****

**Team Doge**

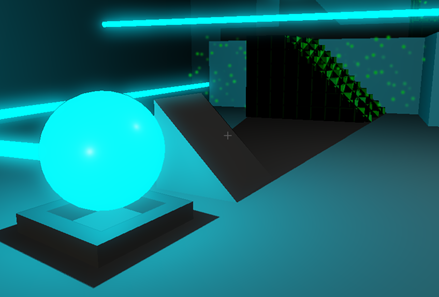
**Ben Allen, Jin Cheah, Derrick Dent, Gwen Stevens, Logan Wilson**

****

Table of Contents

* Introduction/Overview
  + Development Specification
* Game Mechanisms
  + Core Mechanic
  + Player roles and actions
    - Walking
    - Picking up objects
  + Environment
  + Appearance
  + Sound
  + Console
* Game Elements
  + Universal Elements
    - Scoring Rules
    - Names
    - Special Powers
    - Algorithms
  + Other Elements
* Algorithms
  + Deforming Blocks with Shadows
  + Optimizing Shadows
* Story Overview
  + Character Description
  + Player Goals and Achievements
  + Scripts
* Game Progression
* Bibliography

Introduction/Overview

Element Zeta is a 3D, first-person puzzle game, focusing on the way lights and shadows affect the game environment. In this world, there is a new substance called Element Zeta. It is emitted as a beam and can be diffused into light. The light from Element Zeta has the ability to affect other substances called Light Solid and Dark Solid. The player navigates various levels and solves the puzzles within them. The puzzles become more challenging and new mechanics are introduced as the levels progress. The player unlocks a new level after each puzzle is solved, and the goal is to complete every level.

# Development specification

* Hardware
  + Windows
* Software
  + 3d engine: Unity
  + Programmed using C#
* Estimated development time
  + 3 months

