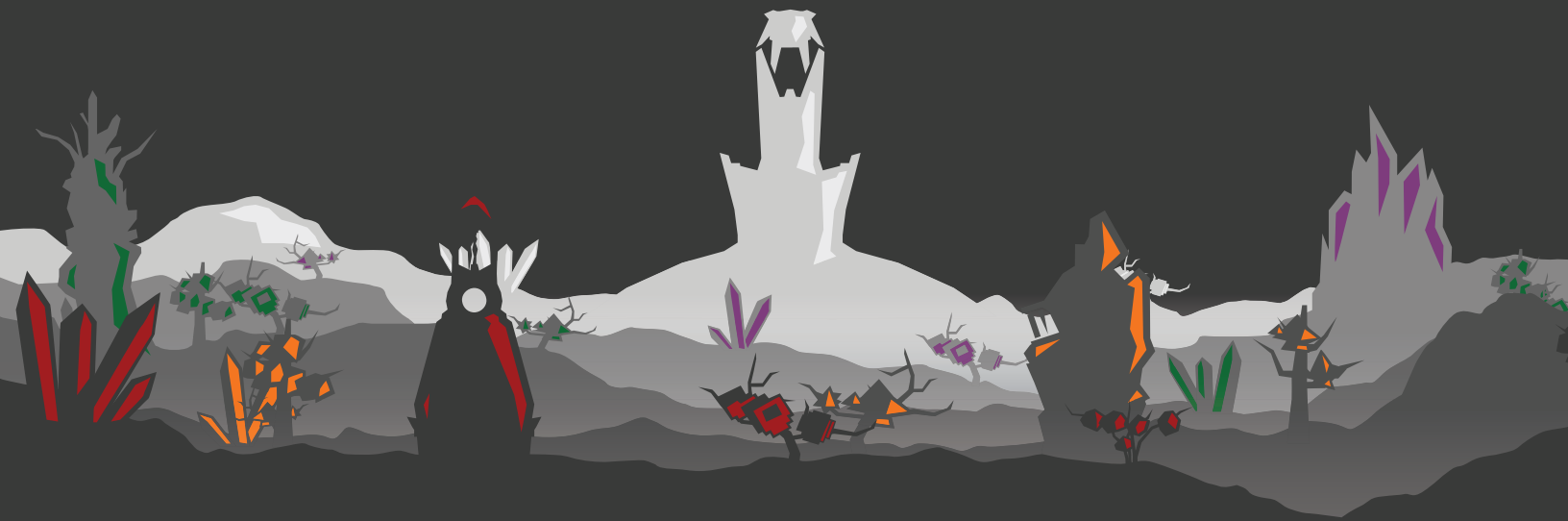




# ANOTHER END

## DESIGN DOCUMENT



IMD 4901A - SENIOR PROJECT  
SUPERVISOR - DR. ALI ARYA  
LAST MODIFIED - APR. 16 2015

---

BETHANY DUNFIELD MARCO BRITO MATTHEW FOURNIER  
RYAN BOTTRIELL ZARA TOOTH

# EXECUTIVE SUMMARY

Another End is a procedurally-generated, story-driven adventure game built for the desktop web browser. The world of Another End has five major non-player characters: Entity, the all powerful master of the world, and the four other glyphs, who are trying to usurp Entity and achieve dominance for themselves.

The two core player actions within Another End are exploration, and interaction. Players can explore an infinite procedurally generated world, with many different plants, objects, and areas to discover. Player interaction takes the form of using a variety of powers, which can have both local and world-changing effects. Four of the five characters have their own set of powers which are made available to the player through their choices and actions within the game. Players can use their powers to not only affect the physical world, such as by destroying a tree, but also to affect the influence levels of the glyphs, and help bring them either into or out of balance.

Another End features both a procedurally generated world and story. The environment is generated through a combination of random factors, including the fertility of the area, its temperature, and the “civilization” density of the area. Another End’s narrative is created through procedural linking of “story nodes”. Each node has a set of prerequisites which must be met, either through current game state variables, or environmental conditions.

Another End is intended to be played with the mouse and keyboard, but has an experimental version implementing a virtual reality display using the Oculus Rift and natural interaction using the Leap Motion controller.

The game makes use of a two-tiered architecture, encompassing both a local game client, and a remote web server. The local client, based in the desktop web browser, handles game engine functionality and rendering to the screen. The procedural generation has been offloaded to the remote server, running node.js, in order to save client resources. Objects and story nodes are stored within an Elasticsearch index, and are used by the server to generate content appropriate to the current game state.

The visual style of the game is heavily geometric and “clean”, but not without depth and texture. A variety of colour palettes have been developed for the game. The in-game heads up display, and menu systems have been designed to match this visual style. The only exceptions to this style are the in-game hands which appear when players are using the Leap Motion Controller. Music for Another End is being composed by Daniel Campoli, a student in the music department at Carleton University.

The game was completed on April 10, 2015, when it was shown at the 2015 IMD Senior Project fair.



# TABLE OF CONTENTS

1 INTRODUCTION	4
2 SPECIFICATIONS	5
2.1 GAMEPLAY	5
2.2 VISUAL STYLE	5
3 SCHEDULE	6
3.1 SUMMARY OF COMPLETED WORK	6
3.2 MILESTONE SCHEDULE	7
4 TECHNICAL REQUIREMENTS	10
4.1 HARDWARE	10
4.2 SOFTWARE	10
5 RISK ANALYSIS	11
5.1 RISK FACTORS	11
5.2 PREVENTIVE ACTIONS	12
5.3 CORRECTIVE ACTIONS	13
6 PROJECT MANAGEMENT	14
6.1 TEAM STRUCTURE	14
6.2 VERSION CONTROL, TEAM MANAGEMENT TOOLS	15
6.3 TEAM COMMUNICATIONS	15
6.4 QUALITY CONTROL	16
7 FURTHER READING	17
8 REFERENCES	18



# 1 GAME DESIGN

Another End is a procedurally generated adventure game, where the player's actions and choices change their experience. This section will discuss the complete game design for Another End.

## 1.1 WORLD DESIGN

### 1.1.1 BACKSTORY

This section summarizes the backstory for Another End. A more detailed explanation of the story and lore influences can be found in Appendix A.

Originally all that existed in the world was Entity, an all powerful being with the ability to create, destroy, protect, and perform great illusions. Entity grew bored with the world of nothingness. After building a tower to live in, Entity then created four glyphs to span out and help build the world. To create the four glyphs Entity divided its powers and assigned a power to each glyph. Architect was assigned to add new objects to the world, Protector was assigned to insure the longevity of the world, Illusionist was assigned to create space and time, and Destroyer was assigned to remove old objects from the world.

The leadup to the game story arc involves Destroyer becoming dissatisfied with its role, as there were no objects to be removed from the new world. Destroyer confronts Entity, and is reassured that in time, Destroyer will have purpose. However, Destroyer lacks the patience to wait, and begins destroying objects in the world, sparking a conflict among the glyphs. This conflict escalates through a series of events leading to the glyphs separating, and locking themselves away in their own towers. Without the power of the Glyphs, the world begins to decay.

The player appears as an unknown presence within the world. The glyphs, aside from Entity, have fallen dormant. Entity is initially wary of the new arrival, but soon comes to realize he can use the player's power to lock the Glyphs. It is up to the player to decide whether to take sides with Entity or one or more of the four other glyphs.



### 1.1.2 WORLD LAYOUT

The world in which Another End takes place will be made up of trees, plants, rocks, and a variety of ruins. In this world will also be “ruins” which are old buildings and random man made objects that the player can discover and explore. Different areas of land, or “biomes”, will be procedurally defined. Biomes are marked by differences in temperature, fertility, terrain height, civilization etc. These values are further explained in section 4.3.2 Environment Generator.

In the world, the five non-player characters (NPCs) will each have their own tower. Entity’s tower is in the middle of the world surrounded by the four glyphs’ towers. The glyphs’ towers are quite a distance away and approximately in each cardinal direction. These towers will be randomly placed each time the game starts and the world will be procedurally generated between them. See Figure 1 below for a diagram of the World Layout.

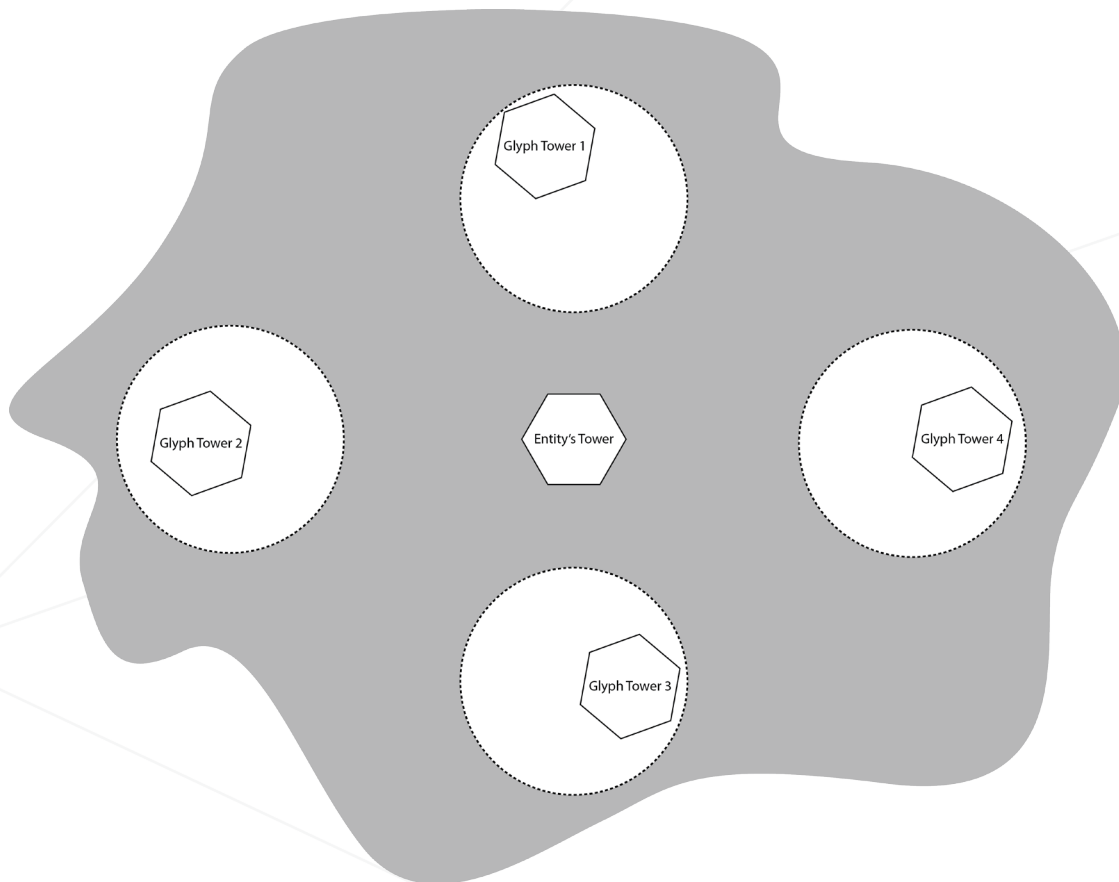


Figure 1: World Layout, Grey area is procedural world, dotted circle is area in which one glyph tower will be generated.

