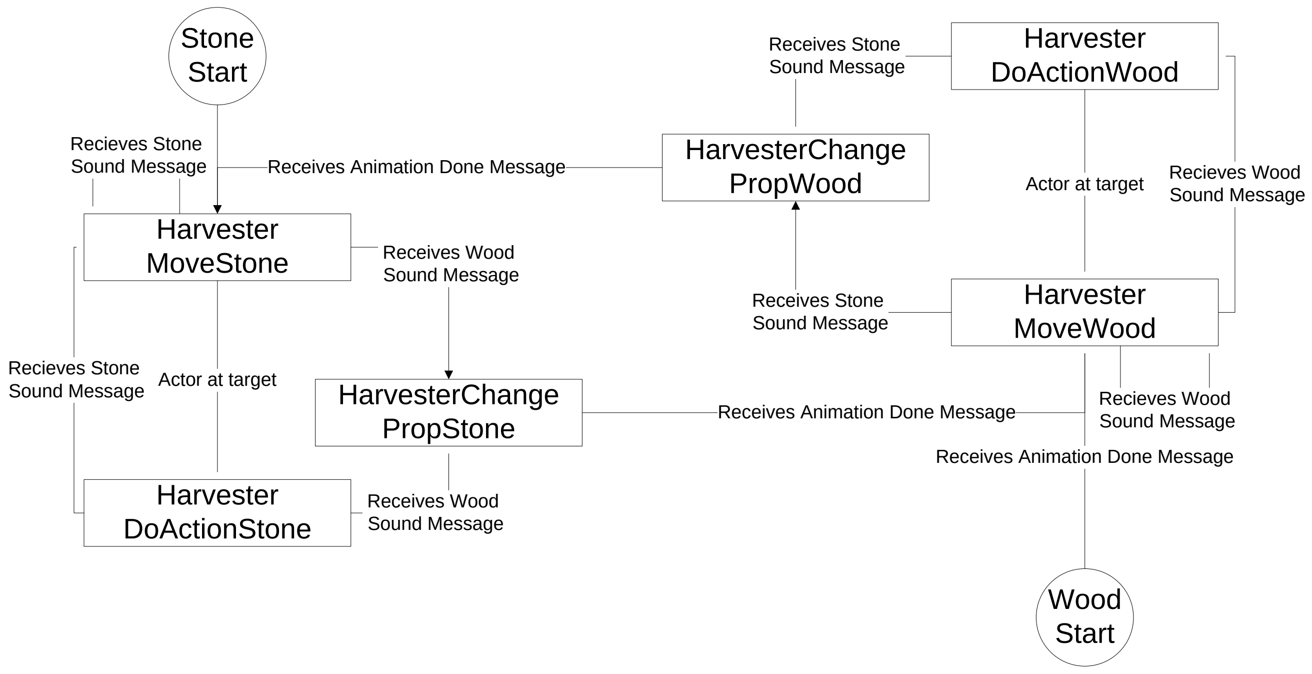

Stone Start
Harvester MoveStone
Harvester DoActionStone
HarvesterChange PropStone
HarvesterChange PropWood
Harvester DoActionWood
Harvester MoveWood
Wood Start
Recieves Stone Sound Message
Receives Animation Done Message
Receives Wood Sound Message
Recieves Stone Sound Message
Actor at target
Receives Wood Sound Message
Receives Stone Sound Message
Actor at target
Recieves Wood Sound Message
Receives Stone Sound Message
Receives Animation Done Message
Recieves Wood Sound Message
Receives Animation Done Message

Figure 35: State machine diagram without global states

There are a large number of transitions in this diagram. There are also a large number of duplicate states whose only difference is containing wood- or stone-specific state transition data. The solution is not elegant and the number of transitions increases exponentially with the number of states, which is difficult to debug and worse to expand upon.

Now, compare this with Figure 36:

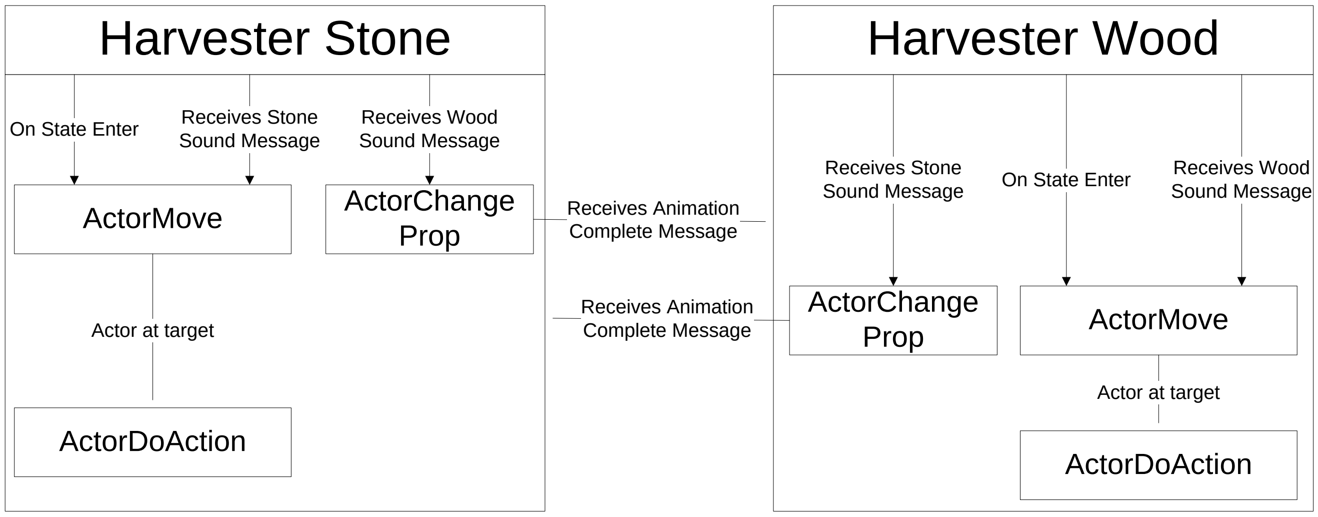

Harvester Stone
On State Enter
Receives Stone Sound Message
Receives Wood Sound Message
ActorMove
ActorChange Prop
Actor at target
ActorDoAction
Receives Animation Complete Message
Receives Animation Complete Message
Harvester Wood
Receives Stone Sound Message
On State Enter
Receives Wood Sound Message
ActorChange Prop
ActorMove
Actor at target
ActorDoAction

Figure 36: State machine diagram with global states

Not only do the global states simplify the process and remove redundant states, but the state transition process becomes linear. Global states are traversed in a consistent fashion, which allows for specialization of certain local states to handle variables set and removed at the beginning and end of global states, as well as further state specializations.

4.2.5.3 Local States

Local states are where the implementation of what an actor does is stored. Local states control the running of actor's animations as well as the timings of his actions. We strove to make the local states as generic as possible so they could be re-used between actors.

In certain cases specialized non-generic states were necessary. These states are switched to in place of generic local states, and contain functionality specialized to certain situations. This was extremely useful during development, as basic actor functionality was put in place immediately, and writing specialized local states could be done modularly as features were added to *Blinding Silence*.

4.2.5.4 Navigation Mesh

Blinding Silence uses a fast, open source library called Recast[11] to automatically generate its navigation meshes for levels. *Blinding Silence* uses a plugin[12] by C4 developer Marko Ludolph to integrate Recast with C4.

Recast creates a navigation mesh over a voxel mold created from level geometry. The mesh is made up of convex polygons to facilitate path finding. The Recast plugin is run once to create the nav mesh, and then saves that navigation mesh. While this limits levels to containing non-dynamic navigation, it lends itself to the non-singleton resource and sound engine techniques discussed earlier.

---

[11] http://code.google.com/p/recastnavigation/

[12] http://www.terathon.com/forums/viewtopic.php?f=4&t=7793&p=76102

4.2.4.5 Path Finding

Blinding Silence uses Recast's sister library Detour, found at the same location, for its path finding. Detour is designed to work with the navigational meshes generated by Recast, and uses the same plug-in developed by Marko Ludolph to integrate with C4. Detour finds a path based on provided start and end points. *Blinding Silence* creates splines for the actors to follow for the following reasons:

- Splines can be quickly drawn from a series of navigation polygons
- Splines can be cached to speed up rapid destination transitions
- Steering forces are more useful for dynamic pathing, which is not necessary due to the static nature of the navigation mesh and actors not reacting to anything other than sounds
- Splines reinforce the 'mindless' nature of the actors

Detour takes an array and fills that array with a series of contiguous navigation polygons starting from the closest polygon and ending on the closest polygon that can be pathed to. This means that in cases where a resource is on the other side of a barrier, Detour will find the closest point on the available navigation mesh to the provided destination. Detour does not determine whether or not a point can be pathed to, so further logic must be applied in order to avoid actors walking through walls and other objects. *Blinding Silence* solves this problem through level design augmented with pathing error testing.

4.2.4.6 Path Smoothing

As Detour returns a list of navigational polygons, *Blinding Silence* incorporates path generation and smoothing in order to provide actors with a path to follow. Paths generated by Detour may not be optimal, so *Blinding Silence* uses path smoothing to make paths look more natural.

In Figure 37, you can see the original path before path smoothing.

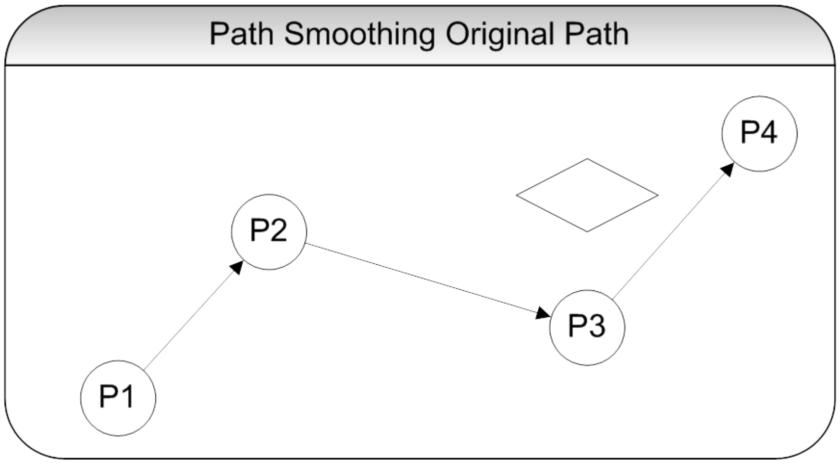


Figure 37: Unsmoothed path

The first step is to cast a ray from P1 to P3. If the ray does not collide with something, as in Figure 38, we know P2 was not necessary for navigation

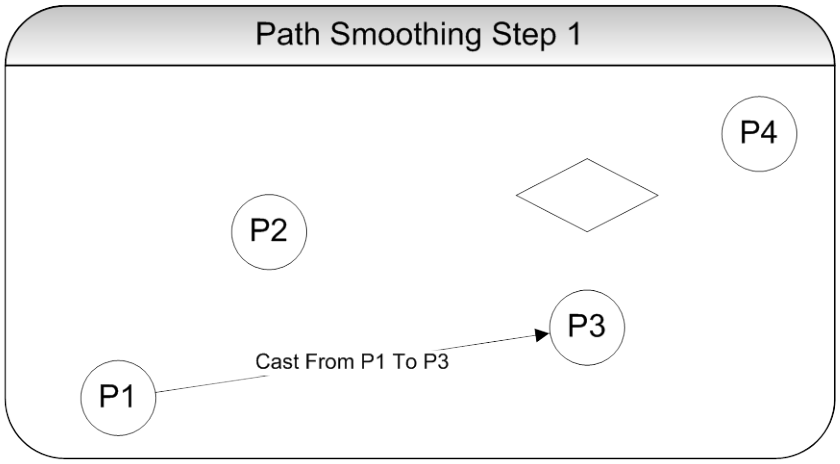


Figure 38: Path smoothing ray casting

There was no collision when we cast the ray. This means that P2 is unnecessary and we can remove it from the path, as shown in Figure 39.

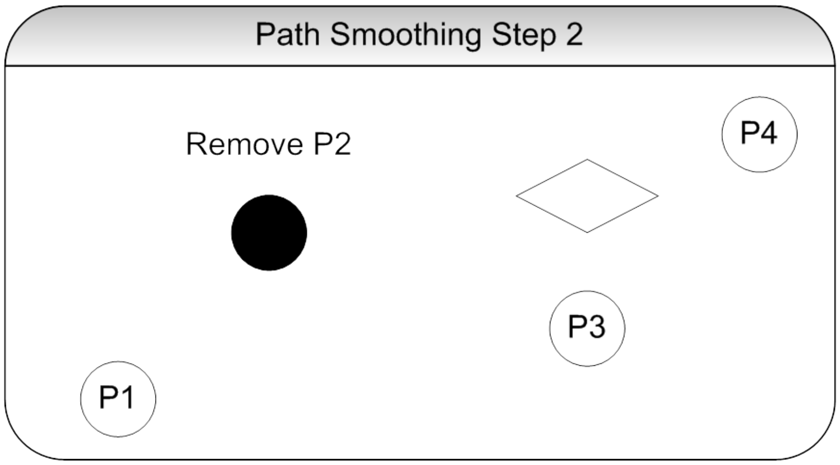


Figure 39: Removing a point

We now repeat the first step, except this time we cast the ray to P4. In Figure 40, this results in a collision.

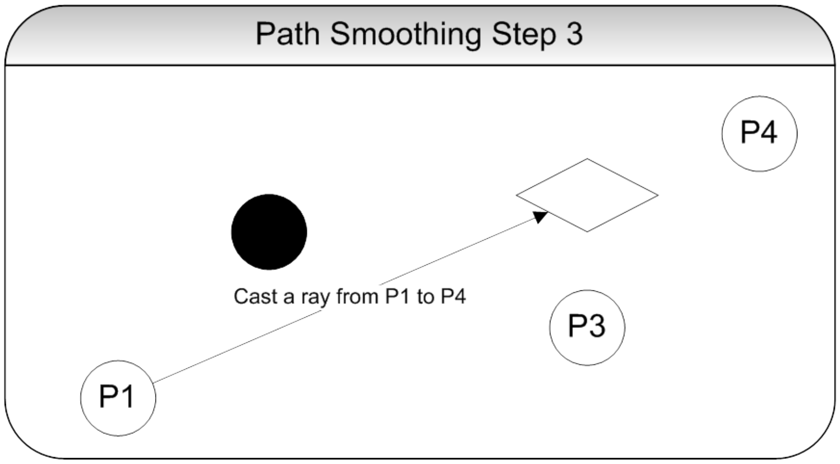


Figure 40: Ray cast with colliding object

We collided with an object when ray casting to P4, so the final path is P1, P3, P4. Figure 41 shows the final path returned.

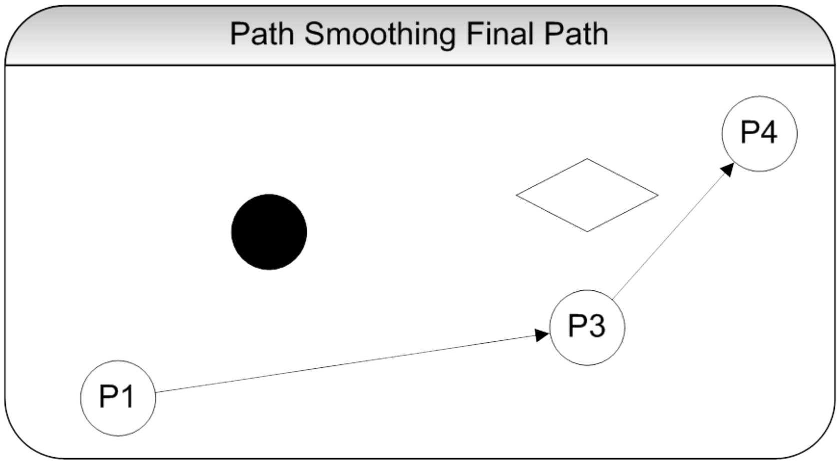


Figure 41: Smoothed path

5. Project development

This section describes the process of developing *Blinding Silence* from term A term 2009 to D term 2010.

5.1 Design

Before the first term of the project, development on the *Blinding Silence* MQP was divided into three sections. As *Blinding Silence* was originally planned to be developed over three terms, each section directly corresponded to a single term. A term was reserved for designing the game, B term for development, and C term for game testing and refinement.

A term progressed as planned. By the end of the term we produced a design document that fully described our game, which can be found in Appendix G. However at the end of A term the team revisited the design and concluded it was too complicated. As a result of this, the soul stealing mechanic was removed from the game, and a few levels were drawn up to mirror this.

5.2 Development

Over B term development was constantly one week behind schedule. Although the project planning had been set up to allow for this, winter break came on with similar results. When C term started, the game was still not at a playable state due to several critical bugs.

The entirety of C term was taken over by development. While the game was always about a week away from being playable (and therefore testable), non-critical improvements to the game both pushed back addressing present critical bugs and created new ones as fast as the old critical bugs were addressed. Near the end of C term it was clear that the game would need further development before it would be feature complete, let alone playable. In response to this we extended development into D term.

In order to extend into D term, the team had to reserve only 1/6th credit to avoid conflict with other classes. This caused the first few weeks of D term to progress with only minimal work spent on *Blinding Silence*. However, the team got together over one weekend and brought the game to a playable state, including running a few rounds of ad-hoc testing with available test volunteers. The results of these tests indicated that not only was *Blinding Silence* fun and intuitive as the original document had focused on, but the extra term spent developing had paid off in quality of the final product.

5.3 Conclusion

Although *Blinding Silence* did not achieve feature freeze until a few weeks before the end of D term, the game ended up being quite well received both on project presentation day and with testers. The original design was overly ambitious, but taking more time to develop into a product close to the original design has proven to be successful, and if the project had been cut down in scope to stick to the development timeline, the final product might not have been nearly so well received.

Though parts of the original design were cut, namely the Wii MotionPlus and animatics, their absence does not detract from the game as a whole. Instead removing these potential bottlenecks when we did allowed us to push through and create a solid game. Even without them, the development of *Blinding Silence* required the creation and integration of many complex systems that the team had never worked with before. Even without the delays, it is apparent that the proposed design could not have been easily completed in just two terms of development.

In the end, the team met the majority of the goals that were laid out. We created a fun and interesting game that challenges players to think outside of their normal playing habits. Everyone who has tried *Blinding Silence* has enjoyed it, which means that the overall development was, in fact, a success.

Appendix A: Original story

In the beginning, Olome formed from nothing. It formed in such a way that the great light was always shining on the northern hemisphere, and never on the south. It was here that the First Ones came upon the world, and took it as their own. They set the world spinning so that for 4 hours a point on the northern hemisphere was dark, and for 4 hours the south was light. They then build their homes in the north, and lived in peace with the world.

Over time, however, the First Ones lost their knowledge. They began discovering things once forgotten, and expanding, losing their touch with the land. The people became greedy, and were different. It was into this world that Jabin was born.

Jabin was always a loner, never quite getting along with others. He has the double curse of being blind and extremely intelligent, which may seem a gift to offset a disadvantage, but Jabin's peers lacked empathy for these sorts of things. The child-raising traditions of Jabin's hometown placed him unsupervised with many other youngsters, who quickly decided his seeming arrogance and thorough unwillingness to play physical games made him an uninteresting companion, and the attitude prevailed into adulthood. Jabin grew up apart from society, and when the time came to choose his life as a grown man, he left the town behind him and entered the forest as a hermit. Although such a choice was rare, it was not unheard of, and none could convince Jabin to stay in relative safety.