Game Mechanisms

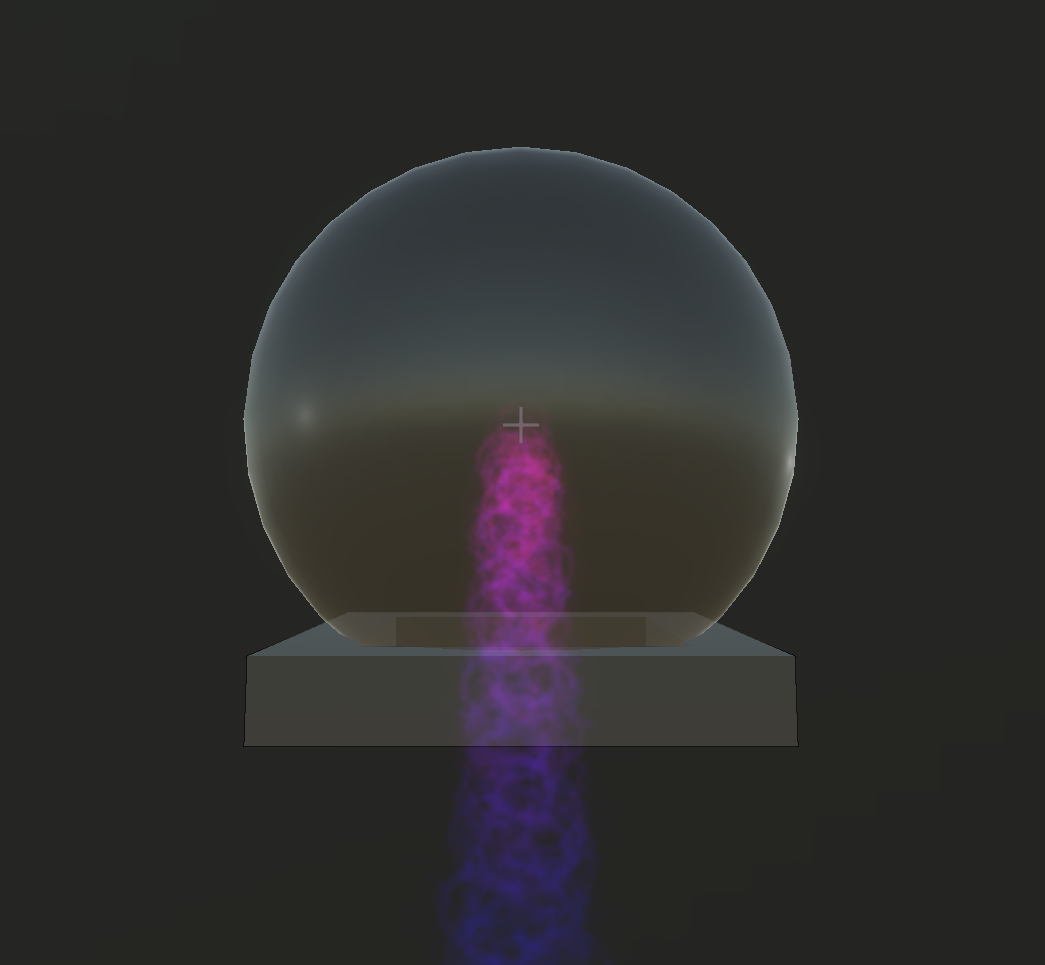
# Core Mechanic

The central mechanic is that light can change the environment. When Dark Solid blocks are in the presence of light, they dematerialize. If a shadow is cast on the Dark Solid block, only the part outside the shadow will dematerialize, leaving the block with an altered shape. The inverse is true for Light Solid blocks. Using this mechanic, the player can construct ramps or destroy obstacles using lights and shadows.

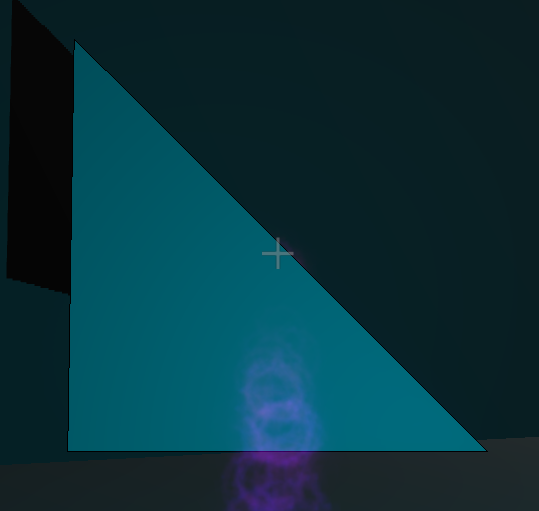
# Player roles/actions

The player has three possible actions. They can walk around and pick up certain objects.

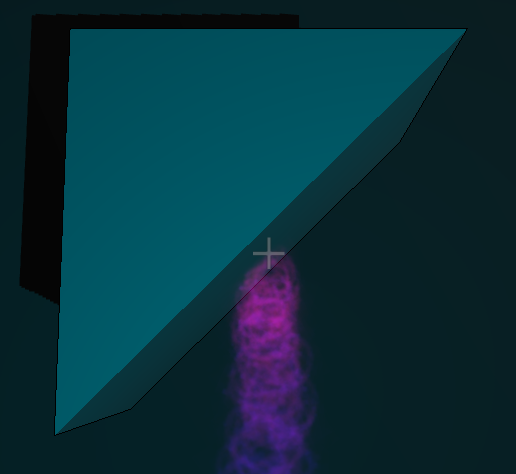
Walking

* + The player must walk on solid objects. A player may not walk on top of a dark solid if it has light on it. Similarly a player may not walk on light solid if it has shadow on it.