### 1.1.3 PROCEDURAL GENERATION

One of the core goals of Another End is to ensure that every experience a player has with the game is unique. To accomplish this, many aspects of Another End are procedurally generated to various degrees. The game's story is generated by procedurally linking story nodes. These nodes will be chosen based on the player's progression and choices throughout the game. The environment of the game is also procedurally generated. The objects which populate the world are a mix between pre-modelled and procedurally assembled composites of pre-modelled meshes. Object placement is procedural-based on the environmental variables at any given point. Further detail of the procedural generation process can be found in Section 4.3.

## 1.2 GAMEPLAY

### 1.2.1 MECHANICS

Players are free to explore the infinite world of Another End. They are given access to a variety of special powers which allow them to manipulate and interact with the world. These powers are used to complete story events in the world, which affects the influence of each glyph in the world. Higher levels of glyph influence gives the player access to stronger powers related to that Glyph. Altering the influence levels of each glyph in the world is intended to become the players overarching goal. With this in mind, the gameplay revolves around using the given powers to accomplish this. The powers and energy system are described in detail in section 1.2.3.

### 1.2.2 CORE GAME LOOP

The core game loop, seen in Figure 2, is a simple loop which provides an easy to understand set of gameplay tasks for the player. One of the goals of the story and world generation systems is to ensure that the player is provided with new and interesting opportunities to apply this core loop. The loop has three distinct stages: explore world, use powers, and alter glyph influences. Although other mechanics exist outside of this loop, it is understood that the story and gameplay will inevitably return to this iterative loop.

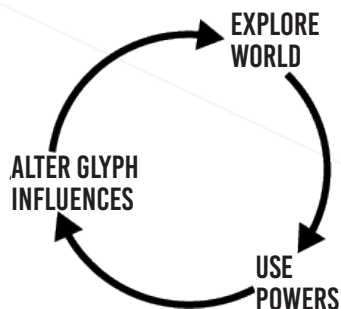


Figure 2: Core Game Loop

### 1.2.3 POWER SYSTEM

Throughout playing the game there are sixteen different powers that the player can unlock. The powers are divided by level and by associated glyph. A full list of powers can be found in Appendix B.

The level of the powers are rated from 1 to 4, level 1 being the lowest (base power) and level 4 being highest. The four powers in level 1 are all available from the start of the game, with each power being associated with one of the four Glyphs. Upon using a power, a short 'cooldown' is experienced, and powers cannot be used until the completion of this cooldown.

To unlock additional powers the player must visit towers in the game which are inhabited by one of the four glyphs. While visiting the glyph's tower, the player is presented with the choice to either lock the glyph into Entity, or set it free. If players choose to reconnect the glyph to entity, the glyph's influence is locked at the balanced level, and the player is given permanent access to the level 2 power associated with that Glyph, with no opportunity to unlock any further powers for that character. If the glyph is set free, the player will immediately unlock the level 2 power associated with the Glyph, and be given the opportunity to unlock the higher level powers. level 3 and 4 powers are unlocked by raising that Glyph's influence in the world.

## 1.3 STORY DESIGN

### 1.3.1 STORY SYSTEM OVERVIEW

The narrative of Another End is procedurally generated. The story of the game is made up of procedurally linked "nodes". Each node can be thought of as a story event. Rather than design a linear story, the entire "story world" of Another End has been designed, to allow for a more diverse experience [1]. Nodes make use of game state variables, including characters and their current influence levels, nearby environmental objects or biomes, and powers currently available to the player. Figure 3 shows a graphical representation of what data is stored within a story node.

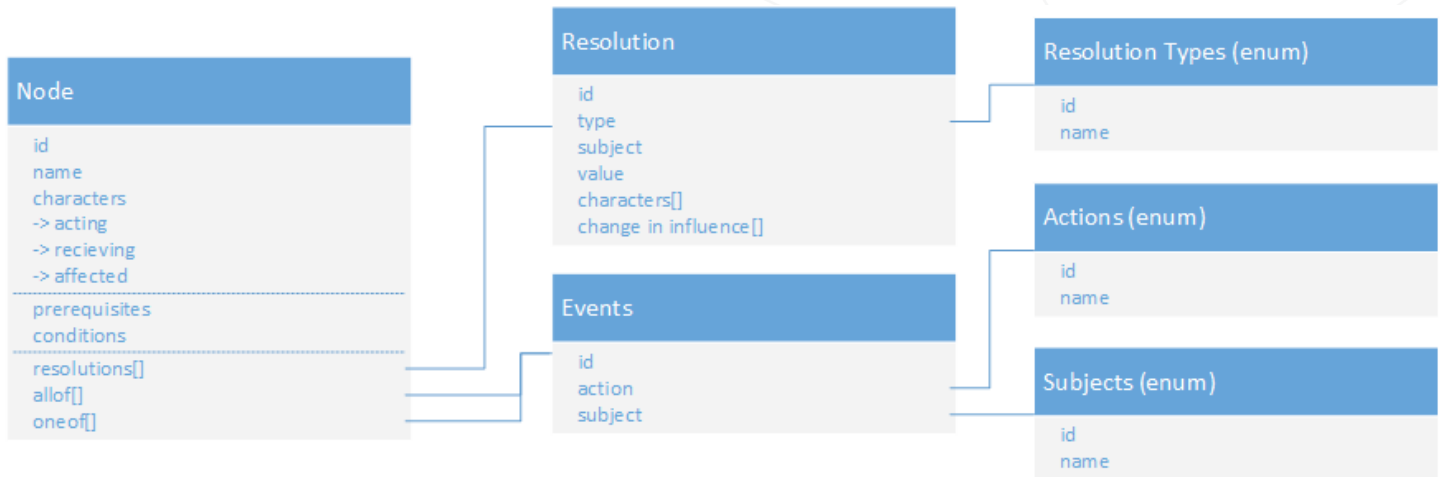


Figure 3: Story node database structure

### 1.3.2 CORE STORY ARC

The core story leading up to Another End is discussed in section 1.1.1. Though the story experienced by the player is procedural, there is a core story world that has been defined. Players are dropped into the world with little direction, however characters can provide motivation for the player to move forward. Entity, for example, asks the player to “lock-in” Glyphs upon reaching their tower. The Glyphs themselves, on the other hand, directly ask the player to visit their tower and set them free. Upon reaching the glyph’s tower, players may choose to either reconnect them or set them free. This choice, repeated for each of the glyphs, creates the core story arc of the game. The game ends, should players choose to reconnect all four of the glyphs back to Entity, with the conclusion of that story line or the complete dominance of another glyph.

## 1.4 PLAYER MINDSET

While the player is inside the world of Another End, they will experience a state of curiosity and uncertainty. The game’s infinite world and open-ended story will give players the feeling of curiosity which motivates them to explore and continue their experience within the world. Players will be emotionally invested in the characters, and will experience tension through their conflicts. By involving players within the relationships of Entity and the Glyphs, the player will gain a sense of control or influence on the game world. However, the conflicting choices presented to them by the characters within the game will cause players to be unsure of their actions.



## 2 INTERFACE AND CONTROLS

### 2.1 MENU SYSTEM

Figure 4 shows a flowchart for the menu system for Another End. Figure 5 and 6 shows the final visual designs of the menu system. In addition to usual menu items such as settings and a link to the external game website, there are three unique sections to Another End's pause menu. First, the map page shows what is loaded around the players position, giving insight into the procedural terrain generation system. This map graphic is also displayed as the initial world loads in real time. Similarly, the story page provides a navigable UI in which the player can explore his/her unique story timeline and re-visit completed story nodes. Finally, the powers page provides descriptions and icons for each of the 16 powers available in the game.