It was in the forest Jabin found the staff. He had lived a few years in seclusion, learned sounds to avoid, smells to approach. He had never before heard the high-pitched buzz that called to him that day, and cautiously investigated. Curiously, although he could easily tell the direction from which the sound was coming, it never seemed to get louder. Jabin nearly fell when he bumped into a three-foot staff sticking straight out of the ground. Reaching out to the source of the sound, Jabin discovered the top of the staff had some sort of cut precious gem seemingly growing out of it. Jabin hardly registered that the ringing sound had ceased, due largely to the fact that a voice was speaking to him.

Jabin could not tell a direction the voice was coming from—the voice seemed to come from inside his mind. It announced itself as a remnant of the First Ones, and informed him that difficult times were approaching. Jabin quickly realized the staff was speaking to him, and over time he learned to trust the voice.

During the next few weeks Jabin noticed a change overcome him. He was hearing sounds from farther away than he used to, and slowly he was recognizing exactly what was making the sounds. Within a few days of this discovery Jabin realized he could tell the surroundings of the sound as well. It was that night the darkness came.

Jabin wasn't sure how he knew it was darkness, but it was undeniable. A warm clammy calm swept through the forest, sounding quite the opposite of a breeze of wind. Rather than rustling the leaves of the trees, total silence swept across the woods, moving from south to north and dampening everything in its path. When the silence reached him, Jabin felt the air turn warm, but his body turned cold and began shivering. An almost painful burst of heat shot up the arm holding the staff, and shot down his veins to the rest of his body. Jabin had a moment to attempt to process what had happened, before he emptied the contents of his stomach onto the ground.

Once he had his breathing back under control, Jabin realized the darkness was sweeping north, toward his hometown. Wasting no time, Jabin set off to make sure his family was all right. Tapping the staff in front of himself like a cane, he lit up the darkness and wandered into the night.

Jabin was unprepared for what he discovered in his hometown. Although he knew it should be daytime, he felt no sun and could smell no rain. Although he was blind, he knew he would find no people around the moment he stepped onto the main road through town. There was no noise coming from the town at all, and he could locate no bodies. What he did find was the effect of many feet traveling in the direction of the nearby city. Having no real choice, Jabin stole some abandoned food and followed the trail.

The trail took him into the forest, away from the area he had called home for the past few years. The thought of traveling somewhere completely new was daunting, but his thoughts were lightened when he heard a sound coming from down the path. He rushed to meet it, excited for what he might find.

Appendix B: Zimri background and script

Zimri is a sentient stave. He originally was a crewmember of the Tenacious, a space ship transporting a human colony through space. Unfortunately, a technical malfunction caused the crew to end their journey in an inhospitable system. In order to create a planet capable of supporting life, the crew smashed their ship into a relatively small planet.

The ship was organic, a living creature capable of travelling the stars. It fused with the planet, using the same technology used for artificial gravity to preserve the lives of the passengers. The passengers were then deposited on the planet, along with some of the crew. There they made their homes.

Some crewmembers remained to watch over and assist the ship in its transformation of the planet. Zimri was one of these. Near the end of his life he fused his consciousness with that of a tree near the edge of the southern continent, watching over the shores. Over time he developed his relationship with the planet, and grew minor psychic abilities. When the descendents of the passengers returned, Zimri was used for building and firewood. He saved a single branch, being carried by a returning craftsman to the northern continent where the people lived.

He was left behind and waited for the right person to come by. Through his connection with the planet he knew an evil was coming. Luckily, he found an appropriate companion before it was too late. The story he tells is as the humans know it, many generations later.

Animatic script:

In the beginning time, Olome was born of the darkness and light. The first ones learned to live in balance with the two forces, and the world settled into equilibrium.

On the light side, the first ones of Olome grew and advanced, but were careful never to disturb the Balance.

On the dark side the forests grew dense, and there was only silence.

None who ventured into the forests ever returned the same.
As generations passed, the knowledge of the first ones faded, and the Balance was forgotten.

Civilization on Olome began to expand. Humans consumed their resources at an unsustainable rate. Economic collapse seemed inevitable, until a breakthrough was made:

A new type of crystal was found, and scientists learned that it could produce unimaginable levels of power.

Soon the cities flourished once again, and the human expansion continued. However, even this new resource was finite, and was soon nearly depleted. The human ‘solution’ to this problem caused the darkness that now envelops you.

You, Jabin—a man without the power of sight—were still able to see the corruption of your civilization. Turning to the forests, you left humanity behind.

That is when our paths crossed.

I am a remnant of the first ones. When we met, I was not strong enough to talk to you, but the actions of those you left have changed that.

In the dark forests, they discovered a new source of crystals, even stronger than the last, and they began to use these crystals throughout their domain.

By doing so, they have re-introduced the darkness to the light; the equilibrium is broken. Now the darkness has spread over all of Olome, and the world you knew has ground to a halt.

We must venture through the forest and reach the city you left. There we can find the source of the darkness, and attempt to restore the Balance.

You must do this, or Olome will wither and die in darkness.

Appendix C: Asset Lists

These are the final asset lists for the game, based on the lists made in the initial design phase. The status column lists assets as follows: A “C” indicates that the asset was planned in the design phase and has been completed and included in the final game; an “X” indicates that the asset was planned but was cut from the game for some reason and never completed; an “N” indicates that the asset was not originally planned but has since been added based on ongoing development needs.

Visual Art

	Description	Status
Human		
Base Mesh	Base human mesh for all human characters. Includes arm stump for attachments	C
One-socket mesh	Mesh for actors with attachments on one arm only (operator, harvester, craftsman)	C
Two-socket mesh	Mesh for actors with attachment points on both arms (carrier)	C
Biped rig	Generic biped rig	C
Animation		
Chop tree	Harvester chopping a tree	C
Mine stone	Harvester mining	C
Pickup wood	Carrier grabbing wood	C
Drop wood at pile	Carrier dropping wood at a pile	C
Drop wood anywhere	Carrier dropping wood and switching to stone	C
Pickup stone	Carrier grabbing stone	C
Drop stone at pile	Carrier dropping stone at a pile	C
Drop stone anywhere	Carrier dropping stone and switching to wood	C
Hammer wood	Craftsman working on wood	C
Chisel Stone	Craftsman working on stone	C
Walk	Basic walk	C
Craftsman Chisel Walk	Basic walk, but with a hand in a grasping position to hold the chisel	C

Walk with wood	Carrier walking with wood	C
Walk with stone	Carrier walking with stone	C
Pull lever	Operator using lever-arm	C
Idle 1	Basic idle	X
Idle 2	Basic idle with foot tapping	X
Harvester Change prop	Pass hand behind back, and bring it to the front again with a new prop	C
Craftsman Change prop (x2)	As with the harvester, but the craftsman needs to swap in and out of holding a chisel for stone work	C
Props	▨	▨
Cane	Player's cane	C
Axe	Harvester's tool for trees	C
Pick-axe	Harvester's tool for stone mining	C
Claw	Carrier's tool for carrying wood (2 hooks)	C
Grabber	Carrier's tool for carrying stone (4 hooks)	C
Chisel	Craftsman's tool for stone-work	C
Mallet	Craftsman's tool for stone-work	C
Hammer	Craftsman's tool for wood-work	C
Lever	Operator's prop for activating things	C
Clothing	▨	▨
Hard hat	Head-gear for all workers	C
Hard hat with glasses	Special hard hat for operator	C
Beard	Large, exaggerated beard for harvester	C
Safety goggles/eye attachment	Craftsman's eye-gear	C
Tool belt	Craftsman's tool belt	X
Levels	▨	▨
Forest Content	▨	▨
Lush trees	Tall, healthy trees with leaves (2 variations	X
Dead trees	Short, dead trees, skinny (2 variations)	C

Stumps	Stumps; not harvestable (2 variations)	C
Boulders	Very large rocks mined by Harvesters	C
Rocks	Small rocks scattered around the landscape	X
Small crystals	Small sized crystals at the end puzzles	C
Large crystals	Larger crystals at the end of puzzles	C
Resources		
Log	Logs carried by Carriers	C
Log pile	Wood pickup points for Carriers	C
Log pallet	Wood dropoff point for Carriers	C
Stone	Stones carried by Carriers	C
Stone pile	Stone pickup points for Carriers	C
Stone hopper	Stone dropoff point for carriers	C
Industrial Structures		
Wood fences	Fences blocking the player, but that the player can see through	C
Stone fences	Fences blocking the player and that he cannot see through	C
Metal fences	Fences blocking interactors from leaving a puzzle are. No effect on the player and cen be seen past	C
Force field doors	Block players but can be turned off - effect	C
Electric wire support system	Support towers/telephone poles for wire routing between crystals and the things they power	C, but not used in game
Wood switch	Switch made of wood (with animation)	C
Stone switch	Switch made of stone (with animation)	C
Wood structure	Simple wood structures under construction by Craftsmen (3 variations) piece of fence	C
Cut stone	Partially shaped block of stone being worked on by Craftsmen	C
Bridge	Bridges over rivers, potentially blocked by forcefields. 3 variations: wood, stone and metal	X
Mine structure	Structure, slightly embdded in the environment, holding the boulders to be mined	X

UI	▨	▨
Main menu	The main menu screen	C
Choose level	The level selection screen	C
Options	The options screen	C
Graphics	The graphics settings screen	C
Sounds	The sound settings screen	X
Shaders	▨	▨
Character shaders	Shaders, tweaked for color	C
Environment shader	Specific shader to help environment visibility and interest	C
River shader	Specific shader for rivers, which constantly emit sound	X
Crystal Shader	Shader that pluses the ambient visibility of the crystal	N
Animatics	▨	▨
Intro	Tells the background story of the world	X
Credits	Simple credits screen	C
Images	▨	▨
Instructions	Imagery to teach the player about the world and how to play	N
Completion	Screens shown at completion of a level or the game	N
Logo	Looping movie on the main screen of the game’s logo	N

Audio

Asset	Description	Status
Sound Effects		
Chop wood	Hit wood with axe	C
Hit wood	Hit wood with cane	C
Hammer wood	Hit wood with hammer	C
Mine stone	Hit stone with pickaxe	C
Hit stone	Hit stone with cane	C
Chisel stone	Work on stone with chisel	C
Hit crystal	Hit crystal with cane	C
Hit metal	Hit metal with cane	C
Hit leaves	Hit leaves with cane	X
Hit ground	Hit ground with cane	C
Footstep, ground	Walk on ground	C
Footstep, wood	Walk on wood	X
Footstep, stone	Walk on stone	X
Footstep, metal	Walk on metal	X
Footstep, ground, loud	Stomp on ground	C
Footstep, wood, loud	Stomp on wood	X
Footstep, stone, loud	Stomp on stone	X
Footstep, metal, loud	Stomp on metal	X
Water splash	Object dropped into river	X
River ambient	River flowing	X
Pull lever	Gears clicking	C
Lever sound, wood	Wood sliding and colliding	C
Lever sound, stone	Stone sliding and colliding	C
Force field ambient	Machine humming	C
Force field turn on	Ascend flange slide	C
Force field turn off	Descend flange slide	C
Deaf noise	High pitch ringing	X

Crystal smash	Shattering crystal	C
Dialogue	▨	▨
Intro	Animatic narration	C, but not used
Other	Staff dialogue	X

Marker Types

This table lists the various markers used throughout the game, and their associated IDs.

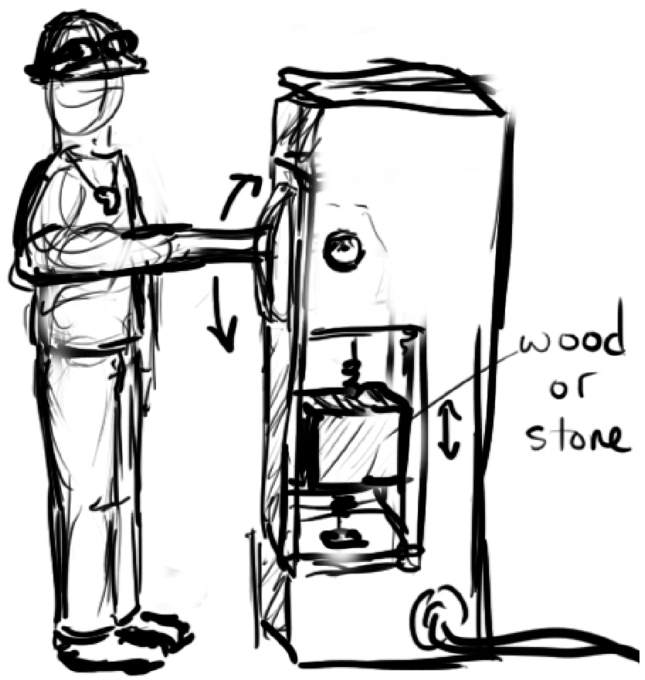
Asset	Marker Type	Marker ID
Characters		
Carrier	Model	carr
Craftsman	Model	crft
Harvester	Model	harv
Lever Puller	Model	optr
Level Assets		
crystal_large	Reference	0000
crystal_small	Reference	0001
stone_wall	Reference	0002
wood_fence	Reference	0003
force_field	Model	door
stone lever	Reference	0005
log	Reference	0006
log_dropoff	Reference	0007
log_pickup	Reference	0008
power_pole	Reference	0009
scaffold	Reference	0010
boulder	Reference	0011
cut_stone	Reference	0012
stone	Reference	0013
stone_dropoff	Reference	0014
stone_pickup	Reference	0015
stump	Reference	0016
stump_big	Reference	0017
tree	Reference	0018
tree_big	Reference	0019
wood_fence_open	Reference	0020

stone_wall_open	Reference	0021
force_field_emitter	Model	ffem
metal_fence	Reference	0023
wood lever	Reference	0024
Resources	▨	▨
Interaction Location	Locator	iloc
Item Locator	Locator	itml
Socket Locator	Locator	socl
Stone Locator	Locator	stnl
Log Locator	Locator	logl
Spawn Locator	Locator	spwn
Right Foot Locator	Locator	righ
Left Foot Locator	Locator	left