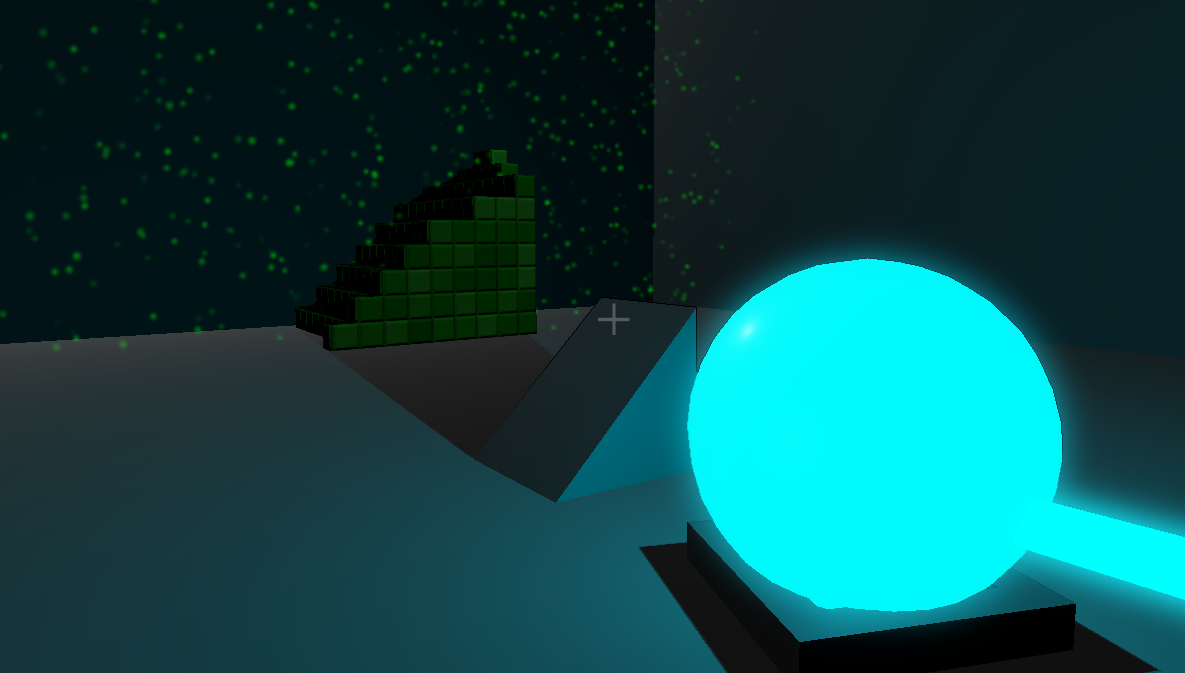
Picking up objects

* + The player may pick up certain objects with “E”. Once an object is in the player's grasp, they may rotate it in order to set it down where it is needed. The player can position them to cast shadows, turn on diffusers, and block lasers to remove obstacles or create platforms that allow them to continue in the level. To denote that a is holding the object, a purple particle beam emits from the player to the center of the object.

Rotating objects

* + The player may rotate objects with “IJKL” that they are holding such as triangles and redirectors in order to use them more efficiently.

# Environment

The environments may have a number of stationary pre-placed elements in it that the player can move. There may also be moving obstacles. Rather than these moving objects directly interfering with the player, they interfere with the light and shadow. This makes it harder for the player to traverse through the level as light and dark solid seemingly move along with the moving object’s shadow.

# Appearance

The game’s aesthetic will be minimalistic. Levels will be designed around simple shapes with no texture. The primary environment color will be white. This will provide a very sterile appearance which can be affected by the lighting. Since light and shadow are the central mechanics, the intention is to have the color scheme of the level be defined by the colors of the lights, rather than the colors of the environment.

# Sound

The music will sound like a mix of ambient and lounge music. It’s meant to be calming, ethereal, meditative, and mixed with a bit of electronic influences.

The following explains the characteristics of the music:

|  |  |
| --- | --- |
| Category | Definition |
| Calming | + Does not agitate the user.  + Music with slow or medium tempo without any rushing of tempo so that no tension exists.  + Music must not take away from the puzzle by being too loud or having too many instruments playing at a moment. |
| Ethereal | + Orchestral sounds(such as usage of the organ instrument) + Choir/hymn sounds(the HMMMs, OOOHs, and AAAHs of a church choir) + Any sounds that feel harmonic to this theme (such as a harp.) |
| Meditative | + Repetitive or be in the background(as part of the scenery, as part of the level.)  + Examples: Repeating droning sounds from electronic synthesizers, sounds of nature(sound of recorded rain, flowing rivers, muted thunder), etc. |
| Electronic Influences | + Sounds produced as a result of electronic synthesizers.  + Virtual synthesizers work just as fine. |