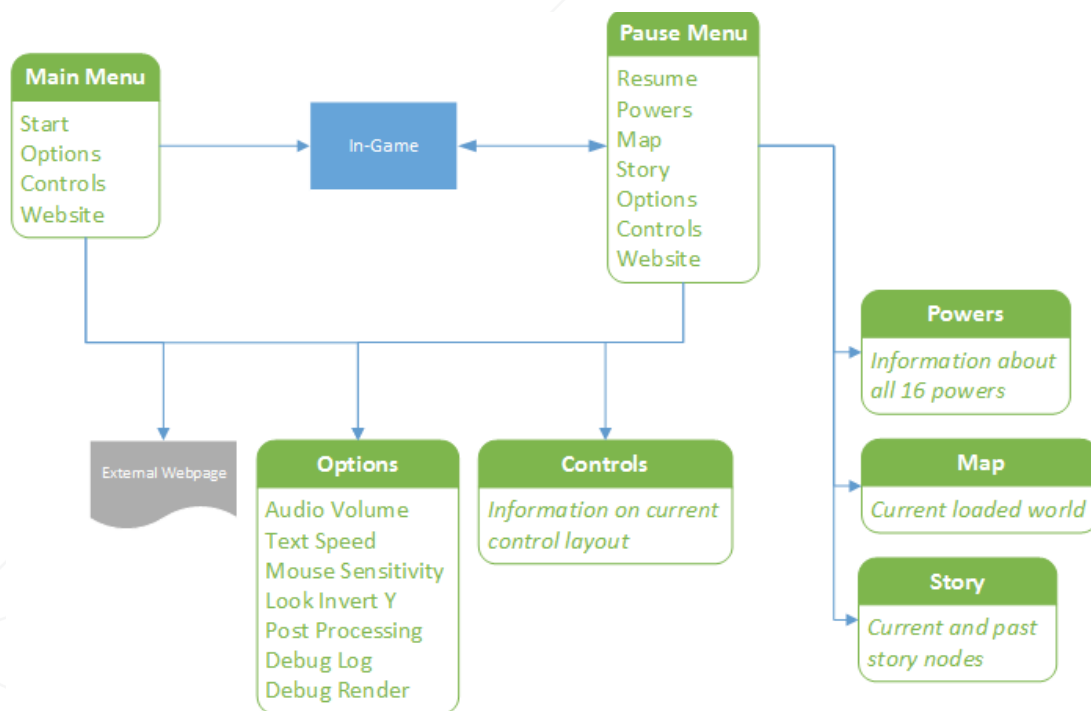


Figure 4: Flow chart of Another End's menu systems

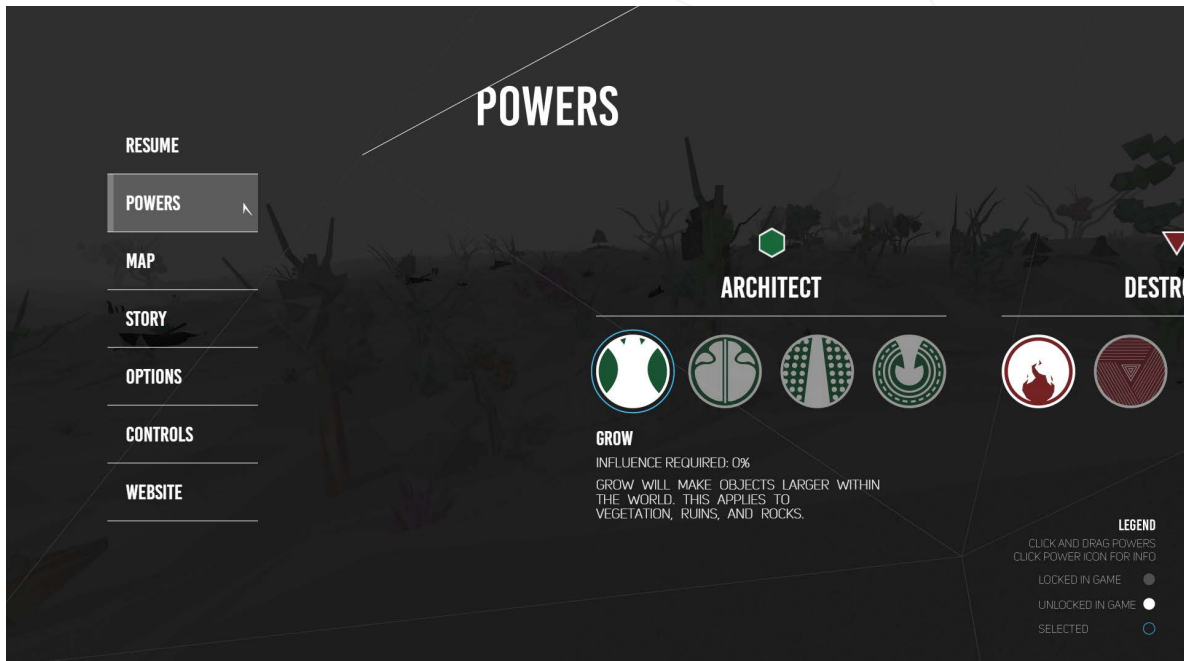


Figure 5: Pause Menu Design



Figure 6: Start Menu Design

## 2.2 IN-GAME HEADS-UP DISPLAY

In the game there will be a Heads-Up Display (HUD) that shows the player key information. On the HUD there will be a horizontal compass bar indicating the direction of the glyph towers and active story nodes, an area displaying the influence levels of each glyph, and an area showing information on the players available powers and energy. More information can be found in section 5.3.2.



Figure 7: Screenshot of Another End's Heads-Up Display

## 2.3 CONTROLS

Another End is mainly designed for the mouse and keyboard.

The game has an experimental input mode which makes use of the Oculus Rift VR and a Leap Motion. The Oculus Rift VR is the main viewing area as well as the input for moving the view around in the world. To limit head movement the player only needs to look 60 degrees either way and the world will continue to pan. The Leap motion will be the primary input for the game, using easy gestures to navigate the world and perform tasks. See Table 1 below for gestures and their actions in the game. Another End will support using the VR Mount for the Oculus Rift in order to use both technologies together [2].




GESTURE	ACTION	IMAGE
Fist	To move around in the world player makes a fist and moves it forward, back, right or left (this action works in conjunction with the Oculus HMD for full motion control).	
Palm Out	The player puts their palm out (like the stop gesture) for power use. For powers that involve aiming the player closes their fingers together to aim and then opens to shoot.	
Open Palm Up	To access the powers menu player turns one palm facing upward.	

Figure 7: Screenshot of Another End's Heads-Up Display

## 3 SYSTEM REQUIREMENTS

### 3.1 CLIENT

Browser with WebGL support

- Internet Explorer 11+
- Chrome 33+
- Firefox 37+
- Safari 8+
- Opera 26+

Hardware

- 1GB dedicated video memory
- Core i5 CPU or equivalent
- 4GB RAM

Internet Connection

- Broadband or better

For Oculus VR use, Another End requires a VR-enabled Chromium build and an Oculus Rift HMD. For Leap Motion controller use, Another end requires a Leap Motion Controller and Leap runtime software. For use together, Another End requires the Leap Motion VR Mount for the Oculus Rift.

### 3.2 SERVER

Unix System running Ubuntu 14.0+

Software

- Node.js v0.11.14-pre or higher with Socket.io v0.9.16 or higher
- ElasticSearch v1.4.2

Hardware

- 1 GB RAM
- 20 GB SSD
- 2.0-3.0 GHz CPU



# 4 DEVELOPMENT

## 4.1 SUMMARY

The game architecture consists of a two-tiered system. The game itself runs on a local client. To save client resources, procedural generation is offloaded to a remote web-server. Many game objects and story nodes have properties stored in a database which is hosted on the same web server.

## 4.2 CLIENT

### 4.2.1 ENGINE

Another End is built in a custom engine designed specifically to handle procedurally generated content. The engine is written in HTML, CSS and JavaScript to be served as a web page through a web browser. The following is a breakdown of the responsibilities of the engine:

#### SERVER INTERFACE

The engine manages a connection to the server-side of our game and requests, receives and properly processes data moving between them. This includes terrain and object data as well as story information and game state information.

#### WORLD AND SCENE MANAGEMENT

Due to the infinite nature of our procedural world, the engine must organize the data that it receives from the server and then load and unload content as the player moves around. The engine organizes the world into an infinite grid, and keeps the squares around the player loaded at all times while unloading squares that move to far away. The engine also loads farther grid squares to lower level detail to save on GPU memory.

#### POWERS

The engine also manages the powers system that allows the player to interact with the world. It must deal with throwing, creating and resolving the effects of powers on the environment. Much of the functionality of the power system, such as power selection, is closely related to the interface and HUD systems.



## STORY

Although the server defines story nodes and their relationships, the engine must receive the nodes from the server and manage them properly on the client side. Story nodes need to be tracked, and the engine must respond when the nodes completion conditions are met. The engine is also consistently updating the story server with information about the game state so that the server can properly resolve and generate new story nodes.

## PARTICLES

The game's particle system brings life into the static world of Another End. The particle system, which is used for the glyph "essences" within the world, is based on boids functionality and is managed by the engine. The engine creates and updates the particle system with information about story nodes and important objects.

## INTERFACE & HUD

The menu system and in-game HUD are also managed by the engine. In order to support VR displays, all menus and HUD components are displayed on billboards within the 3D environment. The engine manages the placement and visibility of all of these components as well as the interaction with them. The engine must also calculate mouse positions and project that into the 3D world so that the user can properly use the menu systems.

## INPUT / OUTPUT DEVICES

The engine is also responsible for interfacing with the Oculus VR and Leap Motion controller. The Oculus system requires special rendering and also provides head movement data which is used in the game to allow the player to look around. The Leap Motion controller provides skeletal information for the players hands. The engine uses this data to render the characters hands in the game world as well as for interfacing with menus and HUD.

### 4.2.2 RENDERING & SHADERS

Another End is rendered in WebGL through Three.js, an open-source JavaScript library which simplifies WebGL rendering in the browser [3]. WebGL allows us to take full advantage of the graphics hardware on the client machine by giving direct access. This greatly increases the rendering capabilities in the web browser.

Another End uses several custom shaders to render properly in WebGL. The shaders are written in GLSL and implemented through Three.js:

#### MAIN OBJECT SHADER

The main object shader is used for all geometry that makes up the generated world in After End. This shader renders objects using vertex colors, and also incorporates all lights in the scene. It is also responsible for rendering the fog. The fog is rendered based on a distance sphere around the camera. However, because the fog only exists up to a certain height, the shader also accounts for occlusion while looking up, out of the fog as well as looking over and into the fog from above.

#### SKYBOX SHADER

The skybox shader is used only to render the geometry of the game's skybox. The skybox edges sit at a fixed distance away from the player and are rendered using a gradient from bottom to top. The skybox shader calculates the proper color of the gradient at any height. Because the skybox is a cube, but the sky should render as a sphere, the skybox shader also corrects for this while rendering the gradient, creating the illusion of a perfect sphere. Using the same methods as the main object shader, the skybox shader also renders the world's fog onto the skybox.

#### POST SHADER

The post shader is a pixel / fragment shader that provides post processing effects to the game's render. After the geometry and fog are rendered, the post shader applies 'filters' to the rendered image to create the final look for the game world. The post shader uses a depth pass and beauty pass in its calculations. This shader deals mostly with color correction and tinting but also provides functionality for fading the screen to and from black.