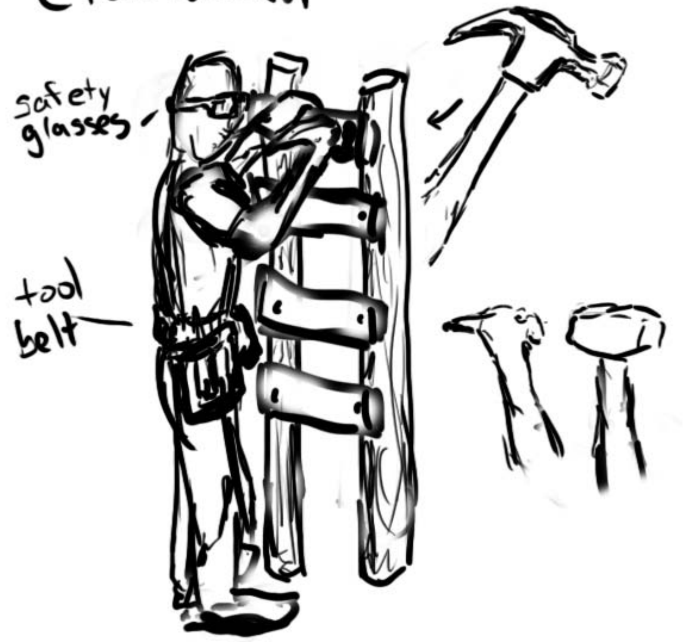
Appendix D: Reference and Concept Art

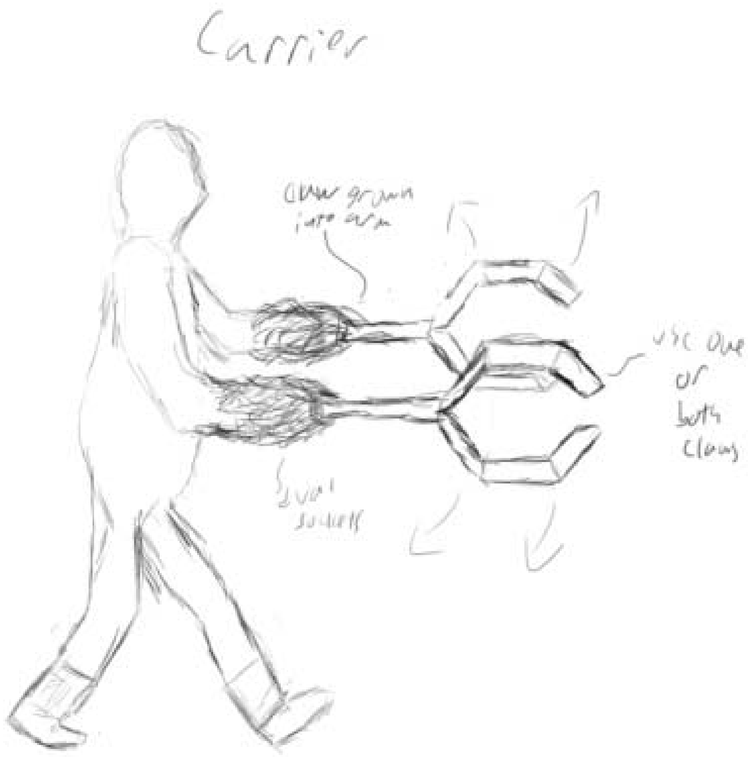
Characters

Operator

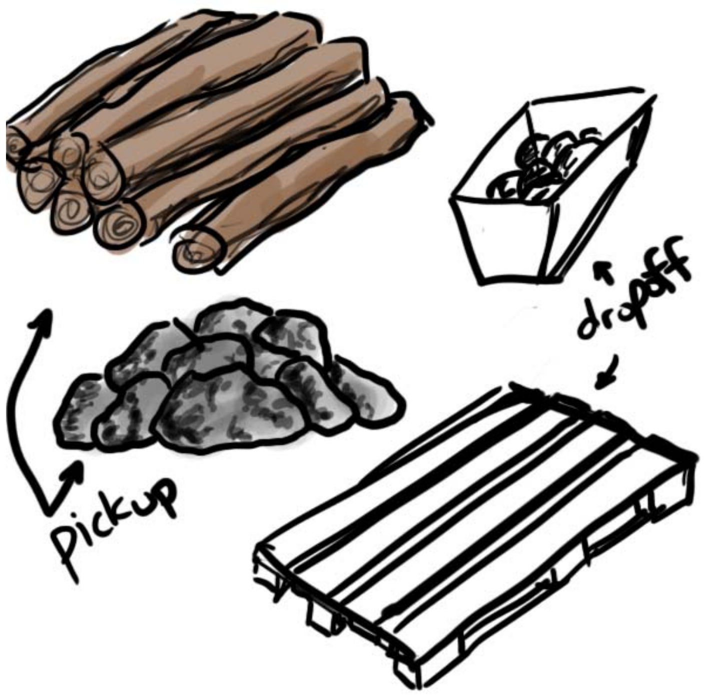


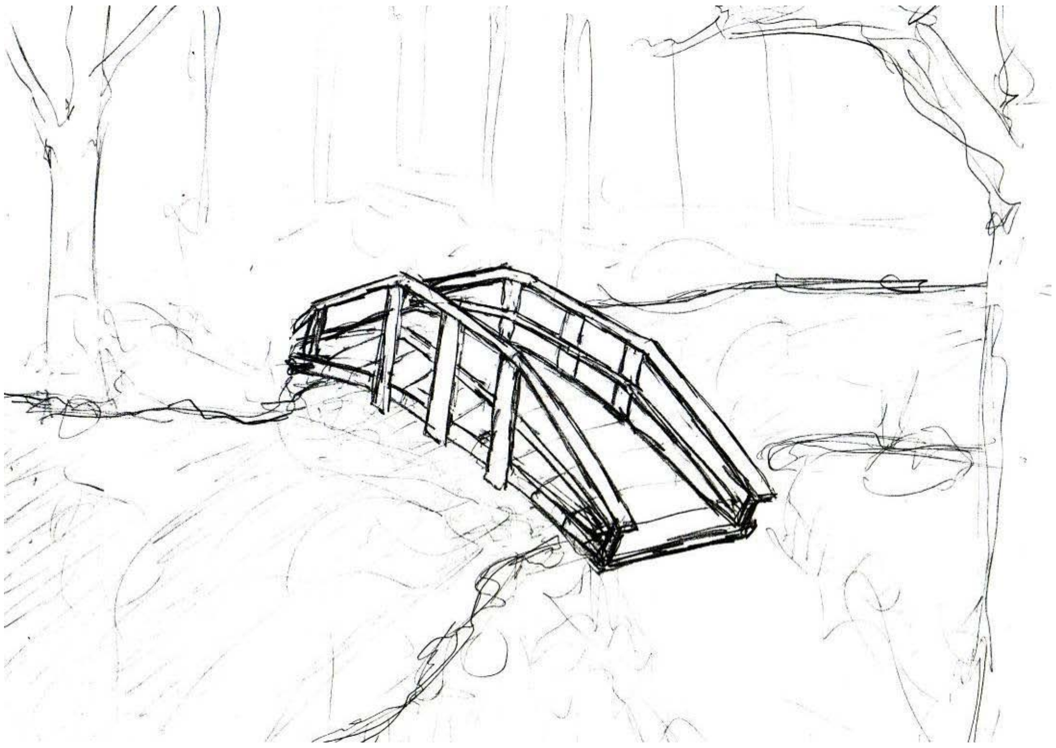
Craftsman




Carrier
claw grown into arm
use one or both claws
dual sockets

Forest


dropoff
pickup



http://media.photobucket.com/image/forest/alexv888/dark_forest.jpg

http://fc03.deviantart.com/fs19/f/2007/238/8/f/Forest_1_by_stock_feele.jpg

http://upload.wikimedia.org/wikipedia/commons/6/6a/The_lumberjack_was_here.jpg

http://upload.wikimedia.org/wikipedia/commons/c/c4/MUWO4193.JPG

http://upload.wikimedia.org/wikipedia/commons/6/62/Contrasting_Tree_Types_Coexist_in_a_Forest.jpg

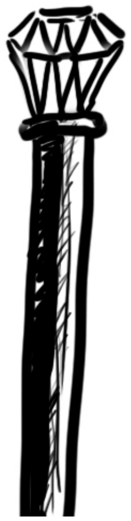
http://www.timhaufphotography.com/gallery/Panoramas/pan0508/SouthAfricaMkuzeGiantFigTreeForest243_lg.jpg

http://upload.wikimedia.org/wikipedia/commons/d/db/Zrywka_drewna_776.jpg

http://pearl7.files.wordpress.com/2008/11/tree-stump.jpg

http://www.geocities.com/tyedye776967/river_log.jpg

Cane



http://www.swords24.eu/images/products/en/Cold_Steel_African_Walking_Stick.jpg

Props

http://www.wealddown.co.uk/images%20shop/434-1USAFellingAxe81cm.JPG

http://upload.wikimedia.org/wikipedia/commons/3/36/Chisel_wood_24mm.jpg

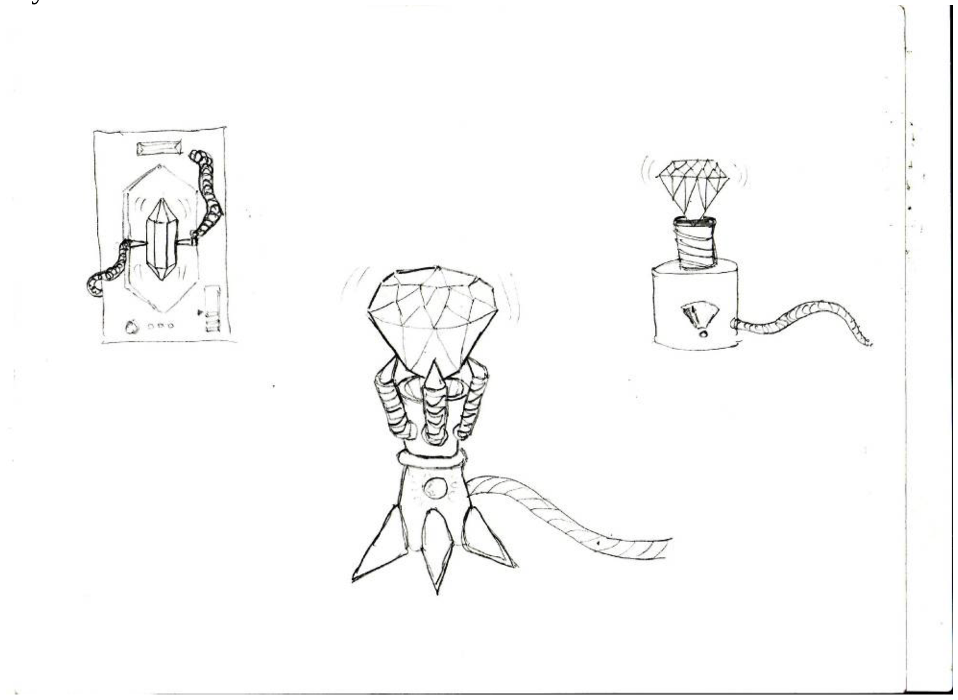
Clothing

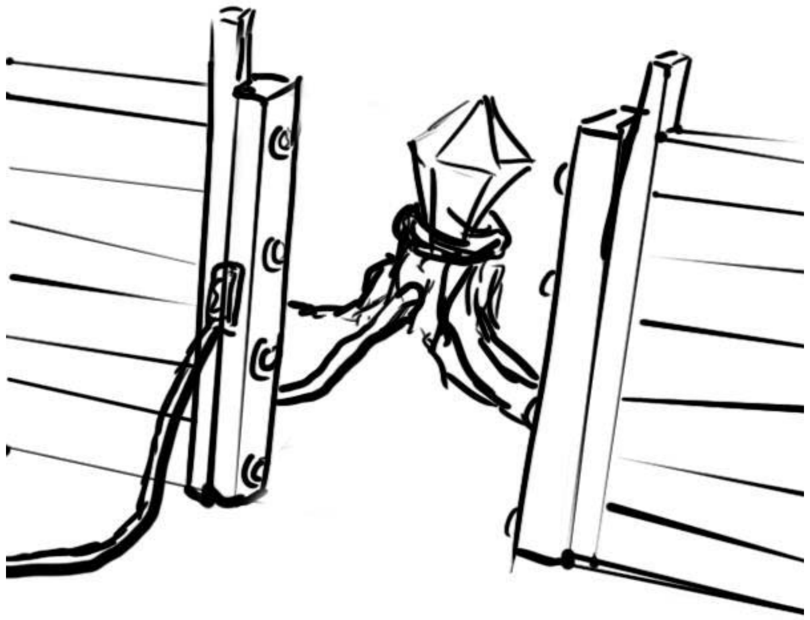
http://upload.wikimedia.org/wikipedia/commons/c/c9/Chainsaw_helmet.jpg

http://www.rd.com/images/tfhimport/2000/Jun00_Using_Tools/20000601_Using_Tools_page001img001_size2.jpg

http://www.singlesourcesupply.com/_wizardimages/Hard%20Hat%20John%20Deere.jpg

Crystals





Appendix E: Introductory Animatic Script

Blinding Silence Introductory Animatic

Open to a black screen with the words '*Blinding Silence*' in white block text, which then disolve to match the pointelized style of the animatic. Shortly after, the text disperses into blobs of white, undulating and floating in the sea of blacks and greys. As the dialog proceeds, the blobs coalesce into a planet that is covered in undulating dark and light blobs.

(2) In the first time, Olome was born of the darkness and light. Over time, the first ones learned to live in balance with the two forces,

The blobs seperate, with the dark one moving to the bottom of the planet and the light ones to the top.

(3)and the world settled into equilibrium.

Additive fade into and untouched grassy knoll in 4a. Then, in 4b, one or more people dissolve onto the same knoll farming. They are lightly detailed, implying that they are still pure and embracing the ballance and the light side they live on. Finally, in 5, basic structures have begun to appear in the background, and farming has progessed into a man chopping at tree. Then fades to black.

(4a) On the light side, **(4b)**the people of Olome grew and advanced, **(5)**but were careful never to disturb the Balance.**(Black)**

Fade up to a dense forest of inumerable trees, stretching out into the distance and fading into darkness. Ove the course of the text, the trees grow taller upward, and vines creep over them. The image is very dark and detailed, and should be forboding.

(6) On the dark side, the forests grew dense, and there was only silence.

Fade to an opening in the trees, filled with abosolute darkness that is pulsing out to the surounding trees. The image slowly zooms into the opening as the text proceeds, and should feel even more forboding than the last image. Then fade to black.

(7) None who ventured in, ever returned the same.**(8/Black)**

Switch back to the image of the planet. The undulating light on the top is starting to fade and darken. Small cities begin to appear on the light side.

(9) As the years passed, the balance was forgotten.

Fade to an image of a small glade of trees, that change one by one into stumps as a pile of logs grows in the front. Buildings and infrastructure begin to appear in the background and the land starts to look desolate.

(10) Civilization on Olome began to expand and consumed their recources at an unsustanable rate. Economic collapse seemed inevitable, until a breakthrough was made:

Fade to a large crystal on black background. Very clean crystal with little detail, not corrupted. In 11b, the crystal starts to pulse energy, and maybe fades into a basic version of the structure used to sap power in the game.

(11a)A new type of crystal was found, **(11b)**and scientists learned that it could produce unimaginable levels of power.

Fade back to the view of the planet. Cities continue to expand on the light side, and the light continues to darken. Darkness from the bottom starts to seep upward. Fade to black.

(12) Soon, their cities flourished once again, and their expansion continued. But even this resource was finite, and was soon nearly depleted.**(Black)**

Close in on a back/side view of a non-descript head, suggested to be Jabin. In 14, zoom out to show the figure (Jabin) looking back down a path at the city he has just left. It is surrounded by a wall and a very desolate landscape.
(13) You, Jabin, a man without the power of sight, **(14)**were still able to see the corruption of your civilization, and left it behind you.**(Black)**

Fade to black, then:

That is when our paths crossed.

Fade up to a picture of the staff, slowly zooming in.

(15)I am a remnant of the first ones. When we met, I was not strong enough to talk to you, but the actions of those you left have changed that.

Switch back to the earlier view of the entrance to the dark forest (from storyboard 7), and slowly zoom toward the darkness. In 17, change to a view inside the forest of crystals on the ground. These are corrupt, and much darker/more detailed than the earlier one.

(16) Deep in the dark forests, they discovered a **(17)**new source of crystals, even stronger than the last, and they began to use these crystals throughout their domain.

Back to the image of the planet. The cities have grown to be huge, and the darkness moves upward from the bottom to start engulfing the entire planet, though doesn't finish in this view. Then fade to black.

(18) By doing so, they have re-introduced the darkness to the light, and have broken the equilibrium. Now the darkness has spread over all of Olome, and the world you knew has ground to a halt.**(Black)**

Fade up to a view of a forest from the light side. It is not as corrupt as the one from the dark side, but it has been engulfed in the darkness. In the distance is a faintly visible/pulsing city. Zoom slowly toward the city.

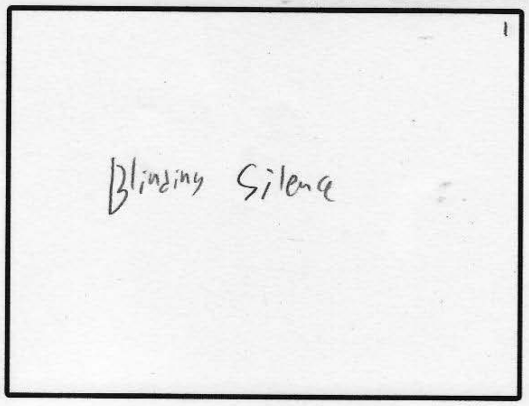
(19) We must venture through the forest, and reach the city you left. There we can find the source of the darkness, and attempt to restore the Balance.

Back to the view of the planet. The darkness completes its spread and engulfs the entire planet. Fade to black.

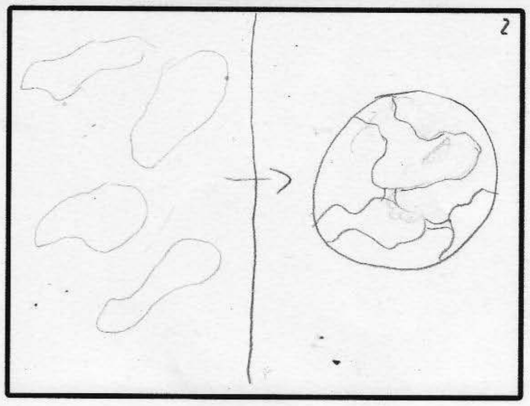
(20) You must do this, or Olome will wither and die in the darkness.**(Black)**

Appendix F: Introductory Animatic Storyboards

Title Blinding Silence Intro animatic **Date** ________ **Page** 1

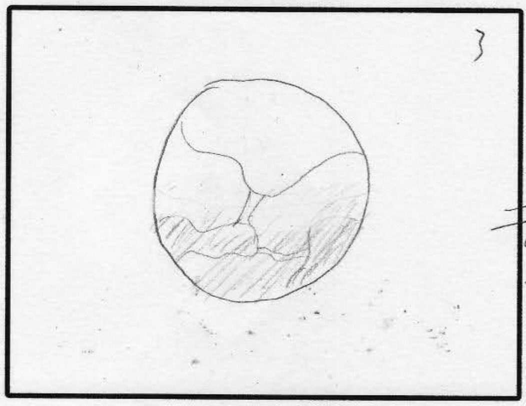


fade in title as block text, then dissolve into stylized text that matches other images
↓
Then fade to black in a glow pulse
(opening)

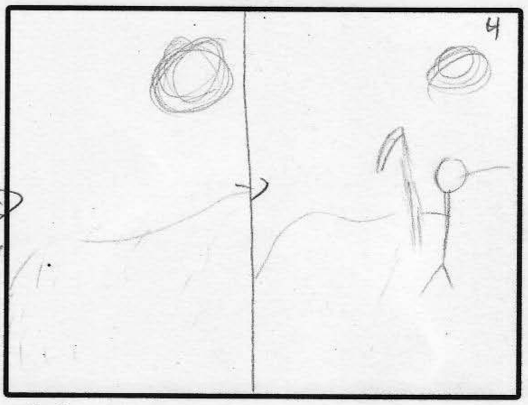


Non-descript blobs of light, undulating in a sea of blacks and greys. eventually coalesces into a planet that is covered by undulating ~~dark~~ and light areas

("In the first time)



additive fade



blobs of light move to top dark to bottom

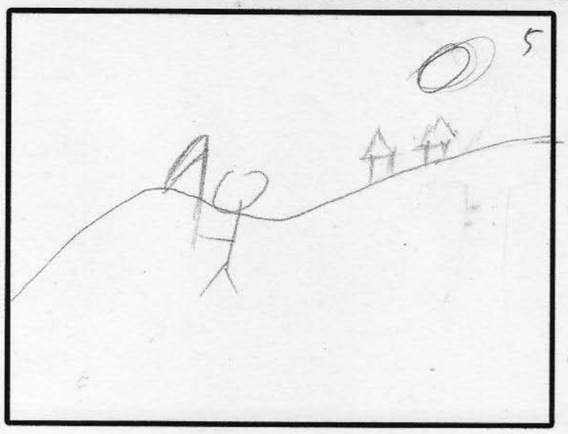
("and the world settles into equilibrium")

- grassy knoll, untouched
- Same, but with a person (or people) farming

very light → little detail (its the pure/light side)

("On the light side")
↓

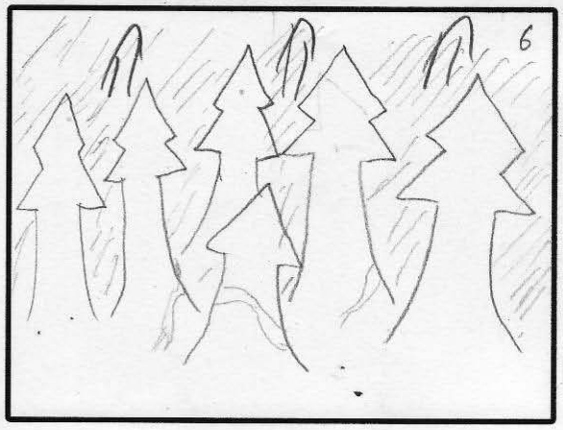
Title Blinding silence Intro animatic **Date**______ **Page** 2



-Same as 4 but now basic structures are appearing and another man chops a tree

->fade to black

("but wee careful never to")



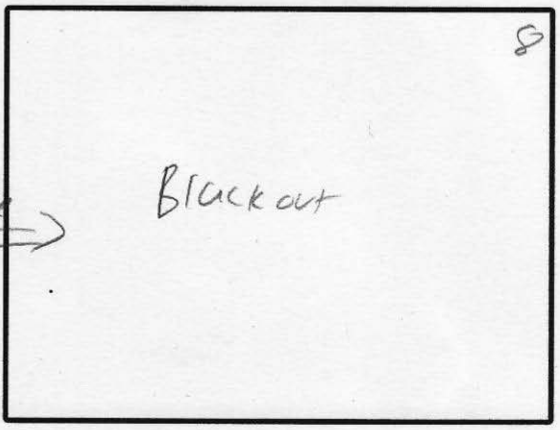
- forest, lots of trees stretching in to darkness
- vines on trees
- trees grow upward/denser, more vines grow and get longer
- very dark, stippled, and heavy detail
- forboding

(on the dark side?)