# Sound System

The following is the implementation of the sound system:

* Any grabbable object will have a sound attached to it when it collides to with other objects in the room, except for the player.
* Some sounds are attached to triggers and are only run when the player is in them. Example would be the winning pedastal.
* Some sounds are played only after a puzzle is solved or level has ended.
* Some sounds are played as background music.
* Some sounds are made when buttons are clicked.

The music system is designed to allow ease of attaching music to any objects in the scene. There are two classes of great importance to facilitate this purpose.   
  
The first class is the SoundManager. It is a class containing methods to play, or stop sounds and a hidden library of music that’s currently playing.

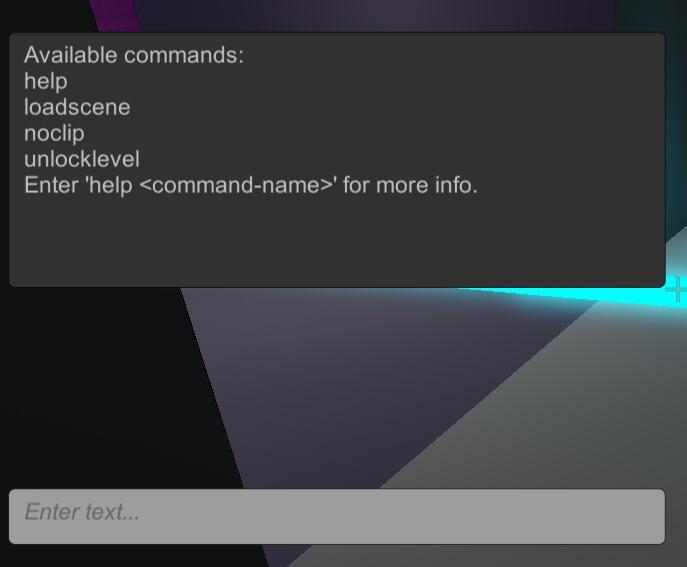
The second class is an abstract class PreloadAudio, the main body that allows ease of attachment. It contains methods to find out which audio clip is being used, and load the clip.

These two classes form the backbone of the music system. On the other hand, the frontline of the system consists of 5 classes. Each of these classes inherits from the PreloadAudio class and implements incomplete methods. They also contain a SoundManager object, which they reference to play the music. The purpose of these classes is to play different types of sound:

1. AddSoundToCollision: Plays sound when an object collides with another
2. AddSoundToTrigger: Plays sound when the player passes through it
3. BackgroundMusic: Plays background music
4. ClickSound: Plays sound when any button on menu screens are clicked.
5. PrepareSound: Play sound only when a certain condition occurs. This class is used when the player wins the level(thus winning sounds play) or when a certain puzzle is solved, etc.

Complete Music System.png

# Console - Prototype Feature

Implemented as a debugging and developer’s tool, the console will be a command line that will parse input and execute various commands. Some commands will output to the console.

* help: prints out a list of console commands you can  
   make others will load scenes or unlock other areas   
   for easy debugging or exploration.
* loadscene: loads the specified scene.
* noclip: allows the player to pass through objects,   
   including walls
* unlocklevel: unlocks the specified level in the hub.

The console has two main parts: the user interface, and the parser and executer.

* User interface
  + the command line, where the player enters the commands, and the output screen, where text is returned to the player.
* Parser and executer
  + takes the input given to it be the console, parses it, and executes the called commands.
  + When given an invalid command, it will send an error back to the console to output. Otherwise, it will execute the command, and send any relevant text back to the console to output (e.g. entering the help command will have the parser and executer output the list of possible commands to console)

Game Elements

# Universal elements

## Scoring rules

* + Goal State - There is one condition to winning a level: reach the goal, which is inaccessible until the level’s puzzle has been solved.