The rendering of Another End follows a distinct process. First, a depth pass is rendered to a separate render target using a special Three.js depth material. The skybox is then rendered with its shader as the background. The geometry is rendered on top of the skybox using the main scene shader. Following this the post processing is applied to the rendered world and then the interface and HUD are rendered over top. This process is visualized below in Figure 8.



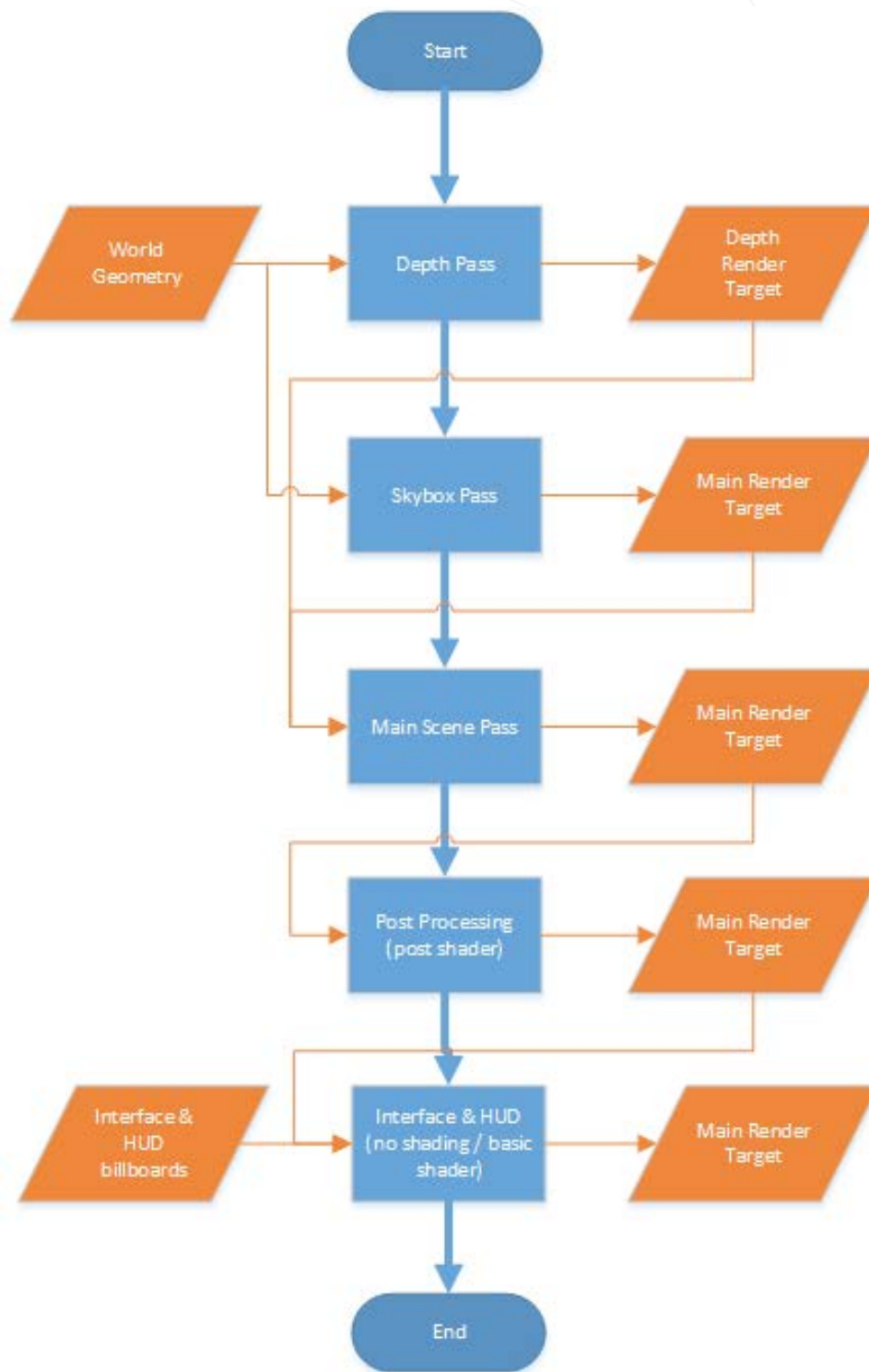


Figure 8: Another End Rendering Pipeline

## 4.3 SERVER

The game server runs on node.js, and communicates to the client using websockets and the Socket.IO library. The server handles both communication with the client and the procedural generation of the game. To that end, there are three major classes. Generation is accomplished by both the Story Generator and the Environment Generator. Communication between the two generators is accomplished by passing events between the two class objects. The main Server class acts as the communicator between the two generators and the client, sending and receiving client events as they happen, and ensuring client events reach the proper generator. Figure 9 illustrates the relationship of the server-side game files. Game objects used for generation, including story nodes and environmental objects, are stored within a Database hosted on the server.

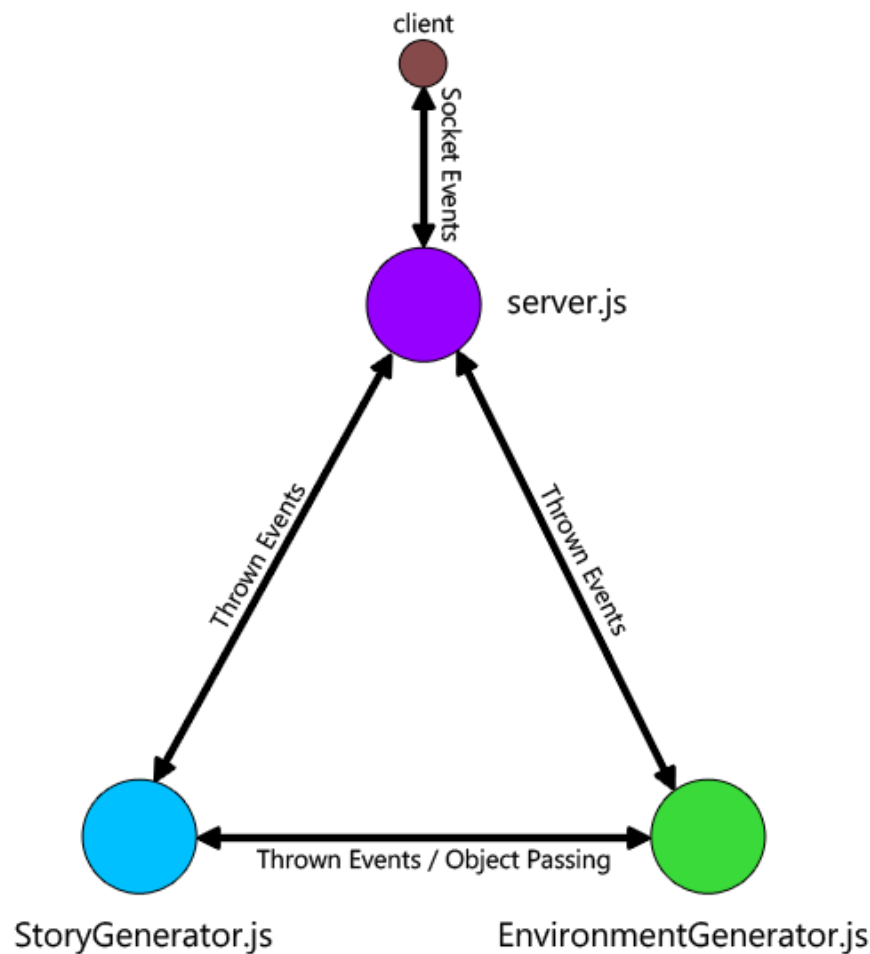


Figure 9: Server-side file relationships for Another End

#### 4.3.1 STORY GENERATOR

The Story Generator handles the flow of the game's story as perceived by the player. This involves assignment, monitoring, and resolution of story nodes as the player progresses. The Generator decides when new story nodes are necessary within the game. When a new node is required, the generator evaluates the current game state, including player progression, character influence levels, and the position of the player and the associated environmental variables. Story nodes that match the current game state are selected from the database, and from those results a node is selected. If no nodes are found, the generator will either wait to try again, or attempt to generate another node for a different area of the game world.

Within the story node, the event(s) that must occur to resolve the node are stored. When a node is loaded by the generator, it activates the event listener(s) associated with these events, and / or sends a message to the client so that the appropriate client-side event listener(s) can be activated. Upon receiving the event(s), the generator can resolve the node as per its definition. This may include changing game state variables such as character influence levels or player power availability. After the node has been resolved, the cycle repeats until the player either reaches an end node or leaves the game.

#### 4.3.2 ENVIRONMENT GENERATOR

The Environment Generator handles creating / constructing the physical world of the game. This involves creating the actual terrain mesh, which is passed to the client, and determining which objects will fill the world. Unlike the terrain, objects are not passed from the server to the client, rather the server tells the client which object to load and where to load it.

Environmental Characteristics are defined in two ways. For the purposes of this document, "Nature" in terms of Environment will refer to the procedurally generated aspects of the environment. Environmental characteristics in Another End are either defined by "Nature", or by what Glyph's territory they occur in. Table 2 gives a summary of the various environmental variables and what they define.

Defined By	Characteristic	Description
Nature	Terrain Height	Physical Y coordinate of the terrain. Terrain Height contributes to the temperature variable (higher heights have lower temperature).
	Fertility	Defines the amount of plants growing in a given area. Higher fertility gives more and larger plants such as trees. Lower fertility gives fewer plants and more rocks.
	Temperature	Defines the type of plants that can grow.
	Civilization	Defines the density of ruins and civilized (non-organic) objects.
Glyph Territory	Colour	Defines the colour palette to be used in an area, with smooth transitions between areas.
	Colour	Defines the general colour and tint of objects within the biome, overriding the “Nature” colour variable.

Table 2: Summary of Environment Generation variables and their effects.

Naturally defined variables are created through Perlin noise maps. The combination of factors that these noise maps represent gives the final environmental state and appearance at any given position. Environmental objects stored in the database include the acceptable range of the various environmental factors that must exist for them to be valid. For example, a type of tree may only be valid when Fertility is between 0.6 and 1.0, and temperature is between 0.3 and 0.8. When the Environment Generator decides to create an object, it will query the database to find objects whose properties match the environment at that location.