7
fade
ZOOM in

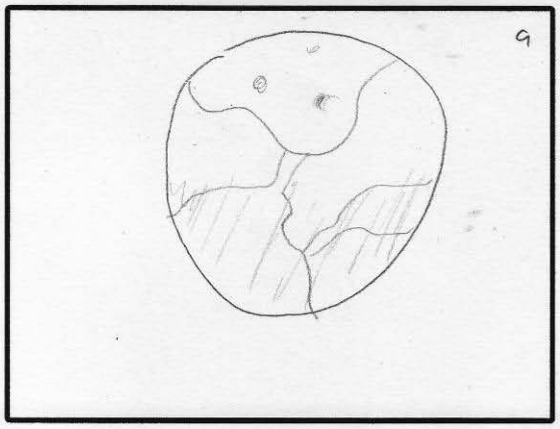
- an opening in the trees is shown with absolute darkness between them, and bleeding out/pulsing -> not somewhere you want to go

("None who ventured in")



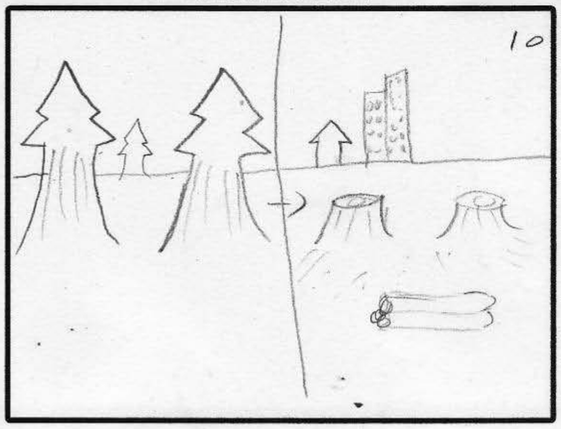
end segment

Title Blinding Silence Intro Animatic **Date**______ **Page** 3



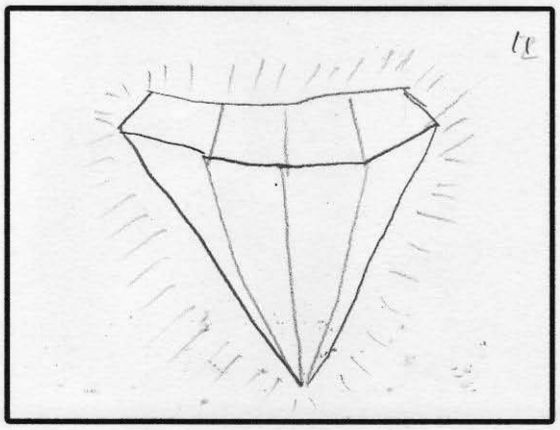
- undulating light on the top starts to fade and it starts to darken
- cities start to appear

("as the years")

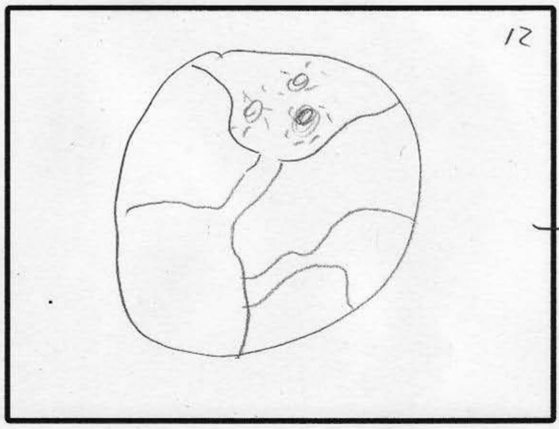


Trees start to disappear into stumps piles of wood and infrastructure appear untill it is desolate

("civilization on alone")



- fade to a crystal on black, very clean w/ little detail
- starts emanating power at point b



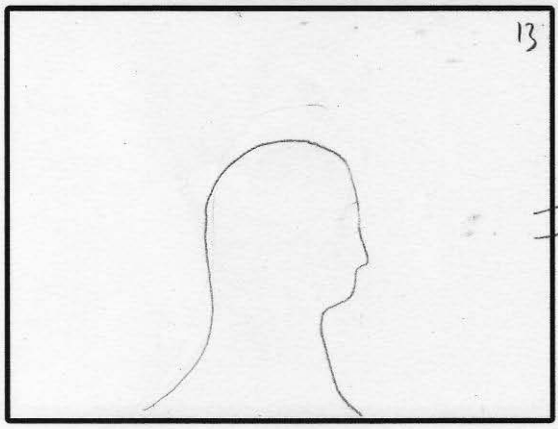
fade to → black

- cities are growing, but the light is still leaving the top side.
- darkness starts to bleed up

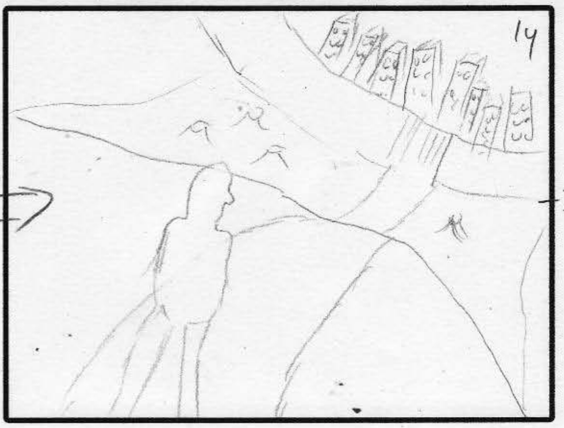
("Soon their cities")

a. ("a new type")

Title Blinding Silence Intro animatic **Date**_______ **Page** 4



Zoom out



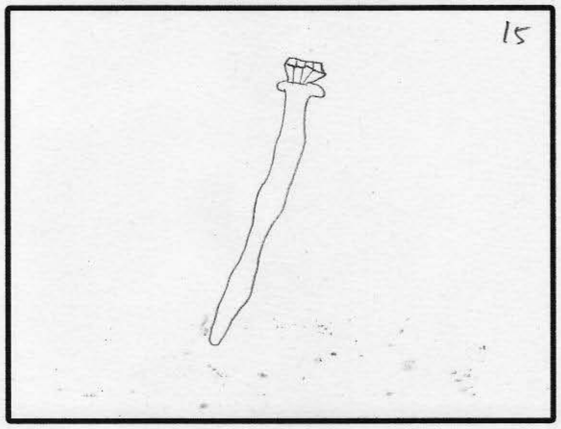
fade out

- Close on Jabin's head
- back/side view, so can't make much out about him
- he is less detailed than what

("You, Jabin")

Zoom out to show him looking back at his city, surrounded by a wall, and with desolate landscape around it

("were still able to see")



16

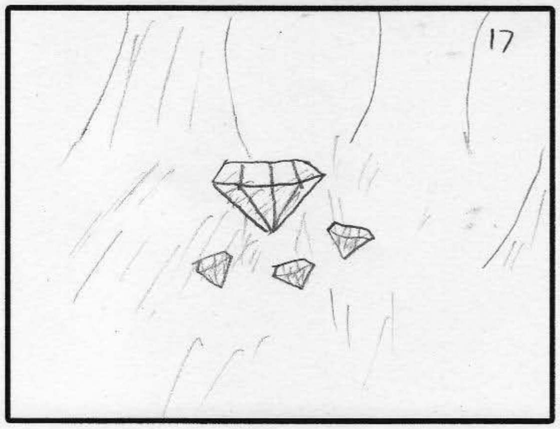
Same as 7

Picture of the staff, slowly zooming in

("I am a remnant")

cut back to 7, zooming into the darkness

Title Blinding Silence Intro Animatic **Date** ______ **Page** 5



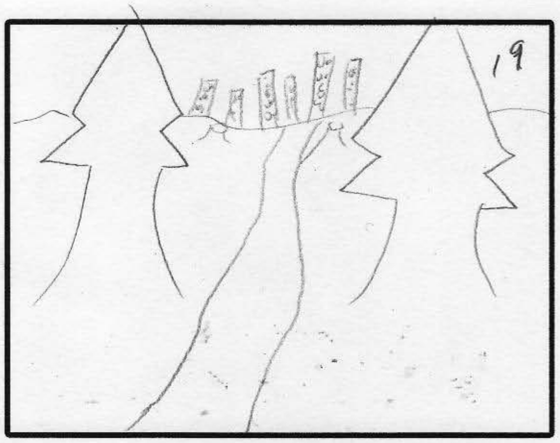
- crystals on the forest floor, with heavy stippled detail, moving detail, etc
- dark

("new source of crystals")

fade out →

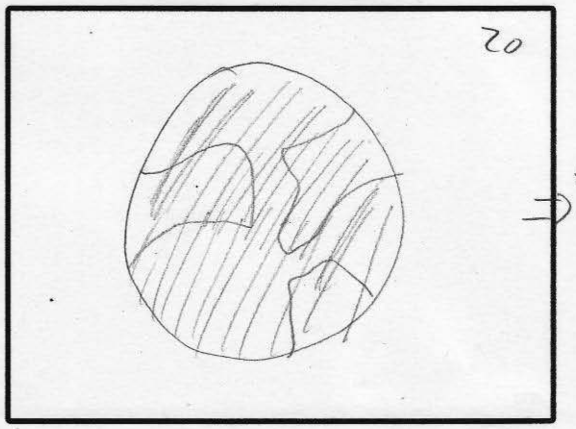
- Cities are even larger, but the darkness moves upward and engulfs the planet
- darkness expands onto the cities too

("By doing so")



- darkened forest, leading to the city far in the distance
- Zooming into the city, slowly

("We must venture")



fade out ⇒

_The darkness spreads to encompass the whole planet, and shrouds the image in black

("You must do this")

Appendix G: Original Game Design Document

The design of *Blinding Silence* changed greatly during development. This appendix contains the design document as it was written at the end of A term 2009, the first term of the project. The largest changes were the removal of the soul stealing mechanic and the simplification of level elements.

2. Gameplay Design

Blinding Silence consists mainly of physical puzzles that rely heavily on the concept of seeing sound. The player will discover the game's story as puzzles are completed.

2.1 Story

Blinding Silence takes place on a fictional planet. Although many things are familiar, there is a constant theme of unbridled capitalism that runs throughout the world.

2.1.1 Backstory

In the beginning, Olome formed from nothing. It formed in such a way that the great light was always shining on the northern hemisphere, and never on the south. It was here that the First Ones came upon the world, and took it as their own. They set the world spinning so that for 4 hours a point on the northern hemisphere was dark, and for 4 hours the south was light. They then build their homes in the north, and lived in peace with the world.

Over time, however, the First Ones lost their knowledge. They began discovering things once forgotten, and expanding, losing their touch with the land. The people became greedy, and were different. It was into this world that Jabin was born.

Jabin was always a loner, never quite getting along with others. He has the double curse of being blind and extremely intelligent, which may seem a gift to offset a disadvantage, but Jabin's peers lacked empathy for these sorts of things. The child-raising traditions of Jabin's hometown placed him unsupervised with many other youngsters, who quickly decided his seeming arrogance and thorough unwillingness to play physical games made him an uninteresting companion, and the attitude prevailed into adulthood. Jabin grew up apart from society, and when the time came to choose his life as a grown man, he left the town behind him and entered the forest as a hermit. Although such a choice was rare, it was not unheard of, and none could convince Jabin to stay in relative safety.

It was in the forest Jabin found the staff. He had lived a few years in seclusion, learned sounds to avoid, smells to approach. He had never before heard the high-pitched buzz that called to him that day, and cautiously investigated. Curiously, although he could easily tell the direction from which the sound was coming, it never seemed to get louder. Jabin nearly fell when he bumped into a three-foot staff sticking straight out of the ground. Reaching out to the source of the sound, Jabin discovered the top of the staff had some sort of cut precious gem seemingly growing out of it. Jabin hardly registered that the ringing sound had ceased, due largely to the fact that a voice was speaking to him.

Jabin could not tell a direction the voice was coming from—the voice seemed to come from inside his mind. It announced itself as a remnant of the First Ones, and informed him that difficult times were approaching. Jabin quickly realized the staff was speaking to him, and over time he learned to trust the voice.

During the next few weeks Jabin noticed a change overcome him. He was hearing sounds from farther away than he used to, and slowly he was recognizing exactly what was making the sounds. Within a few days of this discovery Jabin realized he could tell the surroundings of the sound as well. It was that night the darkness came.

Jabin wasn't sure how he knew it was darkness, but it was undeniable. A warm clammy calm swept through the forest, sounding quite the opposite of a breeze of wind. Rather than rustling the leaves of the trees, total silence swept across the woods, moving from south to north and dampening everything in its path. When the silence reached him, Jabin felt the air turn warm, but his body turned cold and began shivering. An almost painful burst of heat shot up the arm holding the staff, and shot down his veins to the rest of his body. Jabin had a moment to attempt to process what had happened, before he emptied the contents of his stomach onto the ground.

Once he had his breathing back under control, Jabin realized the darkness was sweeping north, toward his hometown. Wasting no time, Jabin set off to make sure his family was all right. Tapping the staff in front of himself like a cane, he lit up the darkness and wandered into the night.

Jabin was unprepared for what he discovered in his hometown. Although he knew it should be daytime, he felt no sun and could smell no rain. Although he was blind, he knew he would find no people around the moment he stepped onto the main road through town. There was no noise coming from the town at all, and he could locate no bodies. What he did find was the effect of many feet traveling in the direction of the nearby city. Having no real choice, Jabin stole some abandoned food and followed the trail.

The trail took him into the forest, away from the area he had called home for the past few years. The thought of traveling somewhere completely new was daunting, but his thoughts were lightened when he heard a sound coming from down the path. He rushed to meet it, excited for what he might find.

2.1.2 Game story arc

The player learns the game's story by completing levels. Each time the player strikes a level-ending crystal, all the Actors whose souls the player did not collect will wake up and tell the player a part of the story. The more people the player 'saves,' the more detail the player will learn about an aspect of the story. However, only a relatively conservative number of Actors must survive for the gist of the story segment to be conveyed.

As the story is told by individuals who have been stuck in the darkness, the player is hearing a story about past events.

2.2 Gameplay Overview

A player has one goal and two tasks in each level of *Blinding Silence*. The player's goal is to get to the end of the level where a mystical crystal lies. The crystal can restore light to the world and 'fix' any Actor the player did not affect. To accomplish this, the player does two things: observe the puzzle, and interact with it. Through this, the player can solve each puzzle in a number of ways.

2.2.1 Observation

Each puzzle in *Blinding Silence* is a miniature machine. Actors are going about their tasks accomplishing a goal such as carrying logs or cutting down a tree. In order to solve puzzles, the player must first get an idea of what is going on. This is not arbitrary; as the elements can get fairly complicated in their interactions, special effort will be made to stress the importance of observation to the player.

2.2.2 Interaction
Once the player has an idea of what they want to do, they can go and change the status quo. The effect the player has can rapidly snowball into altering the entire puzzle, so timing and choosing the right interaction is crucial. As there will be multiple ways to solve each puzzle, some solutions affect fewer Actors, and through that save a greater number when the crystal is reached. Players can emit sounds to influence all Actors within a certain range, or simply stop one Actor (if they can reach him). Only through interaction can the player change the puzzle in such a way as to allow progression to the crystal.

2.3 Hardware
Blinding Silence will have a variety of control devices for interaction with the character.

2.3.1 Wiimote cane
The main method of control will be a cane-like Wiimote housing, complete with nunchuck attachment. The nunchuck will control walking forward and backward, as well as turning. The cane itself will be swung about in a variety of circumstances to create sounds and otherwise interact.

2.3.2 Head-tracking glasses
To add another level of control to an otherwise limited movement scheme a pair of safety glasses with infrared LEDs will be used in tandem with another Wiimote for basic head tracking. The camera will pan and tilt to a limited degree in response to player head movement. This will give players an easy way to quickly check multiple puzzle elements.

2.4 Interactions
The player has a set of interactions he can use to affect the game world. Proper timing and intelligent utilization of these interactions is the main method of solving almost every puzzle.

2.4.1 Cane whack
By swinging the Wiimote cane, the player can hit certain surfaces and objects with the cane, causing the most common sounds the player interacts with. The player can also hit both types of crystals found in *Blinding Silence*, listed later in this section.

2.4.2 Sonar pulse
By making noises into the microphone, the player can create a burst of sound right where Jabin is standing. This causes a small amount of light with which the player can see the immediate surroundings.

4.4.3 Reset scream
By emitting a loud constant noise into the microphone for several seconds, the player can trigger a reset scream, which is a blinding burst of light that blanks out the screen and causes a high-pitched ringing sound to dominate the audio. After a second, the screen slowly clears and hearing slowly returns. The player has been moved to the beginning of the level, and all puzzle elements have been reset to where they started. Also, the screen clears faster if the player shakes their head.

2.5 Actors

Actors are the puzzle elements that interact with the environment, the player's noises, and each other. Actors react to wood and stone sounds, regardless of the sound's source. There are four actors and each respond to sounds differently, as shown in Table 1.

Table 2: Actors

Sound	Harvester	Carrier (empty)	Carrier (wood)	Carrier (stone)	Craftsman	Operator
Wood	Next tree	Nearest wood pile	Nearest wood drop	Nearest wood pile	Next stone workpoint	Pull next wood lever
Stone	Next mine	Nearest stone pile	Nearest stone pile	Nearest stone drop	Next wood workpoint	Pull next stone lever

2.5.1 Harvester
The harvester chops or picks away at a tree or stone, moving only in response to stimulus. When the harvester hears a sound, he moves to the nearest unoccupied resource or the same type as that sound.

2.5.2 Carrier
The carrier transports materials from one pile to another. The carrier switches piles based on what kind of sound he hears, and can be directed long distances without ever finding a pile to drop a resource. When not carrying anything, the carrier will go to the nearest pile of the type that matches the sound heard. When already carrying, upon hearing the same sound as the resource he is carrying he will switch to the next closest drop-off point. If the sound heard is different from what he is carrying, he will drop what he is carrying and go to the nearest pickup pile corresponding to the sound.

2.5.3 Craftsman
The craftsman has two types of working spaces that function opposite of the harvester; the craftsman moves to the opposite kind of position as the sound he hears. This allows for puzzle elements to interlock without moving each other constantly.

2.5.5 Operator
The operator pulls levers in levels, opening doors and activating other objects. The type of lever he pulls corresponds directly with the type of sound he hears.

2.6 Objects
Objects are the puzzle elements that react to player or actor interaction, but do nothing on their own.

2.6.1 Actor Objects
Actor objects are the objects that actors work on directly. As shown on Table 2, there are two types of each actor object, a stone version and a wood version. Each makes a stone or wood sound when struck by the player, and has one type of actor that interacts with it.

Table 3: Actor Objects

Object	On player hit	Interacting Actor

Tree	Wood	Harvester(wood)
Boulder	Stone	Harvester(stone)
Wood pickup	Wood	Carrier(wood)
Stone pickup	Stone	Carrier(stone)
Wood dropoff	Wood	Carrier(wood)
Stone dropoff	Stone	Carrier(stone)
Wood structure	Wood	Craftsman(stone)
Cut stone	Stone	Craftsman(wood)
Wood switch	Wood	Operator(wood)
Stone switch	Stone	Operator

2.6.2 Barriers

Barriers are objects that impede the player's progress.

2.6.2.1 Fences

Fences are the most common barrier, separating the puzzles from levels, and the player from the crystals.
There are three types of fences: wood, stone, and metal.

Wood fences can be seen through, and make a wood sound when struck. Broken sections of wood fences are worked on by the craftsman identically to a wood structure.

Stone fences cannot be seen through, and make a stone sound when struck.

Metal fences make a metal sound when struck, and outline the puzzles. These are used to separate the puzzles form the rest of the level, so actors don't interact with trees or other objects outside the level.

2.6.2.2 Force fields

Force fields are gates in fences. When a force field is on, it is impassable. Force fields are on by default, and turn off when the corresponding switch is pulled by an operator. Force fields turn off when the small crystal in their puzzle is deactivated, and emit a small force field sound when struck.

2.6.2.3 Wire supports

Wire supports are wooden poles that hold wires. The wires stretch from switches to their corresponding force field door, and from small crystals to the level's large crystal. The poles emit a wood sound when struck, and the wires emit a small force field sound.