## Names

* + Element Zeta- a laser like light source that appears as a beam. Can be emitted, diffused, or redirected.
  + Dark solid- A material that is solid when in the presence of a shadow or an absence of light.
  + Light solid- A material that is solid in the presence of light, or the absence of shadow.

## Special powers

* + Diffusers- orb shaped objects that diffuse element zeta throughout an   
    area around it.
  + Emitters- cube that emits a beam of element zeta out into one direction.
  + Redirectors - cube that redirects a beam of element zeta 90 degrees.
  + Rotate objects - the player may rotate any of the objects they can hold..

## Levels

* + Baby Steps- a general tutorial level to introduce the player to the game’s mechanics.
  + The Tower- a vertical ascension-style level where the player has to navigate a tower to the top. It is the first level where the puzzles are more complex and require some thinking through.
  + The Music Hall- a level with musical influence, where the level’s background music changes as you progress through the level.
  + The Dark Room- a level inspired by an Escher painting. The level is dark, with no light sources except for the puzzle itself.
  + The Training Room- a level that provides a heightened level of challenge. It showcases a number of initial puzzle designs we developed during production of the game.
  + The First Trial- the first of three levels to test the player with the game's mechanics. It is a short puzzle and is simply to test the player without any guidance.
  + The Second Trial- the second of three levels designed to further challenge the player. This puzzle is again without guidelines and requires more intuitive thinking.
  + The Third Trial- the third and final level designed to push the player’s puzzle solving. This puzzle is short like the other two, but offers an extended logical challenge to the player.
  + The Chasm- the final level of the game. It offers a player closure to the game’s story and presents one final challenge to the player.

# Other elements

* + Shape objects - 3D shapes such as triangular prisms and cubes will be   
    located throughout levels for the player to move and rotate. These objects   
    can be used to cast shadows on light or dark solid materials to make them   
    appear or disappear.
  + Hub room - A room with doors. Initially one door will be unlocked leading   
    to level one. As each level is completed a new one will be unlocked.

Algorithms

## Deforming Blocks with Shadows

The core of Element Zeta's gameplay revolves around Light Solid and Dark Solid blocks which are either solid or not solid (the player can see and pass through it, but it retains it's shape and is not quite a gas). A naive implementation of this is to check if the light source (the diffuser) is being blocked by anything. This can be done using a simple raycast from the diffuser to the block. If the raycast returns that something is in the way, then the block must be in shadow; otherwise, the block must be in light.

However, what this doesn't account for is the scenario where something is casting a shadow on the block, so part of the block is in shadow and part of the block is in light. To handle this, we split the blocks into many smaller blocks, each doing the same check of raycasting to the diffuser. This allowed for the shape of the shadows cast to be approximated in a blocky, stylized manner. The smaller the blocks, the better the approximation of the shadow.

## Optimizing the Shadows

Each small block was a cube, so even if the cube was surrounded on all sides by other blocks, all six sides still had to be rendered, which was inefficient. This was also inefficient because rendering each block as a separate, small object meant that the parallel structure of the GPU (which is designed to render very large objects quickly) was not being utilized.

To fix this, we imported a voxel renderer into our project. A pixel is a point in a 2-D grid, whereas a voxel is a point in a 3-D grid. By representing each block as a voxel in a 3-D grid, we were able to preserve the same functionality we had before, but with the advantage that a grid of voxels can be rendered very efficiently.

The last hurdle for implementing the light reactive blocks was to optimize the raycasts. A 20x20x20 block would contain 8,000 voxels, which means that 8,000 raycasts would have to be performed to update the block. However, raycasts are not cheap operations, so this many would grind the game to a halt. The solution was to simply not do all of them at once. Each frame, a set number of voxels would be chosen to be updated, usually less than 1,000 at a time. Then the next frame, the next set of voxels would be updated, and so on. This means that the blocks have a small delay before they are fully up to date, but this delay is usually less than a second. The way the voxels update in chunks also adds to the stylized nature of the game, giving it an interesting, digital appearance, as if the game takes place in some sort of simulation.