For Terrain generation, the Terrain Height noise map has a direct connection to the physical Y-coordinate height of the vertices of the ground plane. The other factors, such as temperature and fertility, are used to generate the vertex colour for that specific area.

The second aspect of Environment Generation is procedurally assembling objects. Hierarchically modelled assets have multiple entries within the database with the parameters as to how they may be assembled. For example, a tree trunk database entry will specify which branch and leaf types may be used when assembling the tree. The server will use these parameters to determine which files should be loaded, and the appropriate arrangement to create the entire object.

### 4.3.3 DATABASE

The game's database is powered by ElasticSearch, a “NoSQL” document store with a robust search algorithm [4]. ElasticSearch indexes JSON documents, and organizes them by type. When comparing to MySQL, the “index” of a document corresponds to its “Database”, and the “type” of a document corresponds to its “Table”. This allows for increased flexibility when storing objects for Another End, as ElasticSearch easily allows for Arrays and Objects to be stored within an Object Document. Additionally, as both the client and server of the game use Javascript, the JSON format of ElasticSearch documents is ideal.

Communication between the Server and the Database is done by using the ElasticSearch node.js module to create a client instance on the server. This client can make a variety of queries, and has the functionality to receive callbacks and errors.

ElasticSearch can only be accessed through code-based clients. Therefore, to facilitate database management, an in-house developed GUI is used for performing database functions. The GUI has functionality such as the ability to view Database Documents, the ability to edit or delete Documents, and the Ability to create new Documents and Document types.

# 5 VISUAL DESIGN

## 5.1 STYLE OVERVIEW

The visual style for Another End is very geometric and the feel of the game will be a surreal world. The game will not attempt to be overly realistic, having trees and plants abstracted in both design and colour. The visual aesthetics of the game will be simple, but still containing depth and visual appeal.

The world will be brought to life through both vibrant colour palettes, and the addition of grain for texture. The world is split into five regions each having distinct colours associated to them depending on the character inhabiting the region. Each region will feature slightly different characteristics, which reflect which character's domain the area represents. For example, the sky and the ground within each region will be coloured based on the representative colour of the dominant character in that region.

The Glyphs themselves take their inspiration from their respective spirit animals. The in-game representations of the Glyphs will be heavily geometricized, with solid fills for each of their bodies and faces.

## 5.2 WORLD STYLE

### 5.2.1 GLYPHS

Entity, and each glyph, each have a distinct colour, basic geometric shape, 3D structure, and spirit symbol. Table 3 summarizes these characteristics. Figure 10 shows the characteristics of the Glyphs.

CHARACTER	BASIC SHAPE	3D STRUCTURE	COLOUR	SPIRIT
Entity	Circle	Sphere	White	God
Destroyer	Triangle	Pyramid	Red	Wolverine
Protector	Square	Cube	Purple	Turtle
Illusionist	Rhombus	Diamond	Orange	Raven
Architect	Hexagon	Hexagonal Prism	Green	Spider

Table 3: Character Characteristic Summary

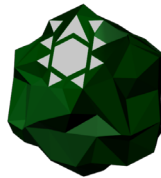
DESTROYER



PROTECTOR



ARCHITECT



ILLUSIONIST



ENTITY



Figure 10: Glyph Designs

### 5.2.2 GLYPH ESSENCES

Throughout the world, smaller representations of the glyphs can be found and interacted with by the player. These are known as “essences”, and behave as floating shapes. Individual essences mimic the original glyph’s basic geometric shape and 3D structure. The colour of the original glyph is also shared with its essence counterparts. The essences are the main point of interaction between the player and the story.

### 5.2.3 TOWERS

Each character within the game has a unique tower which acts as its home and center of power. Every tower has a unique aesthetic, which is designed around the power, personality, and visual appearance of the matching glyph. The below figures illustrate the tower designs for each of the in-game characters.



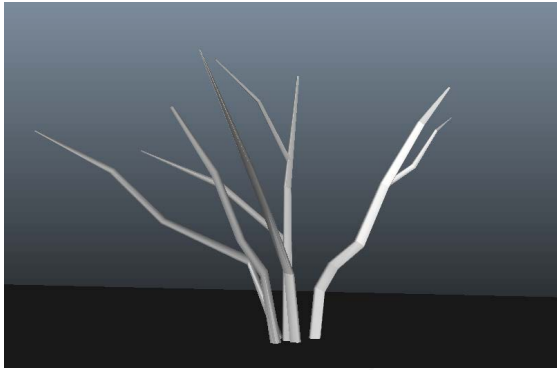
Figure 11: Glyph Towers



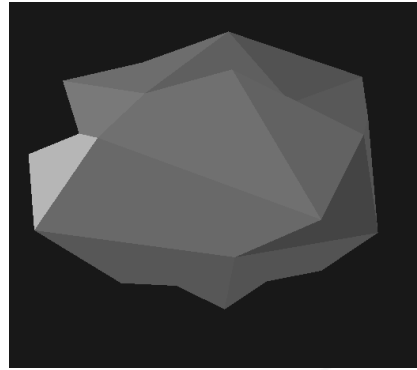
### 5.2.4 ENVIRONMENT

The style of Another End's environment and the objects which populate it is very geometric and abstract. Objects such as bushes and trees are tessellated but still recognizable. Objects within the world will have distinct colouring which does not necessarily correlate with its appearance in the real world. For example, the leaves of a tree may be blue instead of green. Figure 12 shows an example of the environmental appearance of the world. Appendix C has all the different environment models.

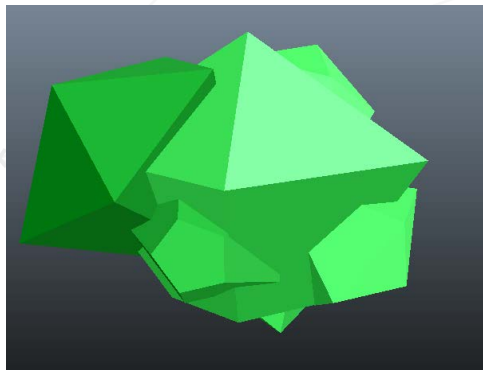
TREE



ROCK



LEAVES



CRYSTALS

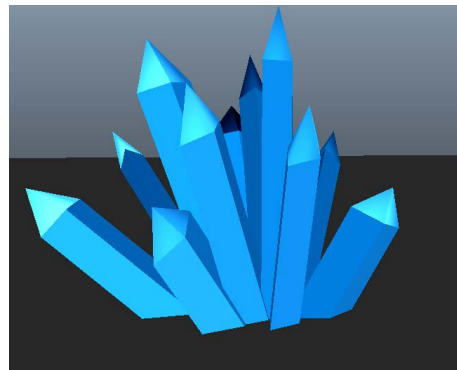


Figure 12: Environment models

### 5.2.4 POWERS

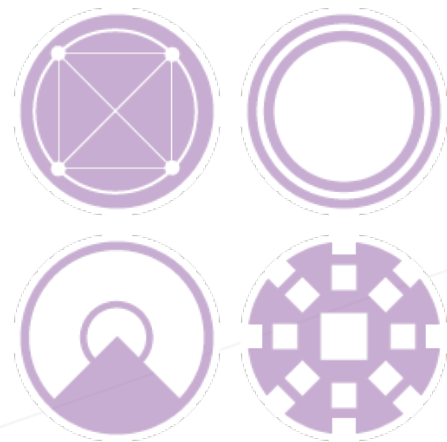
Every power in the game will have a unique visual design along with a 2D icon to represent it. Figure 13, shows the power icons in Another End. As an example, one of the Destroyer powers has red triangles falling from the sky to mimic a rain of fire. The specific colour of the glyph will be associated with it. All of the powers have a different look.

Another End has been designed to be played with both the Leap Motion and Oculus Rift VR. When using the Leap Motion, a set of virtual hand models will be represented in the game as the player's hands.