2.6.2.4 Rivers

Rivers are uncrossable barriers. They emit a water sound when struck.

2.6.2.5 Bridges

Bridges cross rivers. There are stone, wood, and metal bridges, which emit the corresponding sounds when struck.

2.6.3 Crystals
There are two types of crystals; striking a crystal is a constant goal for the player.

2.6.3.1 Small crystals
Small crystals are the goals of each puzzle. When the player gets to a small crystal and strikes it, it emits a large pulse of crystal sound and powers down all the force fields in the puzzle.

2.6.3.2 Large crystals
Large crystals are the goals of each level. After all the small crystals are powered down and the player strikes the large crystal, a massive pulse of crystal sound is emitted, which causes the player to functionally go blind, and ends the level.

3. Artistic Design

Blinding Silence will be viewed from the perspective of a blind man who can "see" sound. Reflected sound provides the listener with details of both shape and distance, but not of color or texture. As such, the visible detail of the world around the player will be minimal, and the shapes will have to be exaggerated to ensure that the player can understand what he is seeing.

A full asset list is available in Appendix A. Reference and concept art can be seen in Appendix B.

3.1 Artistic Tools

Levels in *Blinding Silence* will be created using a combination of Autodesk Maya[13] and the C4[14] level editor. Characters and props will be created in Zbrush[15], and optimized for importation using Maya. Animations will be created using a combination of Maya and Motion Builder[16], along with motion capture data from various sources. Adobe Photoshop[17] will be used for 2D content as needed, particularly for interface and GUI elements, though there will be little demand for textures as the look will be primarily achieved through custom shaders. The custom shaders will be created within the C4 shader editor. Finally, the animatics will be created using Adobe After Effects[18].

3.2 Sonar Vision

The player is blind but can see the world through the use of Sonar Vision. All sound in the world is visible, and the louder the sound, the brighter the visibility of the object to the player. Ambient sound sources and background noise will create dim lighting of the surrounding area while individual and distinct sounds will create visible waves that illuminate anything they hit. The player will see objects hit by these waves mainly by illumination of object edges. Objects will tend to have a silhouetted look with little detail visible inside of the visible edges, as can be seen in Figure 9.

[13] http://www.autodesk.com/maya

[14] http://www.terathon.com/c4engine

[15] http://www.pixologic.com/zbrush

[16] http://www.autodesk.com/motionbuilder

[17] http://www.adobe.com/products/photoshop

[18] http://www.adobe.com/products/aftereffects/

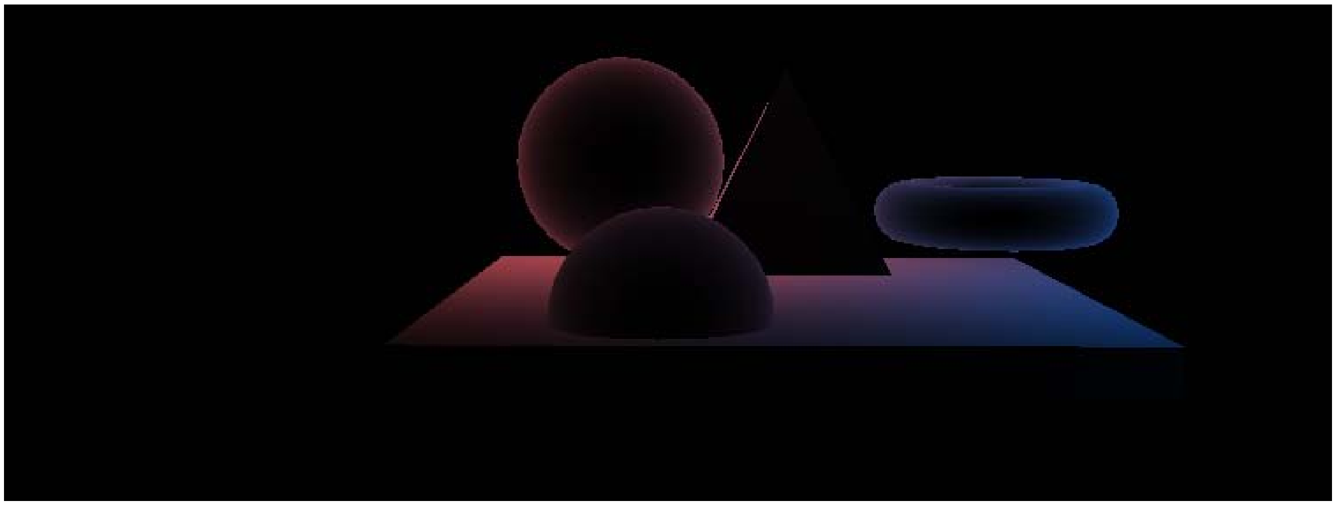


Figure 42: Simple Example of Visual Style

Each type of material will be associated with a particular color; objects made of wood produce one color, impacts on dirt another, ambient noise a third, and the like. When placed against a dark background, the distinct and bright colors will be plainly visible, allowing the player to easily localize the source of the sound. Also, by learning to recognize the different colors and their meanings, the player can quickly assess the world around him.

As an example, the player moves within audible range of a harvester who is repetitively striking his axe against a tree. The area around the harvester has a level of orange ambient illumination as his constant hammering has created a resonance in the area. However, this is a very dim illumination. Each time the harvester strikes the tree, the area immediately around him brightens considerably and slowly fades to its ambient level until the next strike. At the same time, when the tree is struck, it creates a wave of sound that moves away from it, illuminating in orange everything that it touches but with decreasing intensity the farther it moves from the tree. Shortly after becoming illuminated, the objects will start to fade back to darkness. In this way, the player is able to localize the source of the sound (the harvester will always have some level of illumination, brighter than the edges of the waves he creates), while also being able to see his surroundings as the pulses illuminate the world.

In addition to loud sounds that illuminate large swaths of the world to the player, the world will also contain objects that produce sound constantly, but on a much smaller scale. For instance, active power crystals vibrate, and energized power lines hum. These smaller objects will not produce enough sound to illuminate the area around them, but will instead have constant ambient illumination allowing the player to have static reference points within the world, while he influences and moves the larger sound producing entities.

3.2.1 Technical Implementation of Sonar Vision

To represent sound visibly, *Blinding Silence* will use the standard C4 lighting and rendering engine. Every object in the world will use a specialized shader, likely similar to a Fresnel shader. The Fresnel shader will illuminate the edges of the object, while providing less illumination to details on the portions of the object facing the player. It does this primarily by using the inverted normal direction between each vertex and the player camera, causing those vertices that point toward the camera to be darkest and those perpendicular to the camera to be brightest. This will help to provide the silhouetted art style for the game. The standard implementation of this type of shader will need to be enhanced to include normal and parallax mapping to allow for added model complexity and detail in the silhouettes. Also, the shader will be further enhanced to simulate sound absorbing objects, by factoring in the direction to a given light source so that vertices on the opposite side from the light will not be illuminated.

The ideal method of implementation is to have sound-emitting entities spawn spherical lights that will grow outward. As they grow, the intensity of the light decreases, and they are removed from the world once they reach an intensity of zero. Rather than just illuminating within the entire sphere, only the outer shell of the sphere should emit light, imitating the nature of a sound wave; only the objects within this shell would be given illumination while objects that had already passed through it would return to darkness. This could either be implemented using a spherical volumetric light, or by using a texture map on the light.

The above method is not possible using the stock capabilities of C4's lighting system, and the developer of the engine has cautioned against implementation of the volumetric light shells described above, citing excessive computational expense. As such, the initial builds of *Blinding Silence* will spawn standard point lights that will increase in radius as they decrease in intensity until they are removed from the world. Testing has shown that using two point lights per emission, one large and one small, create an effect similar to the one described. This solution provides an acceptable compromise between the artistic vision and the real world technical limitations. If time permits, an improved method may be designed and implemented.

3.3 Level Design

The levels in the first episode of *Blinding Silence* will take place entirely outdoors, within a forest. Following the story, the levels will progress from untouched forest to over-exploited clear-cut areas, ending at the outskirts of the city.

3.3.1 Layout

Level creation and layout will be accomplished using a combination of Maya and C4's native editor. All mesh objects will be created in external programs, and then imported into C4 for use as instanced reference objects. Wires, and similar objects, will be created in C4 using tube extruded along splines. Finally, the base terrain for each level will be created using a combination of C4's voxel terrain system and sculpted planes in Maya. Markers will then be placed on the terrain, providing locations and orientations for objects to be referenced into the game world. If feasible, the C4 world editor will be included in the final build along with helpful tools for level creation by players.

3.3.2 Trees

There will be three varieties of tree within the world, each at a different level of growth. Each variety of tree will have two model variations to help provide believability in the world. All three varieties are able to be used by Harvesters to produce sound. Lush trees are tall, healthy, and covered with leaves. These trees exist at the farthest points from human civilization, where their environment is just starting to be exploited. Dead trees are much shorter, appearing much more sickly and skinny and without any leaves. These trees appear further into the game, where the reach of civilization is greater. Finally, stumps are the left over remains of trees that have been cut down. These exist throughout the world, and in greater number the further the player proceeds.

3.3.3 Mines and Boulders

Boulders are massive pieces of stone found in and around mines, which the Harvesters use. There will be two variations of the boulders. The mines are shallow recesses into the terrain, supported by a simple wood structure and filled with rocks of various sizes. Only the large boulders are mineable by the Harvesters.

3.3.4 Logs

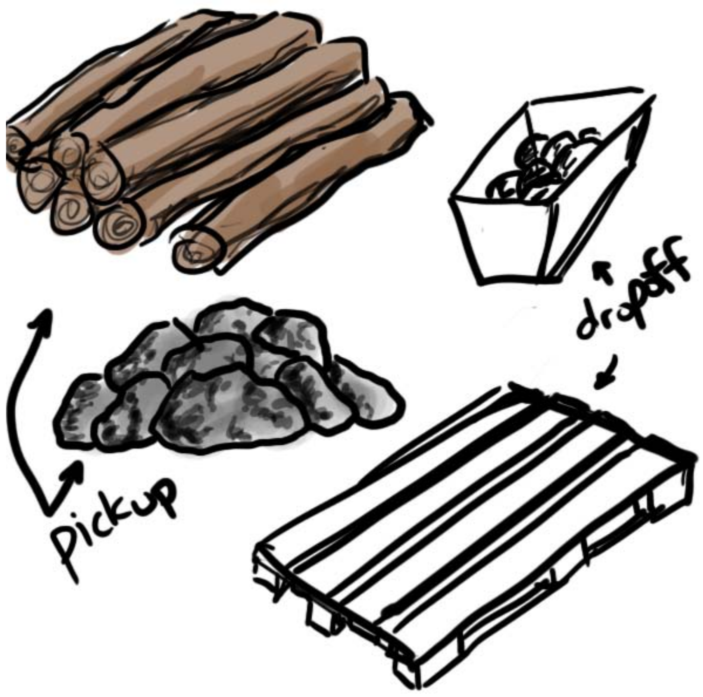


Figure 43: Carrier objects

(See Figure 2) Logs are one of two resources carried by the Carriers. Log pickup points will simply be a large pile of logs. This pile will not be neatly stacked. Conversely, log drop-off points will be neatly stacked piles of logs on top of large industrial pallets. When a log is added to or removed from a pile, the number of logs visible on the pile does not change; the resource is infinite.

3.3.5 Stones

Stones are the second resource used by the Carrier. While in transit, they resemble large roughly hewn boulders. Pickup points are simple piles of these boulders, or varying sizes and shapes, while drop-off points are large hoppers filled with them. See **FIGURE**.As with logs, stone resources are infinite.

3.3.6 Wood Structures

The wood structures are places in the level where the Craftsman works with wood. They are simple structures made of planks and logs, and are part of some larger structure that has not yet taken shape. They will often, but not always, be found near log drop-off points. There will be three variations.

3.3.7 Cut Stone

The cut stones are rocks that a Craftsman has begun to shape into a useable form, namely that of a block. They will often be found near stone drop-off points.

3.3.8 Switches

The basic form of a switch is that of a podium with a receptacle on the side into which an Operator can insert a lever. There are levers made of both wood and stone, which share the same basic form but which have the different materials built up around them. Each lever also has a moving mechanical portion, the most important part of which is a piece that moves up and down the podium support, generating a material appropriate noise on each fall. This allows the switch to self illuminate so that the player can determine what switches he has available.

3.3.9 Small Crystals

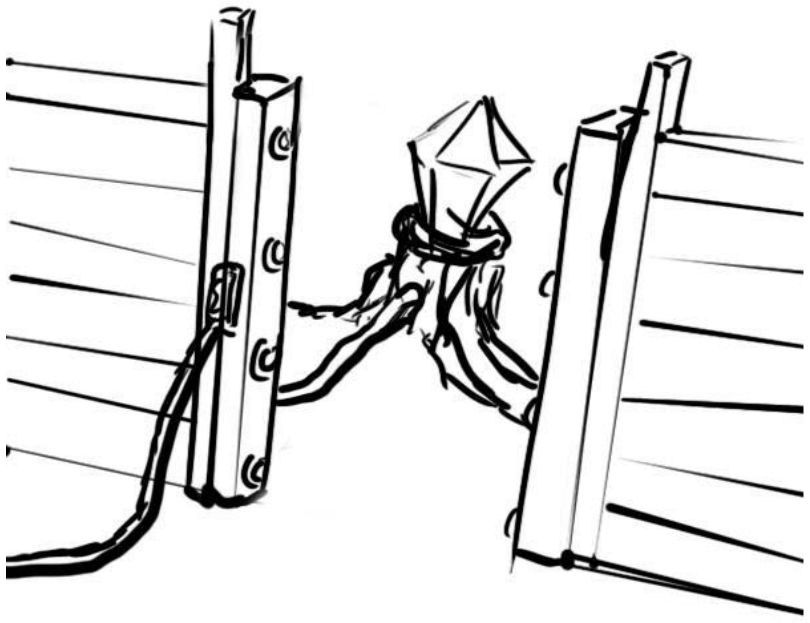


Figure 44: Small Crystal

The crystals found at the end of each puzzle are small and are embedded into a base that harnesses power from them. They can be seen in Figure 3. The base is semi-organic and has partially grown into the ground around it. All doors and switches within a given puzzle draw power from the small crystal and so energized power lines will be coming out of the base.

3.3.10 Large Crystals

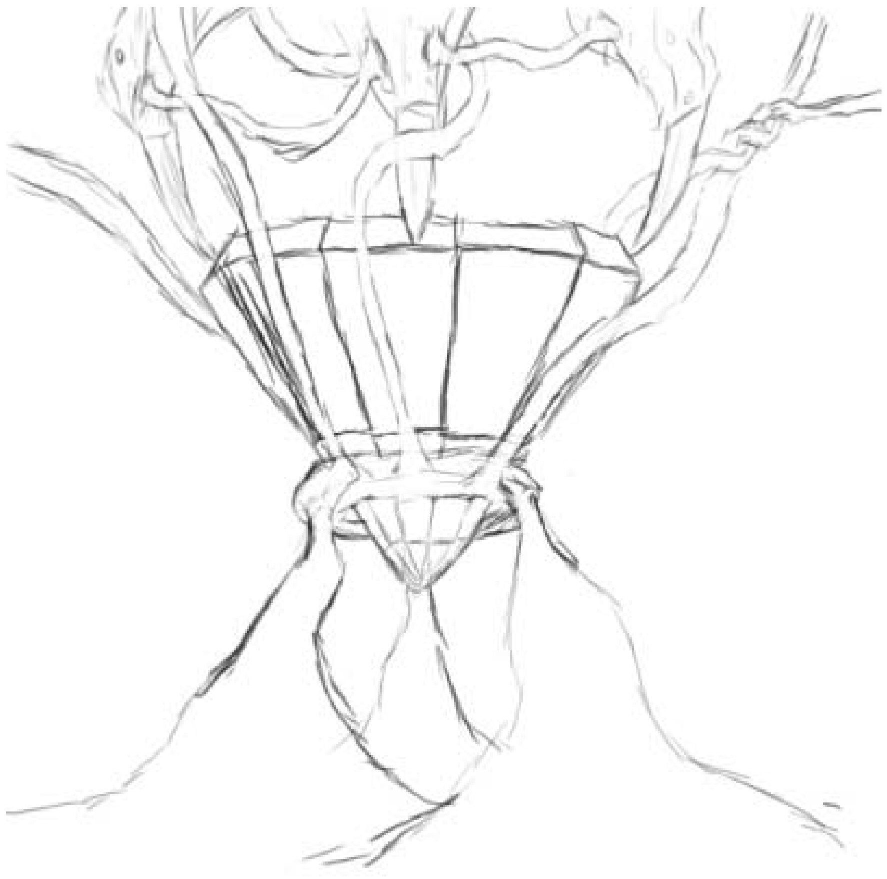


Figure 45: Large Crystal

Found at the end of each level, the large crystals seen in Figure 4 are massive, surrounded by lots of semi-organic structures that are used to draw off power. Large numbers of cables hang from it, humming with power and feeding into the structure around it. They also have power cables that connect to the small crystals.

3.3.11 Fences

There are three types of fences to impede the path of the player; those made of wood, those made of stone and those made of metal. Wooden fence segments will be constructed of wood slats, allowing the player to see through the open spaces to the area on the other side. Stone fences will be solid walls of stone blocks that the player cannot see through. Finally, metal fences are used to cordon off puzzle areas, but are low and can still be seen past.

3.3.12 Force fields

Force fields are shimmering walls, emitting a dull humming noise while active, that impede the player. When active, they are always visible but transparent. They are not affected by sound as they are not solid objects, and will be rendered in the effects pass to generate the desired effect. Force fields are surrounded by emitters, and will always be connected to a switch, as can be seen in Figure 5.

3.3.13 Wire Supports

Large telephone-pole like objects, used to support wires connecting crystals to the things that they power.

3.3.14 Rivers

Rivers provide a constant sound source, and will appear to the player as a glowing mass of shifting color.