Story Overview

# Character description

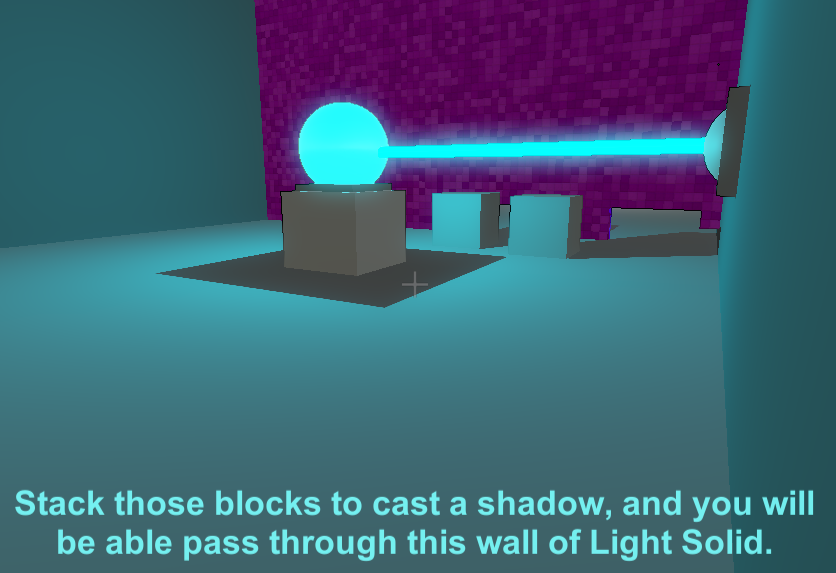
The player is a disembodied entity that casts no shadow. It is of no particular race or gender, rather it is something that can traverse through a room, and must walk on solid ground. The player may pick up and rotate moveable objects.

# Player goals and achievements

* Solve puzzle
* Get to goal spot
* Returns player to hub room
* Opens new door to new puzzle

# Scripts

As there are no enemies or NPCs in the game, to supplement a storyline narrative, there is in-game text dispersed throughout levels.

The narrative follows the player as they travel from level to level. The setting is known as “the facility” to the first narrator. The narrator instructs the player to take certain actions in certain ways; if these instructions are followed, all seems well at first, and the player progresses with little spoken opposition.

However, when you ignore the commands of the narrator, the narrative path shifts to a greater,more sinister conspiracy. The plot line pushes the player to question exactly what is going in this facility, what Element Zeta is, and what role the player is taking in the game. Depending on what choice you make, the ways you solve the puzzles, and who you choose to listen to will shape the narrative you receive.

This narration is managed with a trigger-based and timer-based text scripting system. When the player satisfies certain conditions, or simply just plays the game, these triggers can become active or inactive. Some simply output text, while others trigger a timed response that can either resolve once a certain amount of time passes, or if something isn’t completed in an allotted amount time.

# Game Progression

As the player progresses throughout the game, the game’s mechanics will be utilized to challenge the player in different ways; Initially there will be one level open via a door in the hub room. As the player completes each level, a new room will be unlocked.

The first level will introduce the player to the game’s main mechanics, and presents a set of easy puzzles, along with simple guidelines on how to progress. Meanwhile, the game is shaping around a narrative that centers around if you listen to guidelines as given or not. The illusion of choice is not obvious immediately, but slowly the game shows the importance of it.

The remaining levels in the game will provide additional challenges focused around the the mechanics in different ways, while the levels’ designs and themes are driven in part by the narrative. Nearing the end levels, there are no hints and more challenging logic in the puzzles. Finally, the level level is uniquely large and brings the narration to a close while presenting a final puzzle to solve.

Bibliography

CSE 5912 OSU Autumn 2016

Skalsky, Michal. *Volumetric Lights for Unity 5*. Computer software. N.p., n.d. Web. <<https://github.com/SlightlyMad/VolumetricLights>>.

Unity3d.com

Williams, David, and Matt Williams. *Cubiquity*. Computer software. *Volumes of Fun*. Vers. 1.2.1. N.p., n.d. Web. <<http://www.volumesoffun.com/>>.