#### DESTROYER'S POWERS



#### PROTECTOR'S POWERS



#### ARCHITECT'S POWERS



#### ILLUSIONIST'S POWERS

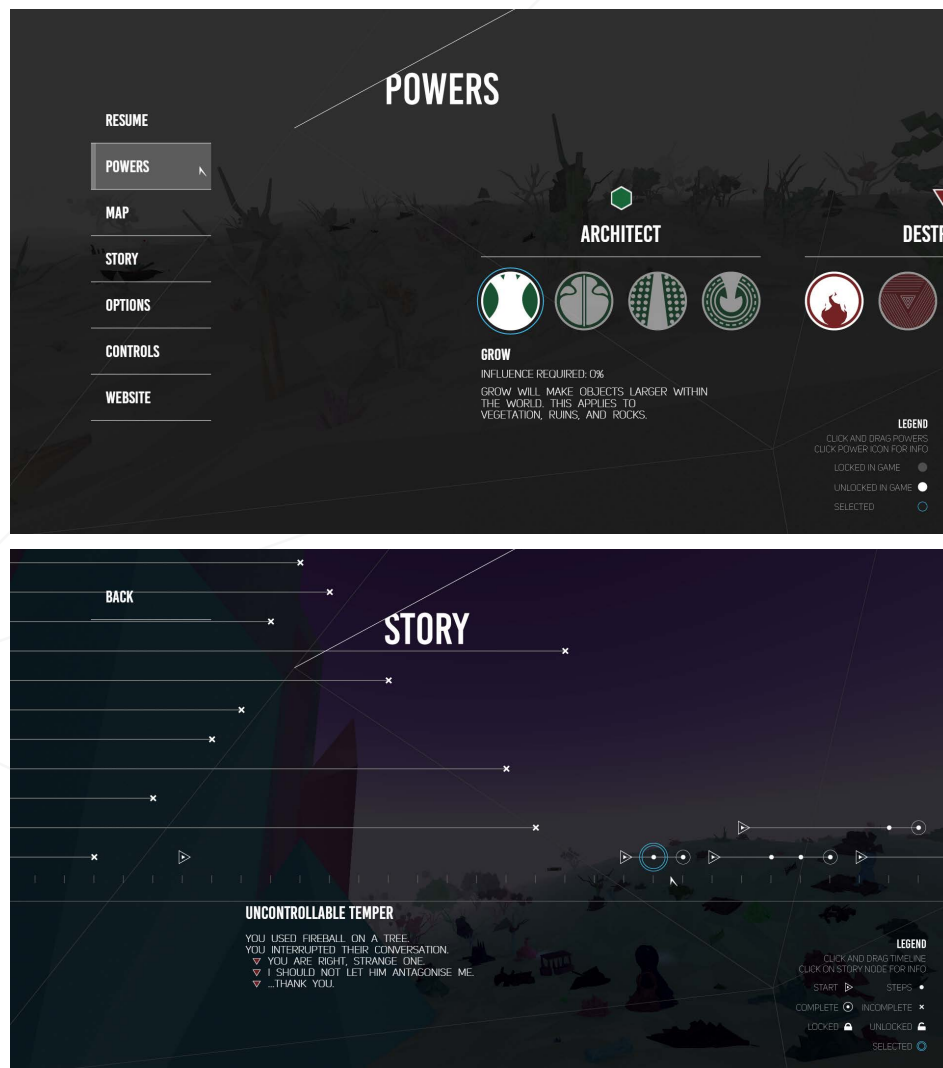


Figure 13: 2D Power Icons

## 5.3 INTERFACE DESIGN

### 5.3.1 MENU STYLE

Another End's menu design is simple and clean for easy navigation. The menus contain options on the left side, the content in the middle, and a legend in the bottom right corner. Most of the elements are white content on a black background. The selected or key content is highlighted with the colour blue. The background of the initial start menu is different than the in-game pause menu. In the start menu the background shows the world generating as the player waits for the game to load. The pause menu has the a dark screen with the game still visible as the player moves through the menu. Figure 14 shows Another End's menus.



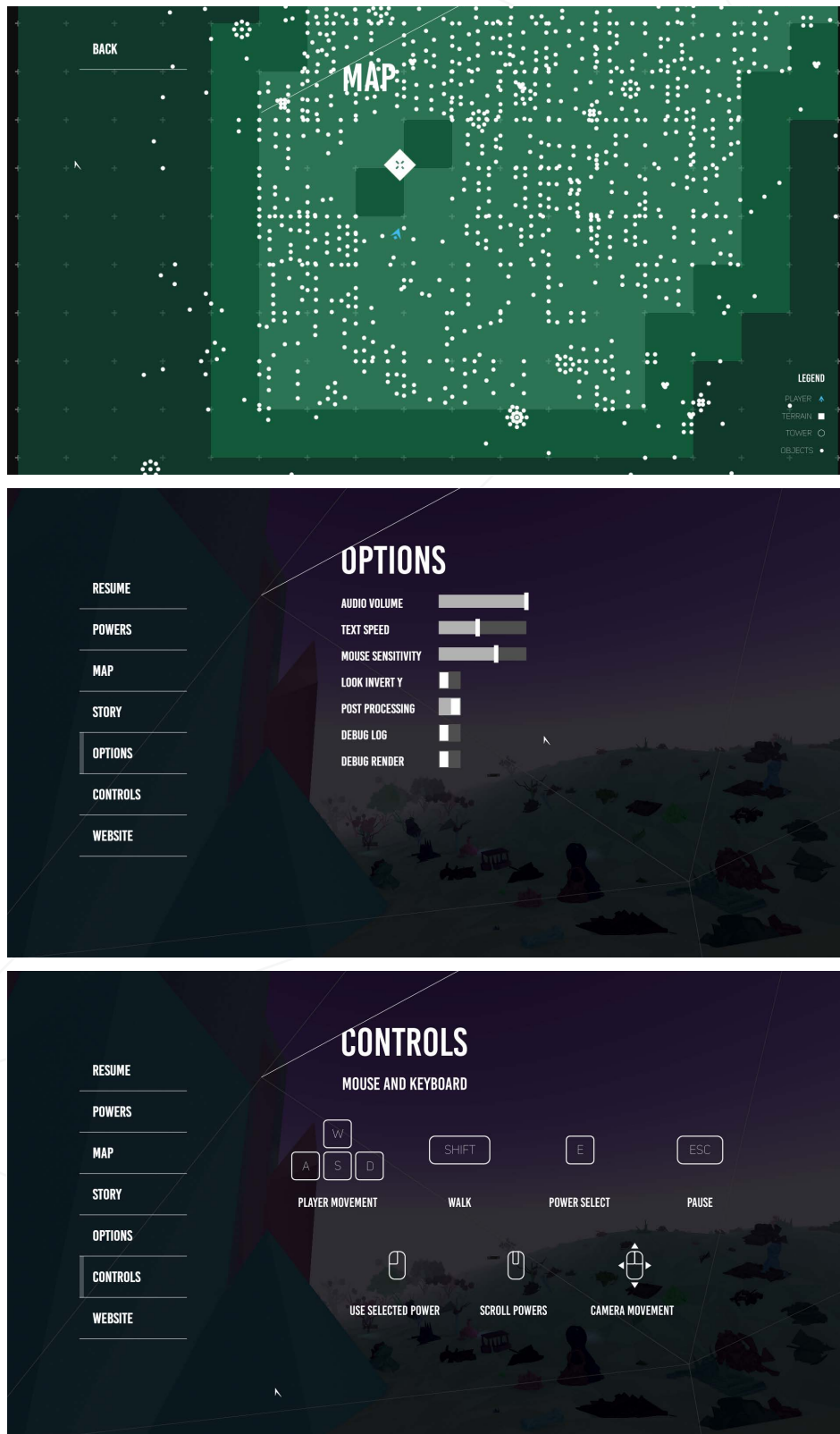


Figure 14: Pause Menu Design

### 5.3.1 HEADS-UP DISPLAY DESIGN

The Heads-Up Display (HUD) system has a similar design to the menu system. Figure 15 shows the HUD for Another End. The HUD is made up of 4 parts: Selected Powers, Character Influence Levels, Aiming Reticle, and navigational compass. The Selected Power, influence meters, and compass all have a transparent geometric background to help distinguish them from the game screen with white and the colours of the glyphs to represent different information to the player. The reticle will change its shape to represent which glyph's power set is currently selected by the user.



Figure 15: HUD Design

### SELECTED POWER SECTION

The Selected Powers section of the HUD, seen in Figure 16, contains information about the power system. This section contains the power currently available to the user. The icon background colour will change depending on which glyph power set the player has equipped. The selected power name is displayed beside this icon. The bar under the power name represents the cool down time between each instance that the player can use the power.

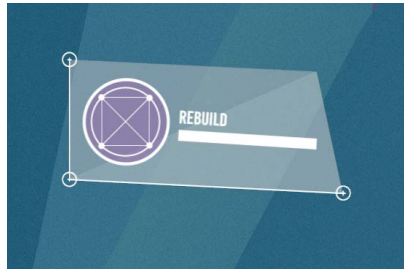


Figure 16: HUD Design

### CHARACTER INFLUENCE METERS

This section of the HUD contains information about the current influence levels of the in-game characters. Four meters are visible, each corresponding to a different glyph. The meters have one line, indicating the balanced level. The influence bars travel between -1 to +1 as the player explores through Another End. The red line signifies that the glyph is still locked and the player must travel to its tower to lock or unlock it.

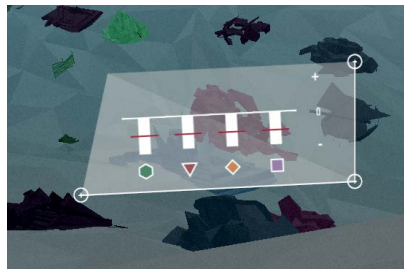


Figure 17: HUD Design

## 5.4 COLOUR SCHEME

The colour schemes in Another End break up the colour wheel into 6 sections, red, orange, yellow, green, blue, and purple. There are separate colour schemes for each of the Glyph areas in Another End. The palettes are then broken up into three land colours and five objects colours that are analogous to the section of the colour wheel selected. The sky colours are broken down into two options. Each option has a sky gradient and a fog colour that can be chosen by the procedural system.

Figure 18 shows the Destroyer colour palette and the sky palette for the Destroyer area. Other colour palettes for the game can be viewed in Appendix D.