3.3.15 Bridges

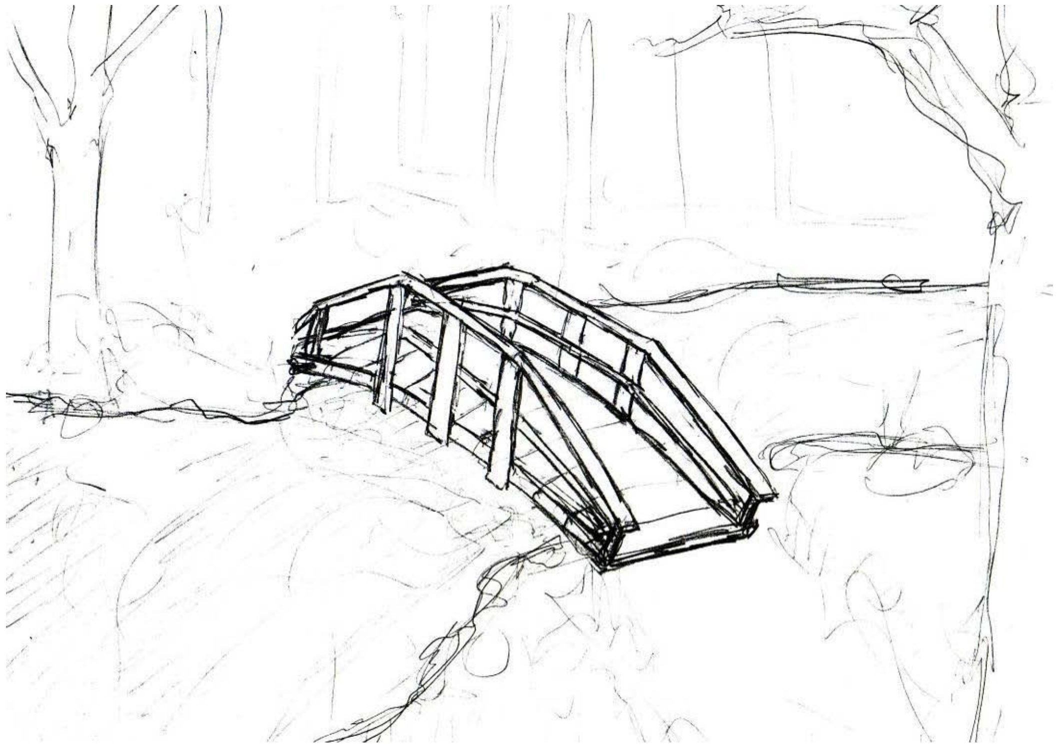


Figure 46: Bridge

Bridges provide a way for the player to cross rivers, potentially with a force field at one end. They are simple structures, created either from wood slats, stone blocks or metal planks.

3.4 Character Design

The artistic style of *Blinding Silence* does not provide much detail to the player, so the characters must be clear and recognizable. More so than in most games, it will be important for the shape and actions of the characters to be obvious from their silhouette alone.

Characters in *Blinding Silence* will be based on generic human forms, which feature realistic proportions along with a single exaggerated element. As they have been doing their jobs repetitively for a long time, they have started to change so as to do the job better. Taking a Harvester chopping a tree as an example, a man cutting a tree with an axe will appear to have a disproportionately large axe fused into his arm, in place of his hands. This will allow the player to easily recognize the different entities around him. Each type of biped character will also have distinguishing clothing and props so that the player can recognize them at a glance. Any character needing to switch between multiple props, namely the builder and harvester, will do so by removing the attachment on their arm and replacing it with another.

All bipedal characters will use a single unified skeletal rig, allowing for transfer of animation between all characters. All bipeds will be based on the same basic mesh with a socket on the right arm to hold props, and a hand on the left. They will differ only in clothing worn and props used. The one exception is the Carrier who's mesh will have two sockets instead of a hand.

3.4.1 Harvester

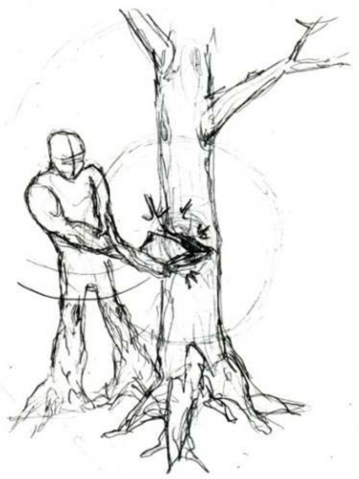


Figure 47: Harvester

The Harvester, shown in Figure 6, chops down trees and mines stone. His only article of clothing is a large exaggerated beard on his face. He switched props between a cane an axe for trees and a pick-axe for stone.

3.4.2 Craftsman

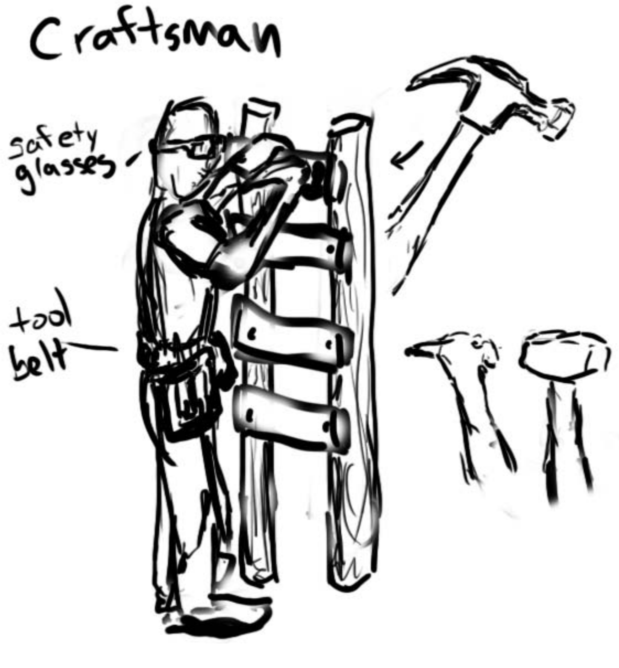


Figure 48: Craftsman

The Craftsman, shown in Figure 7, hammers nails into wooden structures and chisels stones into blocks. He wears a pair of safety glasses, as well as a tool belt. When working on wood, he uses a hammer attachment and holds the structure with his left hand. When working on stone, he uses a mallet attachment and holds a chisel in his left hand.

3.4.3 Operator

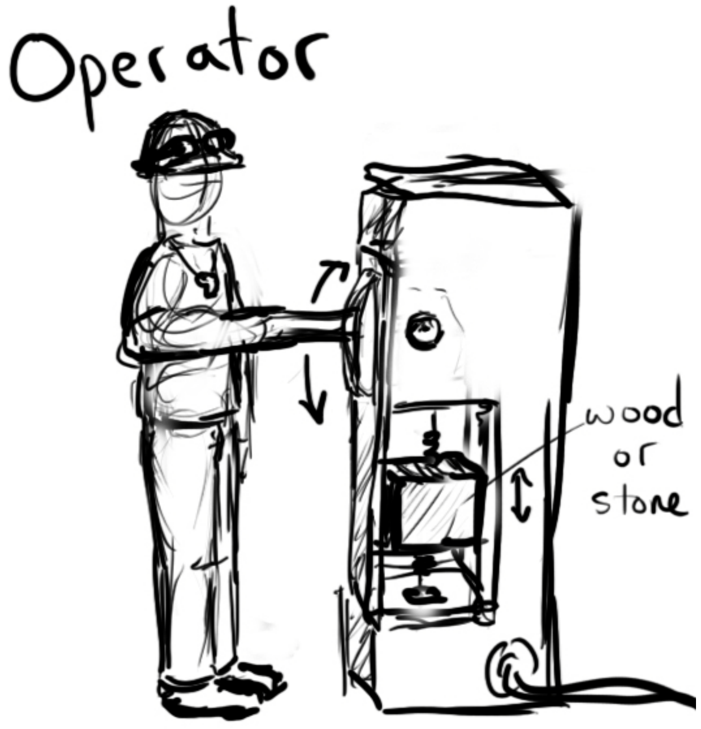


Figure 49: Operator

The Operator's (Figure 8) only job is to activate switches with his lever-arm attachment. He wears a hard hat with sunglasses on it. When not actively pulling a lever or walking to one, he stands idly and taps his foot to the game's beat.

3.4.4 Carrier

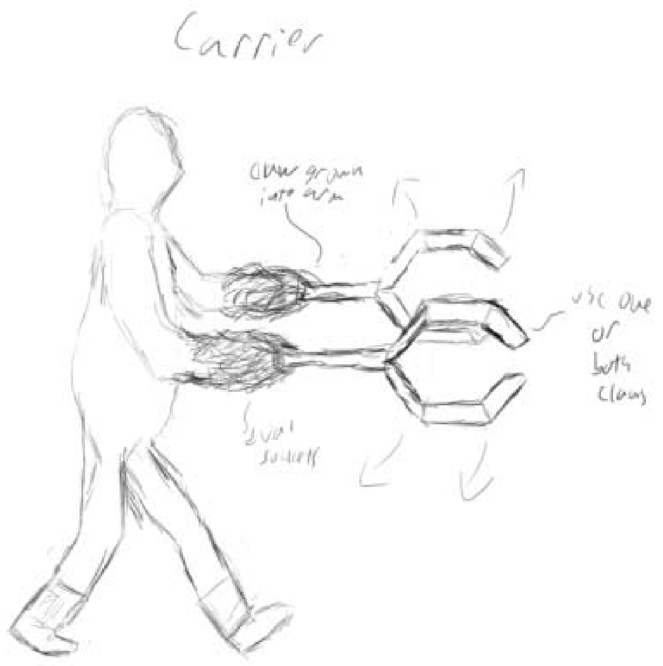


Figure 50: Carrier

The carrier, shown in Figure 9, hauls logs and stone across levels. When laden with a load, he walks half as fast and his gait changes to match the weight and position of what he is carrying. In his right arm is a claw for carrying logs over his shoulder. In his left arm is a four-pronged grabber for carrying stones at his side, similar to the grabbing arm in arcade machines.

3.5 Animation

In a world with little visual detail, accurate movement is important to help maintain believability. The diverse range of characters and interactions in *Blinding Silence* require large amounts of high quality animation, which is difficult to achieve in a short development time when created by hand. Instead, animation will be created using motion capture data, allowing for both increased realism and decreased development time. Stock animations that are appropriate to the game will be pulled from the free libraries listed in Appendix C. All other data will be captured using PhaseSpace[19] motion capture equipment borrowed from the WPI HIVE[20] lab.

3.6 Animatics

Instead of using live cut scenes, the story of *Blinding Silence* will be told through an introductory animatic and stills shown between each level. The content of these will come from two sources: first, rendered stills of certain in-game objects will be used to help maintain the connection to the game world.

[19] http://www.phasespace.com/

[20] http://web.cs.wpi.edu/~gogo/hive/

Second, photographs will be used to provide the settings of the animatics, along with objects and places that do not appear in the game. This content will then be composited and animated to match narrated voice-overs of the staff. It will then have various filters and effects applied to give it a distinctive visual look. The introductory animatic script and storyboards can be found in Appendix D and Appendix E, respectively.

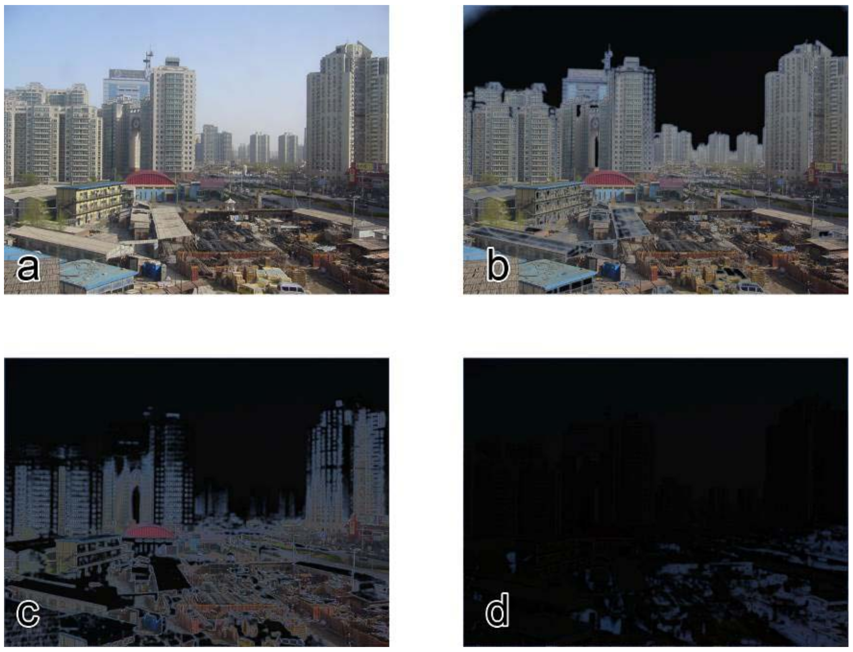


Figure 51: Animatic Progression

Figure 10 shows an example of portions of an animatic depicting darkness and silence falling across the world. Proceeding from ‘a’ to ‘d’, the city goes from completely realistic to a look that better fits the game world as seen by the player.

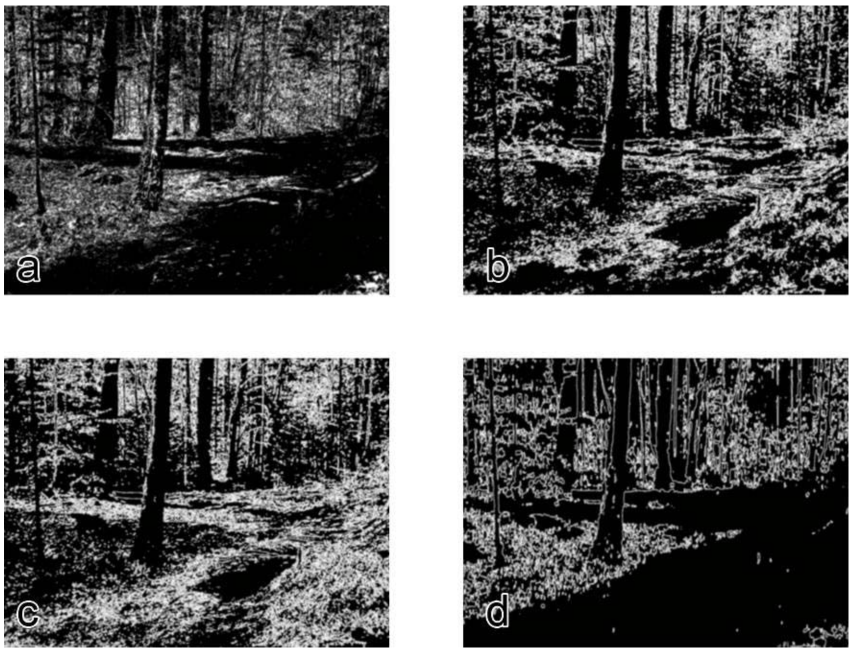


Figure 52: Example Styles

Figure 11 shows different variations in world style, stylizing a real image of a forest in a way that is closer to the simplified look of the in-game world. They range from very detailed and realistic in ‘a’, to very stylized and sparsely detailed in ‘d’, with images ‘b’ and ‘c’ providing more intermediate levels of detail. The amount of detail in any given shot will vary between these styles, depending on the tone and content at that time. Animated characters will then be added to these images, and colored sound-waves overlaid to help reinforce the image and the connection to the in-game world.

3.7 Sound Vision

Blinding Silence will not have a traditional sound score. All sounds that the player hears, even ambient sounds, will be connected to a visible cue in the game world. Through the use of a global game clock, these sound effects will be constrained to a particular rhythm. This will effectively create dynamic game music based upon the positions and states of the player and level entities.

To achieve this unique type of music, sound effects will be designed in accordance with the overall audible experience, similar to the selection of instruments in an orchestra. Realistic sounds will be recorded, then altered to result in a more exaggerated and instrumental feel. The sound of an axe hitting a tree with a constant timing, for example, will become a percussive beat in the song generated by the level.

In addition, sounds will be further associated with their visual counterparts through the use of color. Sound waves which illuminate the world will be colored; each type of sound will be represented with a distinct hue, allowing the player to recognize level elements both audibly and visibly. The colors corresponding to each sound type are shown in Table 5.

Table 2: Sound colors

Material	Color
Wood	🟨
Stone	🟪
Metal	⬜
Water	🟩
Dirt	🟫
Crystal	🟥
Force field	🟪

3.7.1 Voice Acting

The player’s staff talks to him throughout the introductory animatic and the game. To this end, voice acting will be recorded, and then processed to achieve an echoing and ghostly tone.

4. Technical Design

This section contains all relevant design challenges and solutions, as well as all relevant diagrams and charts, and justifies our technical design choices.

4.1 Engine Choice

We have chosen the C4 engine for the following reasons:

C4 has a powerful, artist friendly shading engine
C4 has an active and helpful user community
C4 has well documented source code with tutorials and example code.
C4 has a versatile world editor.

4.2 Technical Aspects

This is a listing of the main technical aspects of the project.

4.2.1 Interfacing with Wiimote and Wii motion plus

Blinding Silence will use the WiiMoteLib library found at http://www.codeplex.com/WiimoteLib to interface the WiiMote with C4. C4's Message Manager will be used to pass messages from this library to the Cane Controller.

4.2.2 Mapping the motion of the Wiimote to the in-game cane.

Once an interface between the Wiimote and the game has been established, the next challenge is mapping the input to the in-game cane.   This will have to be done in real time for one-to-one motion mapping to seem smooth. The main challenge will arise when handling what happens when a user tries to swing the cane through an in-game object. When this occurs, the in-game cane will get out of sync with the Wiimote cane spatially. This could be corrected by keeping track of where the in-game cane would be had it not impacted an object. Then, when the in-game cane's real position moves back into the area it should be in, jump the cane to where it should be.

4.2.3 Interfacing with WiiMote for Head Tracking

Interfacing with the WiiMote for head tracking will also use WiiMoteLib. We will be using this only to rotate the camera.

4.2.4 Interfacing with Microphone

C4's built in sound input capabilities will be used to handle input from the microphone.

4.2.5 Interfacing Input with C4

(Figure 12) *Blinding Silence* has four different types of input: WiiMote and WiiMotion Plus data for the cane, data from the WiiMote's infrared camera for head tracking, data from the Nunchuck for movement, and data from the microphone to create sounds.

Raw data from these input devices are passed to classes in the input device interface layer. Here the data is processed into a form more easily used. This data is then passed to the C4 Manager layer where it will be routed to the controllers that need it. Data from the WiiMote will be passed to their corresponding controllers through C4's Message Manager. Nun Chuck Information and Microphone information are passed to the player controller.