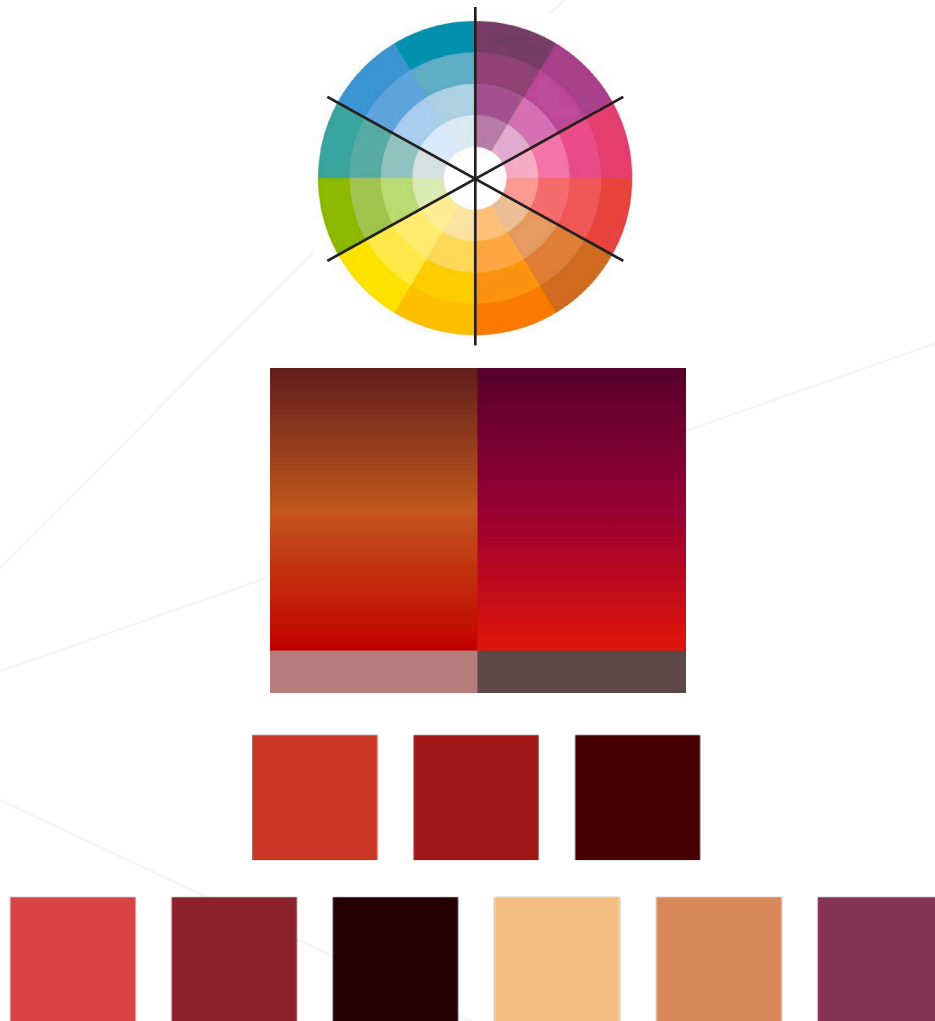


Figure 18: Colour Wheel and Destroyer Land and Sky Palettes



## 5.5 ASSET CUSTOMIZATION

The assets for the game will have varying levels of customization. The five bushes and nine rocks have a small level of customization; they can be scaled uniformly to be small or large, giving variation to the landscape. The trees in the world have a high level of customization. Each of the four tree types have their own set of customization rules. Each Leaf then has a custom placement, scale, and rotation based on which tree it is paired with. An example of one of the tree rules can be found in Appendix E.

## 5.6 ART STYLE INFLUENCES

The geometry of the world is heavily inspired from Cubism. There are three aspects of Cubism that inspired the visual style of Another End, the grittiness, the colours, and the multiple perspectives. Some modern day examples of inspiration came from video games such as Kentucky Route Zero [5] and Eidolon [6], and the interactive film “Rome” [7]. Another large influence for the style is the music video for the song “Delta” by C2C [8]. Their use of mixing of gradients, flat colours, and simple noise inspired the sharp edged objects with vertex colouring in Another End.

The colour palettes for each glyph and the biomes of the world were influenced by the colours seen in the game No Man’s Sky [9], an upcoming video game. Eidolon, mentioned above, is an example of how a lack of environmental movement makes the game appear flat and lifeless. Eidolon has an extremely flat style, with no movement, and few textures or gradients [6]. This reverse inspired the style by being the opposite of the desired look. Figure 19 has examples of art influences.

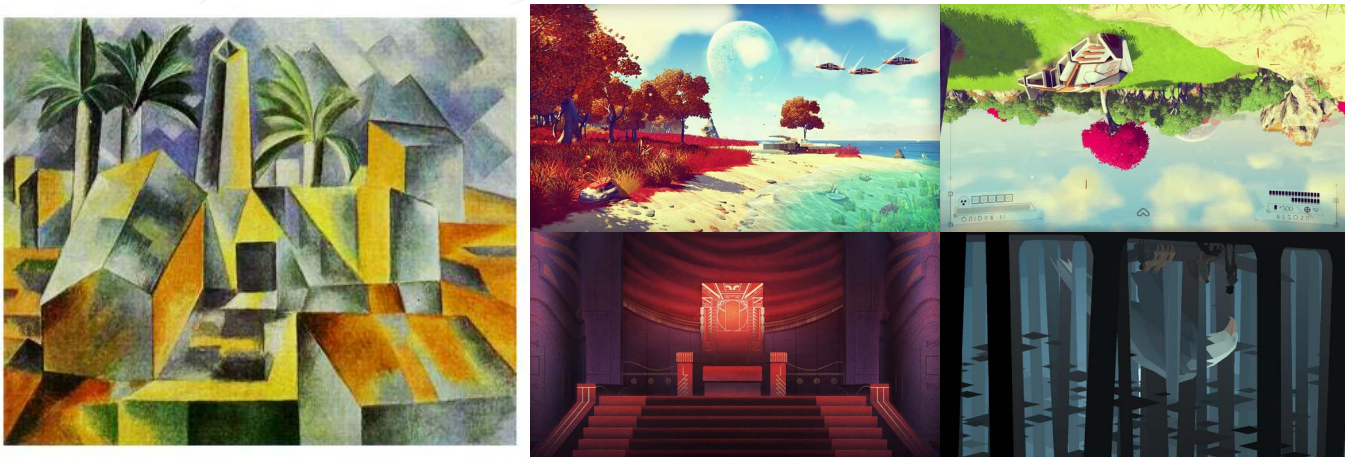


Figure 19: Art Style Influences



## 6 SOUND DESIGN

Another End will makes of a variety of music and sound effects. Music was produced by Daniel Campoli, a student in the music department at Carleton University. Consistent communication and feedback was given to Daniel to ensure the created music suited the atmosphere of the game.

Created music includes a theme roughly 25-45 seconds in length for each character, as well as a short cue roughly 7-15 seconds for each character. The short cues are played when the player completes a story node that increases the influence of that character. Additionally, two themes unrelated to the characters were produced for the use of advertising and the title screen of the game. In addition to music, sound effects were created for actions, characters, and objects in the game.

NAME	CATEGORY	FILE NAME
Destroyer Cue	Music	cue_destroyer.mp3
Destroyer Theme	Music	theme_destroyer.mp3
Architect Cue	Music	cue_architect.mp3
Architect Theme	Music	theme_architect.mp3
Illusionist Cue	Music	cue_illusionist.mp3
Illusionist Theme	Music	theme_illusionist.mp3
Protector Cue	Music	cue_protector.mp3
Protector Theme	Music	theme_protector.mp3
Fireball	SFX	Power_Fireball.mp3
Void	SFX	Power_Void.mp3
Annihilation	SFX	Power_Annihilation.mp3
Black Hole	SFX	Power_BlackHole.mp3
Grow	SFX	Power_Grow.mp3
Duplicate	SFX	Power_Duplicate.mp3
Fertilize	SFX	Power_Fertilize.mp3
Consolidate	SFX	Power_Consolidate.mp3
Morph	SFX	Power_Morph.mp3
Pull	SFX	Power_Pull.mp3
Warp	SFX	Power_Warp.mp3
Multi-Morph	SFX	Power_MultiMorph.mp3
Rebuild	SFX	Power_Rebuild.mp3
Shield	SFX	Power_Shield.mp3



Rejuvenate	SFX	Power_Rejuvenate.mp3
Shield Army	SFX	Power_ShieldArmy.mp3
Falling Powers	SFX	Power_Falling.mp3
Shooting Power	SFX	powershoot.mp3
No Impact	SFX	power_miss.mp3
Menu Click	SFX	menu_click.mp3

Table 4: Sounds in Game



# 7 REFERENCES

- [1] C. Crawford, *On Interactive Storytelling*, 2nd ed. New Riders, 2012.
- [2] J. Newman, *Leap Motion reveals bid to become Oculus Rift's head-mounted VR controller*. PC World, [online] 2014.  
<http://www.pcworld.com/article/2600203/leap-motion-reveals-bid-to-become-oculus-rifts-head-mounted-vr-controller.html>
- [3] Features - *Three.js*. GitHub, 2012, [online]  
<http://github.com/mrdoob/three.js/wiki/Features>
- [4] Elasticsearch, *What is Elasticsearch?*, 2014, [online]  
<http://www.elasticsearch.org/overview/elasticsearch>
- [5] Cardboard Computer, *Kentucky Route Zero*. [Video Game] 2013.  
<http://kentuckyroutezero.com/>
- [6] Ice Water Games, *Eidolon*. 2014. [Video Game]  
<http://www.icewatergames.com/about>
- [7] C. Milk, Rome - *3 Dreams of Black*. [Interactive Film] 2011.  
<http://www.ro.me/>
- [8] C2C, *Delta*. [Music Video] 2013.  
<https://www.youtube.com/watch?v=PZbkF-15ObM>
- [9] Hello Games, *No Man's Sky*. [Video Game] 2014.  
<http://www.no-mans-sky.com/>
- [10] Ubisoft, *Child of Light*. [Video Game] 2014.  
<http://childoflight.ubi.com/col/en-CA/home/index.aspx>

# APPENDIX A: LORE / STORY INFLUENCES

## CULTURES CHOSEN:

North American / Aboriginal: For the characteristics of the glyphs.

Aztec: For the creation story of the game world.

## INFLUENCES:

## ILLUSIONIST:

<http://www.native-languages.org/northwest-raven.htm>

<http://www.native-languages.org/legends-raven.htm>

<http://spirit-animals.com/raven/>

### Influence:

Trickery is influenced by the Native American raven. The raven has a great amount of power and the potential for good. Unfortunately he is irresponsible and enjoys playing tricks. The raven has performed great acts of heroism but can not be trusted due to his characteristics of greed, gluttony, and impatience. Is able to create at will and is able to bend space and time.

## PROTECTOR:

<http://www.universeofsymbolism.com/turtle-symbolism.html>

<http://www.native-languages.org/legends-turtle.htm>

<http://www.whats-your-sign.com/animal-symbolism-turtle.html>

### Legend:

Protector is influenced by the Native American turtle. The turtle is said to of helped create the world by transporting land mass on its back as he swam across the oceans. The turtle is also a great protector and will defend itself from foe be retreating into its shell. Turtle has tons of patience and is therefore never in a hurry. He will take his time while doing tasks. Is often known to have the wisest soul.

## DESTROYER:

<http://www.native-languages.org/loks.htm>  
<http://www.native-languages.org/lusifree.htm>  
[www.whats-your-sign.com/symbolic-wolverine-meaning.html](http://www.whats-your-sign.com/symbolic-wolverine-meaning.html)

**Influence:**

The influence of Chaos comes from the Native American wolverine. The wolverine is often thought of as a deadly monster with the ability to steal a person's soul. Has a translation of "Indian-Devil". Wolverine has an unprecedented amount of raw power but its power may be unpredictable.