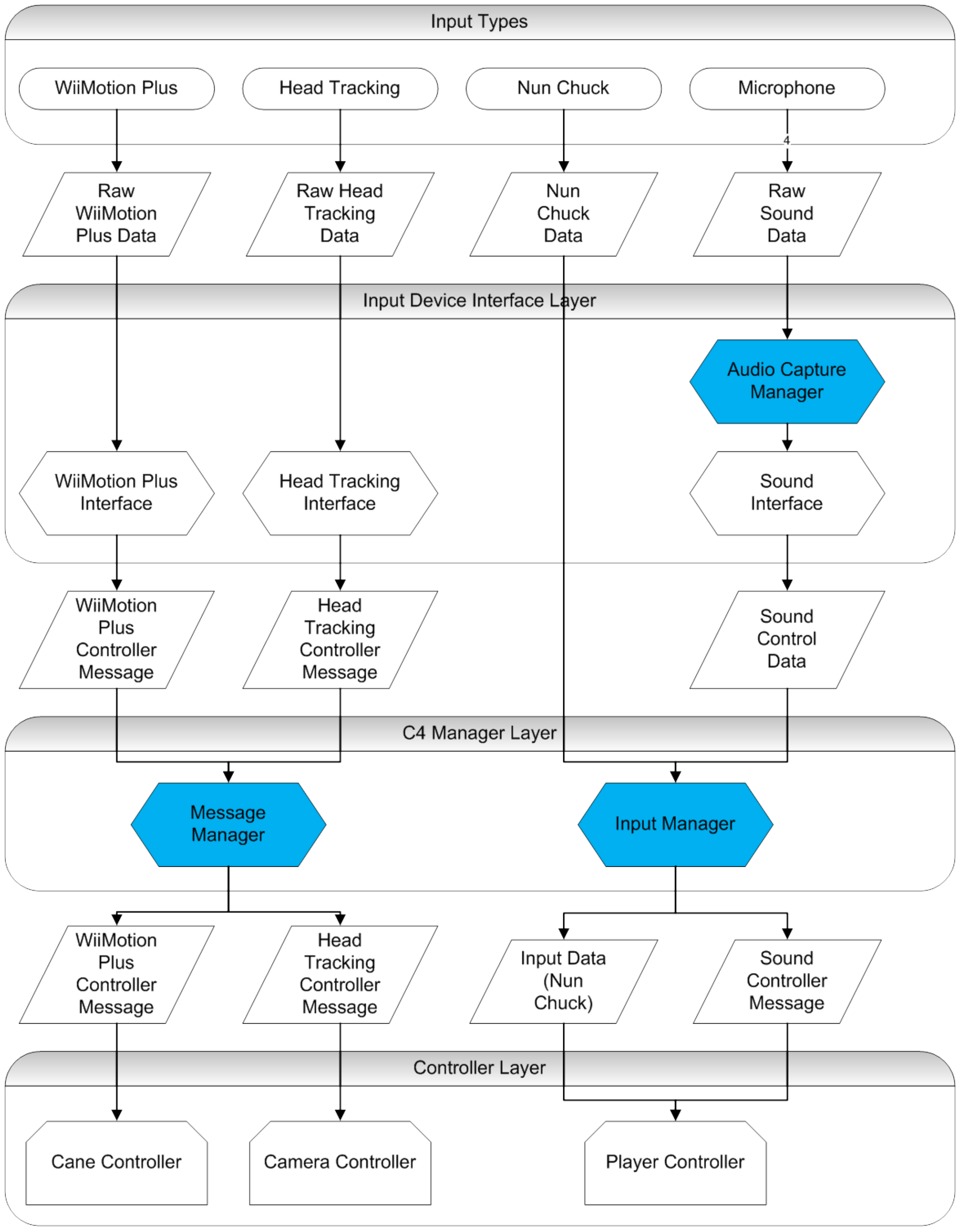


Figure 53: Input Flow

4.2.6 AI

(Figure 13) An engine for AI entities will control how an entity reacts to incoming sound stimuli.

Sound Data is passed to the Actor Controller, which queries its soul to get to the AI Engine. If there is no soul, then there is no AI and therefore no way to get a reaction. If there is a soul, then the sound data is passed to the AI Engine connected to the soul.

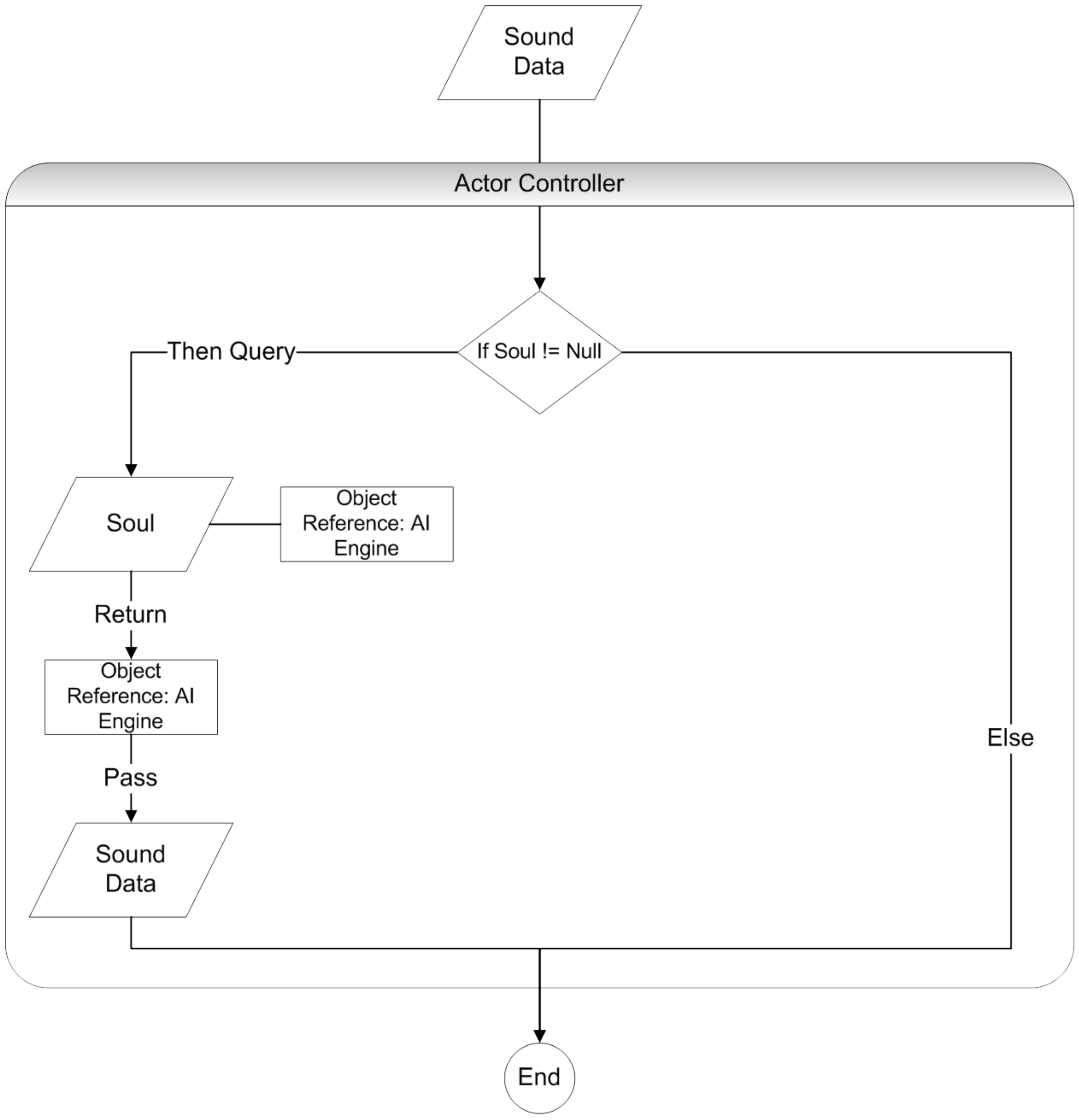


Figure 54: Actor Sound Flow

The Sound Data is then passed on to the AI Engine (Figure 14). The AI Engine queries its rules list to see if any rules match the incoming stimuli. If the rule matches stimulus, it the checks the stimulus against the rules condition graph. If the stimulus matches, the AI Engine applies the Rule's Action Controller to its Action List. The Action controller can add or remove action data from the Action List. When an actor needs to determine what action to take next, the actor queries its AI Engine, which returns the Action Data at the head of the Action List.

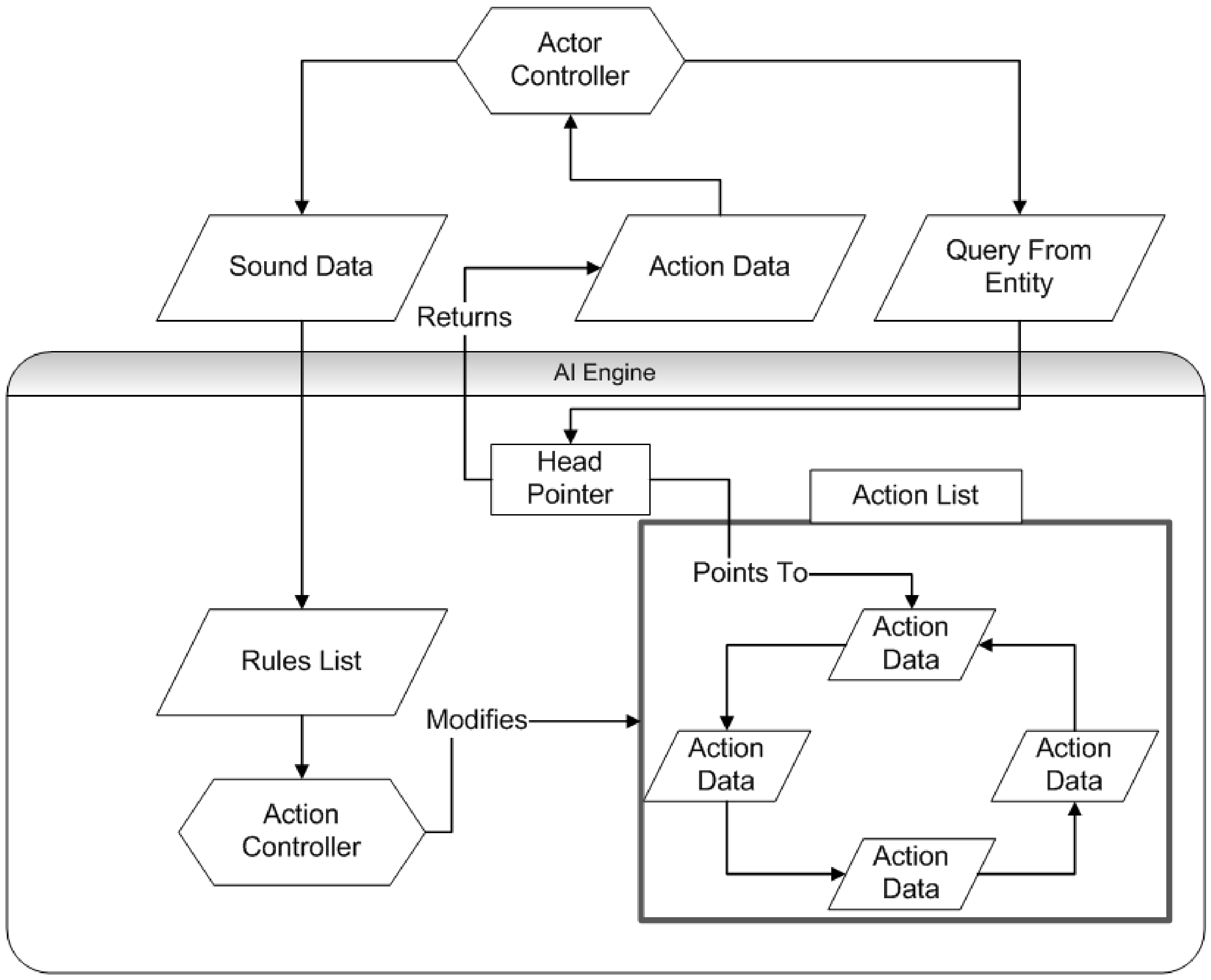


Figure 55: AI Engine

4.2.7 Interfacing AI Engine with event system

The AI Engine will be interfaced with the game engine's event system. This will happen through the entity containing the AI Engine, which will be connected to the Sound Engine Controller. The Entity containing the AI Engine will have registered itself with the Sound Engine Controller, whose job it is to keep track of what entities need to know about sounds.

4.2.8 Determining which entities should receive incoming sounds

(See Figure 15) All entities that can receive sound will be registered with the Sound Engine Controller. When an entity creates a Sound Data Object it is immediately passed to the Sound Engine Controller, which will take that sound data and pass it to all the entities registered with it.

The Sound engine is also responsible for creating OmniSource nodes for the sound and point lights for the light sources in the rendering engine.

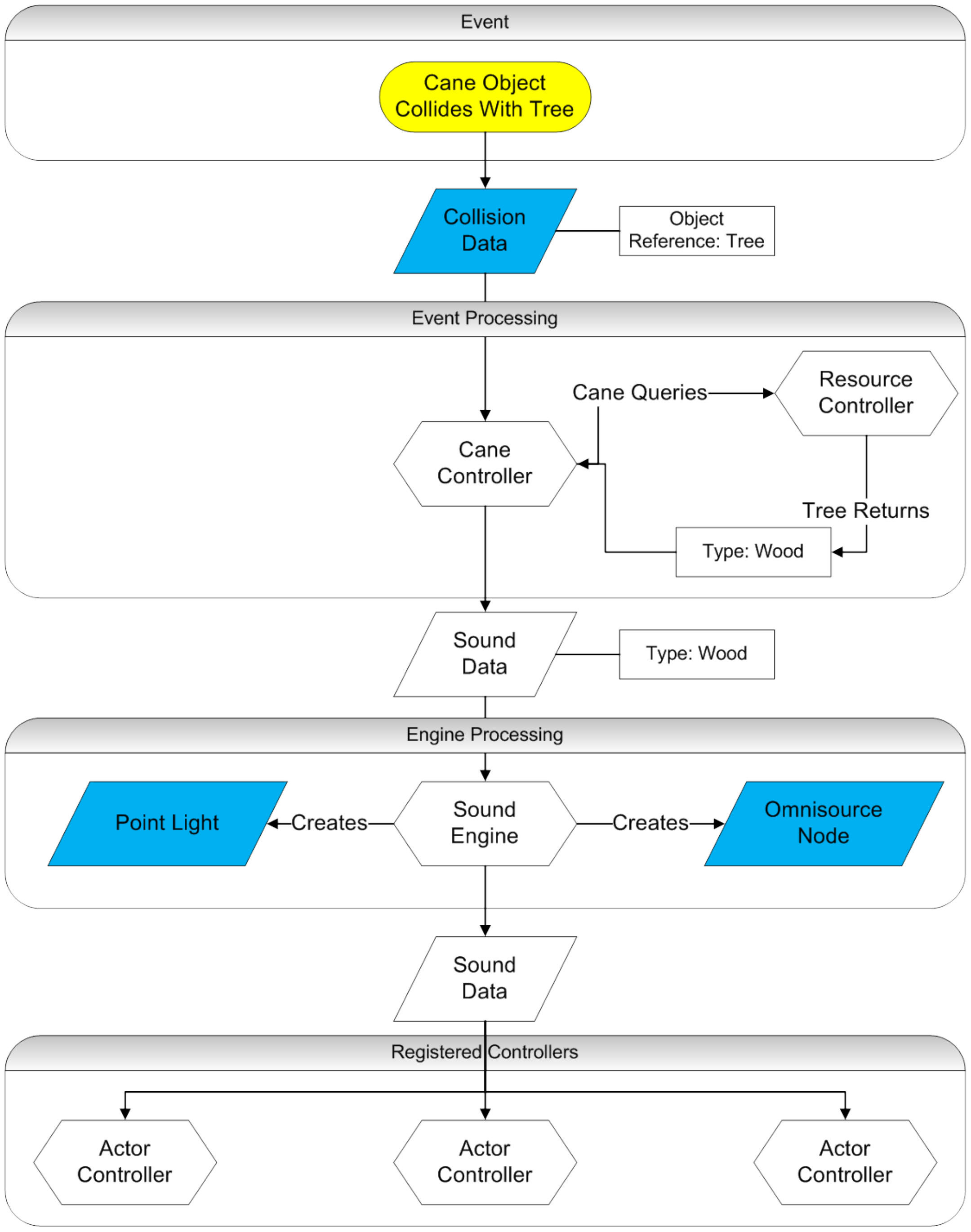


Figure 56: Sound Flow

4.2.9 Lighting Engine

Blinding Silence will use point lights with a growing radius and a diminishing intensity to simulate the “look” of the sound. This relatively straight forward lighting solution will allow the development team to quickly create the lighting engine. Another solution that produces a better looking result can then be implemented later if time allows. The generation and destruction of point lights are handled by the Sound Engine (See figure 8).

Path finding

Blinding Silence will make use of a navmesh for the path finding of actors. This will allow actors to more naturally move about the level. There is a free tool for automatically generating navmeshes in C4, found here http://www.terathon.com/forums/viewtopic.php?f=4&t=7793.

Collada Importer Changes

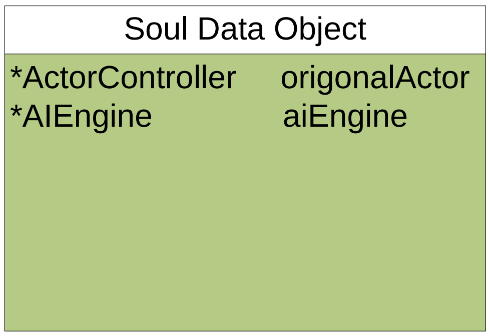
Changes will be made to the World Editors Collada Importer to allow for the direct importation of references markers from Maya. This will help improve the artists work flow. Currently the artists need to place locator markers in Maya, and then manually play reference markers at their location in the world editor if they want to make multiple instances of objects. This change will allow the artists to place the reference markers directly in Maya, saving them a great deal of time.

4.3 Data Objects

In *Blinding Silence* all data will be encapsulated its own data object to increase the extendibility of the game infrastructure.

4.3.1 Souls

Every actor in the game has a “soul”, a driving force that causes the actor to react to stimuli and take actions in the world. Souls can and sometimes need to be taken from actors. The player does this by striking them with his staff. In this case the soul is “stored” in the staff for later use. This use is either powering a small crystal or destroying a large crystal.



Since a soul has so many uses, it will be represented by its own data object. A soul object will contain the reference to the actor's AI Engine. Since an actor goes through its soul to get to its AI Engine, removing the soul automatically removes the actor’s ability to respond to stimuli. The soul also includes a reference to the entity it was originally attached to, so that when souls are “returned” the engine knows where to put them.

4.3.2 Cane

The cane is the player's main way of interacting with the world. The cane will only need to store two pieces of custom data. The first piece of data the cane needs to store is a list of souls currently stored within the cane. This information is necessary to perform several cane specific actions, such as playing the sound of the most recent stored soul or powering a crystal.

The second piece of data the cane needs to store is its real world position and orientation. This is necessary for re-syncing the real world cane with the in game cane if the in game cane collides with an object and the player continues swinging.

Cane Data Object	
ArrayList<Soul>	type
LocationObject	realLocation

4.3.3 WiiMotion Plus Data

The WiiMotion plus is used in game to control the cane. The data coming in from the Wiimote has six values, linear acceleration in the x, y, and z directions and rotation around the x, y, and z axis. The WiiMotion Plus data object will encapsulate this information for easy access and transfer.

WiiMotion Plus Data Object	
Float	accelerationX
Float	accelerationY
Float	accelerationZ
Float	yaw
Float	pitch
Float	roll

4.3.4 Wii Head Tracking Data

There will be head tracking information coming in from a separate WiiMote from the cane. This data is the location of two points in the xy plane. The Wii Head Tracking Data Object will therefore contain Point2D objects, which contain an x location and a y location.

Head Tracking Data Object	
Point2D	point1
Point2D	point2

4.3.5 Pile Data

In the game world there are piles of specific resources. These piles need to have a type, such as wood or stone, and an amount of a resource in the pile.

Pile Data Object	
PileType	type
Integer	remaining

4.3.6 Action Data

Every action in game that an actor can take will be encapsulated in a generic action data object. This object contains type information, what animation to play, and delay information.

Action Data Object	
ActionType	Type
Animation	animation
Long	delayTime

4.3.7 Sound Data

Every sound has a set of information associated with it. A sound has an origin which needs to be stored, such as the tree a player hit. The sound data contains an initial volume, determined by the force at which the object was hit. The object that was hit determines the type of the sound made.

Sound Data Object	
Point3D	origin
String	soundName
Float	volume
*ActorController	actor

A sound also knows what resource was hit, what sound is being played, and the originating actor.

4.3.8 AI Engine Data

The AI Engine drives the stimulus response system of an actor. The AI Engine needs to keep track of a circular list of actions. It also needs to keep track of a list of rules.

AI Engine Data	
CircularList<ActionData>	aList
List<Rule>	rules
ListNode	head

4.3.9 Rule Data

A rule encapsulates the stimulus/response system for the AI Engine. A rule contains a stimulus, an event that the actor containing the AI Engine containing the rule needs to respond to. A rule also contains a Condition Graph that checks to see if the stimulus meets a set of perquisites. The rule finally needs a set of action controllers.

Rule Data Object	
SoundData	stimulus
Condition Node	head
List<ActionController>	actions

4.3.10 Condition Graph Node

The condition graph is a binary tree of conditional values used by a Rule in the AI Engine. A condition graph is a binary tree of conditionals applied to values of an input sound stimulus in a rule to determine if the stimulus meets the necessary minimum criteria associated with the rule. Each node has a type corresponding to one of the following:

A variable in the stimulus. This type has no children
A number. This type has no children.
A Boolean operator. This type has no children.

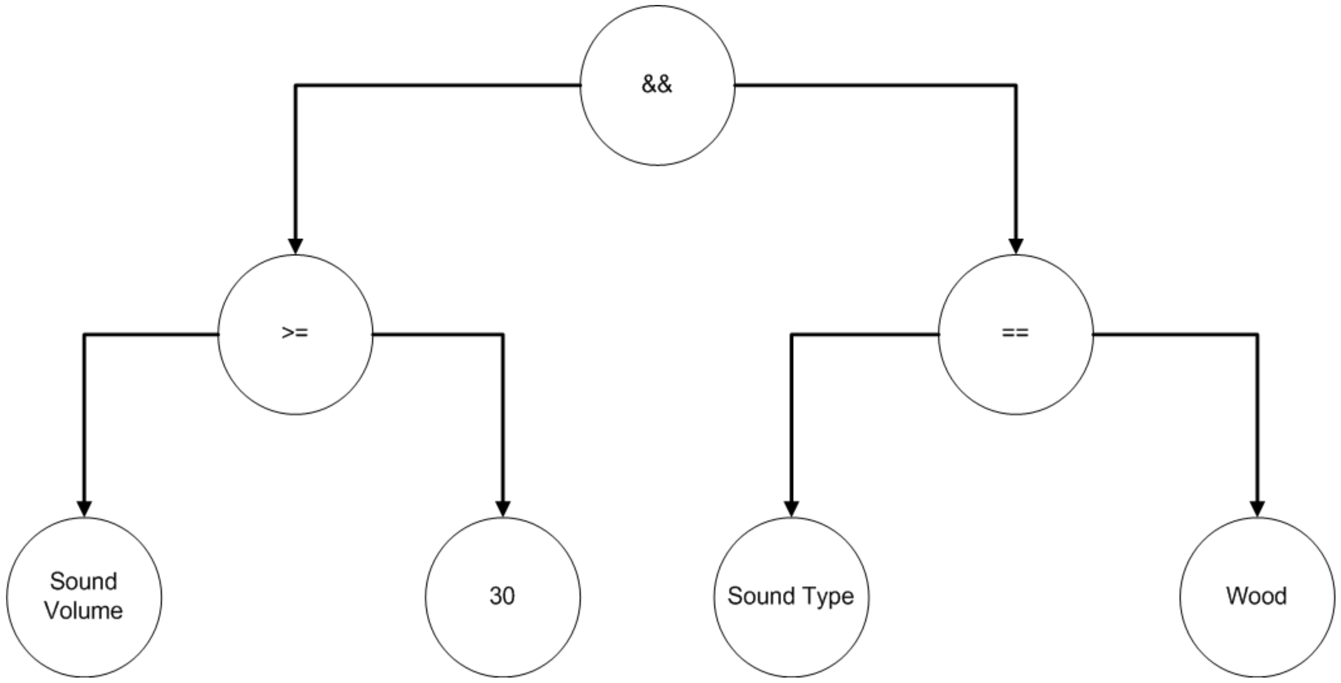


Figure 57: Example Condition Graph

Figure 16 shows how the condition if ((sound volume >= 30) && (sound type ==

wood)) would be described as a binary node tree.

4.4 Controllers

A controller is an object that takes an action on a piece of data. Not all controllers listed here are C4 controllers. Those that are C4 controllers list that they inherit from a C4 controller.

Controller	Description	Makes Changes On	Recieves Data From	Sends Data To	Inherits From
Player	A Player Controller handles moving the players' in-game avatar around.	Player Entity	Input Manager		C4 Character Controller
Actor	A generic controller for in game actor. Handles moving actor entities around the level, playing animations, and generating sound when taking an action.	Actor Entity	Receives Action Data after querying the AI Engine Controller Recieves Sound Data from Sound Controller	Sends location information to the Sound Controller	C4 Character Controller
AI Engine Controller	The AI Engine Controller determines what action an actor should currently be doing. The AI Engine also runs Action Controllers when an incoming stimulus meets the requirements in one of its rules.	AI Engine Data, Action Data	Receives sound stimulus data from the actor controller	Send Action Data to the Actor Controller	C4 Controller
Pile Controller	This Controller modifies pile data when either	Pile Data	Recieves trigger information from a C4 Trigger	Sends resource data to an Actor entity	C4 Controller

	a carrier or a resource gather does add or subtract from a pile		when an actor approaches the pile.		
Mover Controller	A mover controller controls objects the move either a player entity or an actor entity in some fashion. Entities are unable to move while this is in progress.	Player Entity Actor Entity	Recieves data from Interactor Controller when the InterActor is used		C4 Controller
Interactor Controller	An interactor controller controls objects that are interactive, such as levers . These levers then cause changes in the environment.	Interactor Data	C4 actor when interacted with	Mover Controller	C4 Controller
Sound Engine Controller	All entities able to hear sound register with this controller. It then routes sounds to entities within range of the sound. The Sound Engine Controller is also responsible for playing the sounds in game	Sound Data	Controllers of Entities capable of making sound	Any controller registered to hear sound	
Cane	This control	Cane	Recieves	Sends Sound	C4 Capsule

Controller	controls the movement and position of the cane. This controller also handles the mapping of	Entity	WiiMotion Plus Data from the Message Manager Recieves Collision Data from the C4 Collision System	Data to the Sound Engine Controller	Collider
Resource Controller	A resource controller handles the storing of resource specific data, as well as transferring resources to piles	Actor Controller	Actor Controller		C4 Controller
Crystal Controller	A Crystal Controller is the controller for all Crystals	Crystal Entity	Cane Controller		C4 Controller
Action Controller	An action controller is a controller that modifies the action list inside of an AI Engine. It can both add and remove data.	AI Data			
Beat Engine Controller	The beat engine controller keeps track of what musical beat the engine is on. Other controllers query it to determine if they should take action.	Timing Data	Actor Entities query the Beat engine to see if they should start their action	Send Information on Current Beat to Actor Entities	

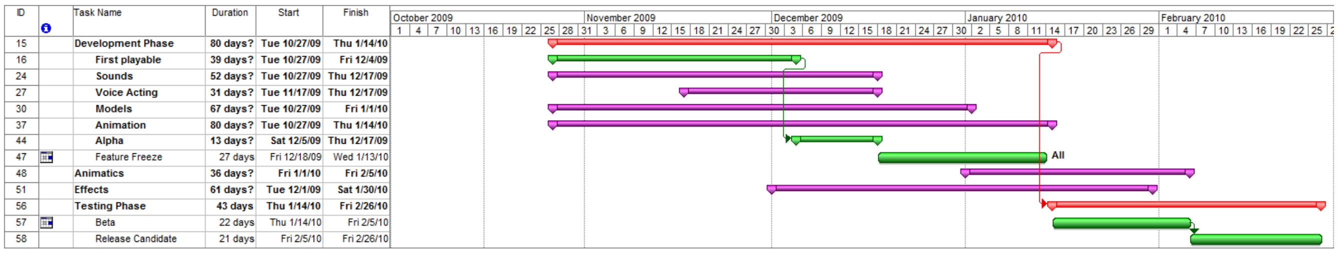
5. Project Planning

5.1 Gantt Charts

Gantt charts are visual representation of schedules.

5.1.1 Overview

Figure 17 shows an overview of the Development and Testing phases, which occur over B and C terms.

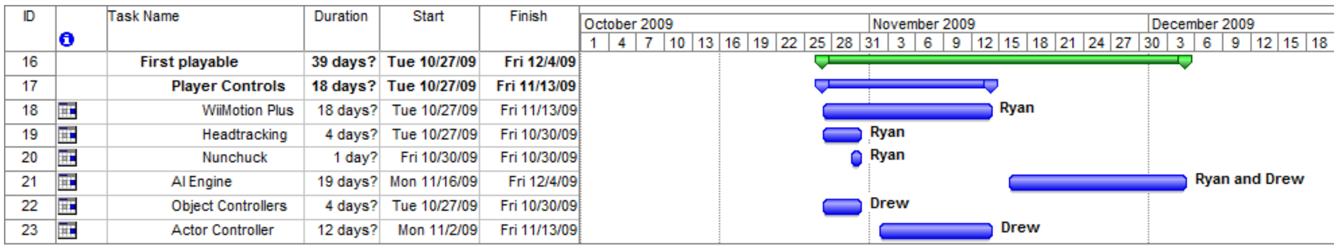


ID		Task Name	Duration	Start	Finish
15		**Development Phase**	**80 days?**	**Tue 10/27/09**	**Thu 1/14/10**
16		**First playable**	**39 days?**	**Tue 10/27/09**	**Fri 12/4/09**
24		**Sounds**	**52 days?**	**Tue 10/27/09**	**Thu 12/17/09**
27		**Voice Acting**	**31 days?**	**Tue 11/17/09**	**Thu 12/17/09**
30		**Models**	**67 days?**	**Tue 10/27/09**	**Fri 1/1/10**
37		**Animation**	**80 days?**	**Tue 10/27/09**	**Thu 1/14/10**
44		**Alpha**	**13 days?**	**Sat 12/5/09**	**Thu 12/17/09**
47	▦	Feature Freeze	27 days	Fri 12/18/09	Wed 1/13/10
48		**Animatics**	**36 days?**	**Fri 1/1/10**	**Fri 2/5/10**
51		**Effects**	**61 days?**	**Tue 12/1/09**	**Sat 1/30/10**
56		**Testing Phase**	**43 days**	**Thu 1/14/10**	**Fri 2/26/10**
57	▦	Beta	22 days	Thu 1/14/10	Fri 2/5/10
58		Release Candidate	21 days	Fri 2/5/10	Fri 2/26/10

Figure 58: Overview Gantt

5.1.2 First Playable

Figure 18 shows the programming team will be completing a playable build of *Blinding Silence* by December 4th.

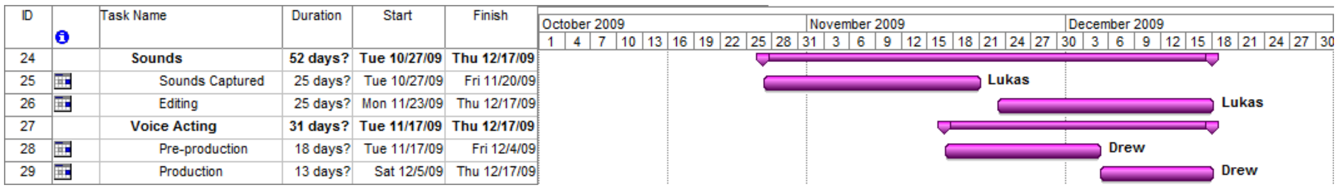


ID		Task Name	Duration	Start	Finish
16		**First playable**	**39 days?**	**Tue 10/27/09**	**Fri 12/4/09**
17		**Player Controls**	**18 days?**	**Tue 10/27/09**	**Fri 11/13/09**
18	▦	WiiMotion Plus	18 days?	Tue 10/27/09	Fri 11/13/09
19	▦	Headtracking	4 days?	Tue 10/27/09	Fri 10/30/09
20	▦	Nunchuck	1 day?	Fri 10/30/09	Fri 10/30/09
21	▦	AI Engine	19 days?	Mon 11/16/09	Fri 12/4/09
22	▦	Object Controllers	4 days?	Tue 10/27/09	Fri 10/30/09
23	▦	Actor Controller	12 days?	Mon 11/2/09	Fri 11/13/09

Figure 59: First Playable Gantt

5.1.3 Sounds and Voice Acting

Figure 19 shows the audio effects of the game will be incorporated by December 17th.

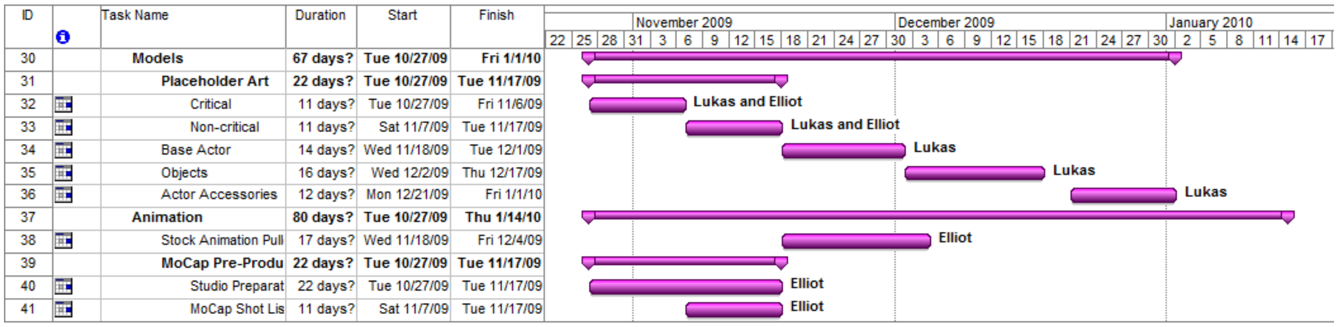


ID		Task Name	Duration	Start	Finish
24		**Sounds**	**52 days?**	**Tue 10/27/09**	**Thu 12/17/09**
25	▦	Sounds Captured	25 days?	Tue 10/27/09	Fri 11/20/09
26	▦	Editing	25 days?	Mon 11/23/09	Thu 12/17/09
27		**Voice Acting**	**31 days?**	**Tue 11/17/09**	**Thu 12/17/09**
28	▦	Pre-production	18 days?	Tue 11/17/09	Fri 12/4/09
29	▦	Production	13 days?	Sat 12/5/09	Thu 12/17/09

Figure 60: Sounds and Voice Acting Gantt

5.1.4 Models and Animation

Figure 20 shows the models and their animations will be completed by January 14th.

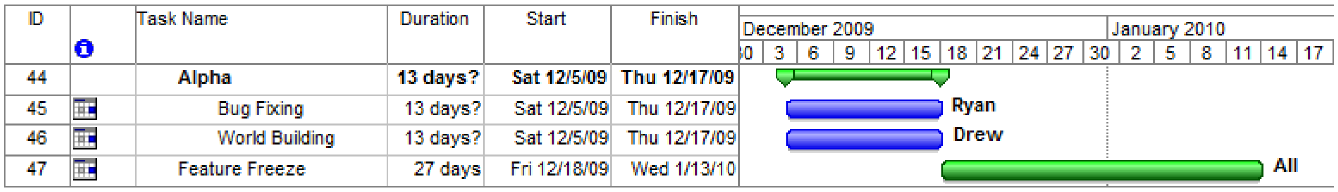


ID		Task Name	Duration	Start	Finish
30		**Models**	**67 days?**	**Tue 10/27/09**	**Fri 1/1/10**
31		**Placeholder Art**	**22 days?**	**Tue 10/27/09**	**Tue 11/17/09**
32	▦	Critical	11 days?	Tue 10/27/09	Fri 11/6/09
33	▦	Non-critical	11 days?	Sat 11/7/09	Tue 11/17/09
34	▦	Base Actor	14 days?	Wed 11/18/09	Tue 12/1/09
35	▦	Objects	16 days?	Wed 12/2/09	Thu 12/17/09
36	▦	Actor Accessories	12 days?	Mon 12/21/09	Fri 1/1/10
37		**Animation**	**80 days?**	**Tue 10/27/09**	**Thu 1/14/10**
38	▦	Stock Animation Pull	17 days?	Wed 11/18/09	Fri 12/4/09
39		**MoCap Pre-Produ**	**22 days?**	**Tue 10/27/09**	**Tue 11/17/09**
40	▦	Studio Preparat	22 days?	Tue 10/27/09	Tue 11/17/09
41	▦	MoCap Shot Lis	11 days?	Sat 11/7/09	Tue 11/17/09

Figure 61: Models and Animation Gantt

5.1.5 Alpha and Feature Freeze

Figure 21 shows the programming team will complete an Alpha build by December 17th, and will spend B-C break working on a feature freeze build.

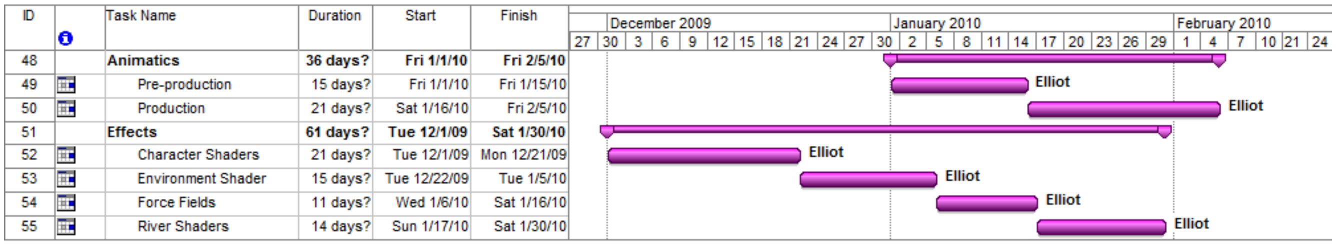


ID		Task Name	Duration	Start	Finish
44		**Alpha**	**13 days?**	**Sat 12/5/09**	**Thu 12/17/09**
45	▦	Bug Fixing	13 days?	Sat 12/5/09	Thu 12/17/09
46	▦	World Building	13 days?	Sat 12/5/09	Thu 12/17/09
47	▦	Feature Freeze	27 days	Fri 12/18/09	Wed 1/13/10

Figure 62: Alpha and Feature Freeze Gantt

5.1.6 Animatics and Effects

Figure 22 shows the animatics and secondary visual effects will be completed over the end of B term and the beginning of C term, straddling phases.



ID		Task Name	Duration	Start	Finish
48		**Animatics**	**36 days?**	**Fri 1/1/10**	**Fri 2/5/10**
49	▦	Pre-production	15 days?	Fri 1/1/10	Fri 1/15/10
50	▦	Production	21 days?	Sat 1/16/10	Fri 2/5/10
51		**Effects**	**61 days?**	**Tue 12/1/09**	**Sat 1/30/10**
52	▦	Character Shaders	21 days?	Tue 12/1/09	Mon 12/21/09
53	▦	Environment Shader	15 days?	Tue 12/22/09	Tue 1/5/10
54	▦	Force Fields	11 days?	Wed 1/6/10	Sat 1/16/10
55	▦	River Shaders	14 days?	Sun 1/17/10	Sat 1/30/10

Figure 63: Animatics and Effects Gantt

6. Conclusion

Blinding Silence is a simple but fun puzzle game. Its straightforward puzzle elements are enhanced by the control and viewing systems, which are unique enough to intrigue without distracting from the tasks at hand. The development of *Blinding Silence* will be completed over the course of approximately 3 months, leaving two months for thorough testing of the gameplay and puzzles.