**ARCHITECT:**

[www.universeofsymbolism.com/symbolic-spider-meaning.html](http://www.universeofsymbolism.com/symbolic-spider-meaning.html)  
<http://www.whats-your-sign.com/spider-symbol-meaning.html>  
<http://www.spiritanimal.info/spider-spirit-animal/>  
[www.shamanicjourney.com/article/6017/spider-power-animal-symbol-of-creation-weaving-our-realities-infinity-balance-past-present-and-future](http://www.shamanicjourney.com/article/6017/spider-power-animal-symbol-of-creation-weaving-our-realities-infinity-balance-past-present-and-future)

**Influence:**

The Architect is influenced by the Native American spider. The spider is extremely creative and can manifest magnificent designs. The spider's web has limitless potential of what it can be woven into. Spider also has patience as it knows that a beautiful design must not be rushed. With a body shaped like an eight and with eight legs, symbolism for infinity is all around the Spider.

**LORE:**

Many, many, moons and suns ago there was nothing in the world but Entity. The all powerful being with the capability to create worlds. Entity became bored with the world of nothingness and decided that it would build itself a home. With its powers Entity built a magnificent tower that rose hundreds of feet high. Entity then placed itself at the peak of the tower.

Entity was pleased with its work, but as it looked out beyond its tower into the world of nothingness it realized that there was more to be created. With such a large world to create, Entity gave life to four legendary glyphs in order to help create the world. Trickery, Protector, Destroyer, and Architect were born. Entity divided its power and gave part of it to each of the four glyphs. The glyphs were sent off to build the world that Entity had dreamed of.



The four legendary glyphs worked tirelessly creating a magnificent world. A world so beautiful that words can not define its beauty. However, one glyphs began to feel excluded and eventually grew angry at the other three glyphs and Entity.

Destroyer approached Entity and spoke. "Trickery is creating and manipulating time and space itself, Architect is creating magnificent designs and bringing them to life, and Protector nourishes these creations to insure their prosperity."

"As was intended"

"What am I to do with only the power to destroy"

"You will wait. If the world is to stay in perfect balance all things that come into existence they must also leave."

Destroyer did as told, and waited and waited. Many days went by but yet the world had nothing to be removed. Then one day Destroyer could no longer remain patient and set fire to a forest. Protector became angry at Destroyer for destroying a forest that he was assigned to protect. Architect then grew angry at Protector for not protecting a forest that he promised to protect.

Protector and Architect then asked Trickery if he could transform the burnt forest back to life. Trickery thought that he would be funny and brought all the trees back but as stone.

As time passed more events arose that caused conflict. Eventually all four of the legendary glyphs grew angry at each other and went off in their own directions to create a world of their own. The glyphs each created a tower in their own part of the world to exclude themselves from the others and entity. The glyphs then hid in their towers, locking away their powers given by Entity.

Years went by without the glyphs leaving their towers. The magnificent world that was created began to rot away until all that was left was the four towers belonging to the beings, and the one tower belonging to Entity.

As Entity had given most of its powers to the four glyphs, it no longer had the ability to create and repair the world. In a last attempt to recreate the world of its dreams, Entity created a being with the last of its power. This being was was capable of using all of the powers given to the original four glyphs. The new being could create change like Trickery, create new life like Architect, protect all of creations like Protector, and destroy any obstacle like Destroyer. Lastly, Entity created a suit of armor to be worn by the new being. The armor allowed for constant communication with

Entity and provided protection. The new being was then sent to reconnect the four original glyphs and rebuild the new world.

This is where the players journey begins

Glyph	Colour	Description / Goal	Interaction / Objective
Entity	White	Strives to bring balance to the other Glyphs (all bars are flattened and equal)	
Architect	Green	Strives to create	Create, Influence Weather, Alter Terrain Positively
Destroyer	Red	Strives to destroy	Destroy, Fire, Alter Terrain Negatively
Protector	Purple	Strives to gain knowledge and preserve that which already is	Research landmarks and ruins, Upgrading objects, Build guardian / shield tower, Reveal Trickster tricks (at high levels without research)
Illusionist	Orange	Strives to falsify information / change the truth to suit it	Create “copies” of objects that aren’t real, Falsify research information, Change nature of objects for the trickster, Alter events already in progress ie. change rain storm to a fire storm

# APPENDIX B: GAMEPLAY POWERS

Glyph	Power	Enum	Description	Fancy Description
Destroyer	Fireball	d1_fireball	Damage one object	Fireball is the basic attack for the destroyer. This power damages objects and ruins in the world.
	Void	d2_void	Remove one object	Void will completely remove objects from the world. Voids effects cannot be reversed.
	Annihilation	d3_annihilation	Destroy all objects in a radius	Annihilation rains fire from the sky and damages anything it touches.
	Black Hole	d4_blackhole	Remove all objects in a radius	Create a small but powerful black hole which sucks in all objects, removing them forever.



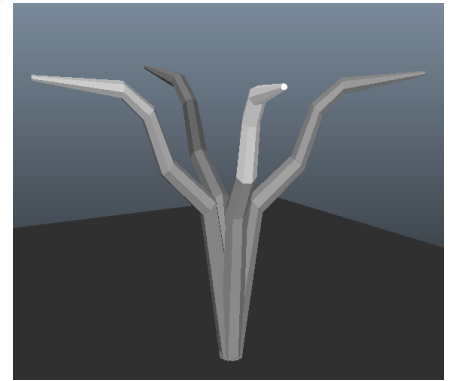
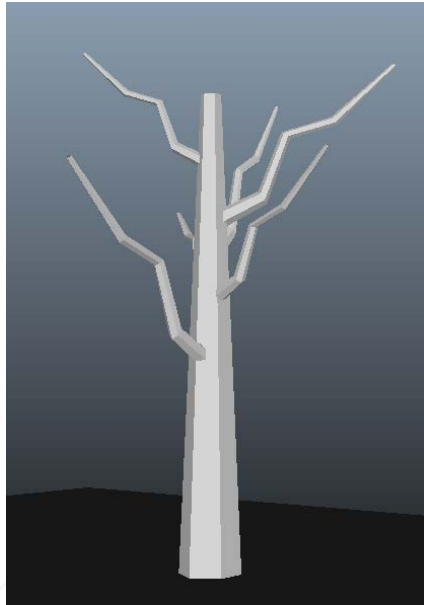
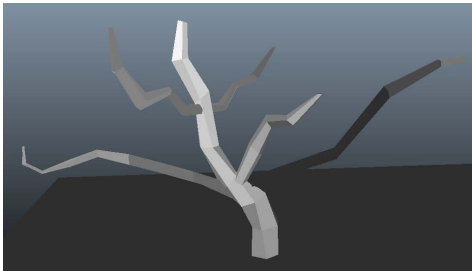
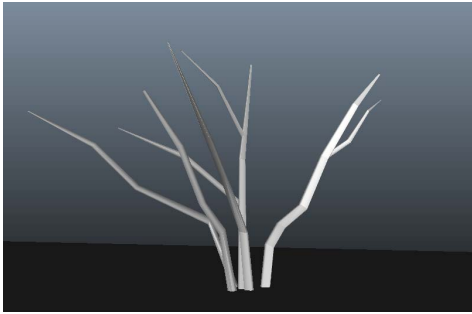
Architect	Grow	a1_grow	Scale up an object	Grow will make objects larger within the world. This applies to vegetation, ruins, and rocks.
	Duplicate	a2_duplicate	Sample an object and create a duplicate	Choose an object to duplicate and then place copies of it around the world.
	Fertilize	a3_fertilize	Scale up many objects in a radius	Creates fertilizing rain that falls from the sky and grows anything that it touches.
	Consolidate	a4_consolidate	Suck up nearby objects and create a mega object	The targeted object absorbs all objects around it, using their energy to grow enormous.
Protector	Rebuild	p1_rebuild	Undamage an object	This power rejuvenates damaged objects.
	Shield	p2_shield	Shield.	Shield will prevent the target object from being harmed or affected by other powers for a short period of time.
	Rejuvenate	p3_rejuvenate	Undamage objects in a radius	Restores life to all damaged objects around you.
	Shield Army	p4_shieldarmy	Shield objects in a radius	Shield Army will protects all objects that are near from other powers for a period of time.

Illusionist	Morph	i1_morph	Change an object to a random other object	Morph changes the targeted object into another.
	Pull	i2_pull	Move an object towards you	Pull allows for the relocation of an object by pulling it towards you.
	Warp	i3_warp	Teleport to where the power lands	Warp to the target location.
	Multi-Morph	i4_multimorph	Change objects in a radius to random other objects	Morph all objects in an area around you, changing their shape

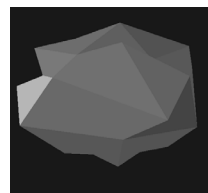
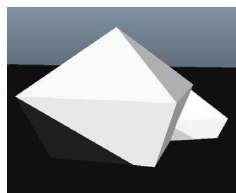
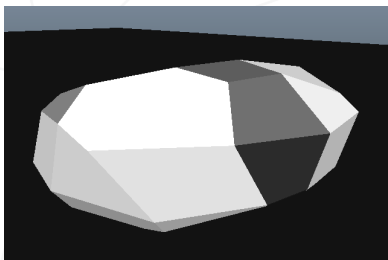
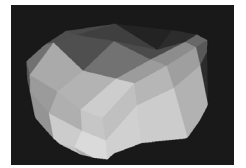
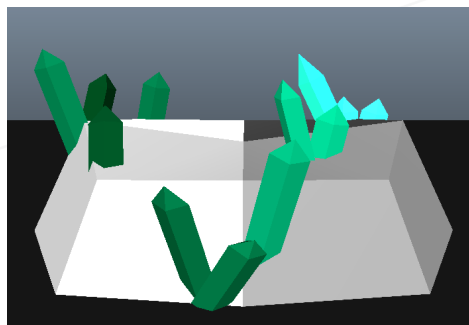
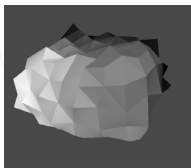


# APPENDIX C: ANOTHER END ENVIRONMENT

TREE

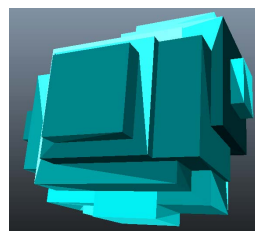
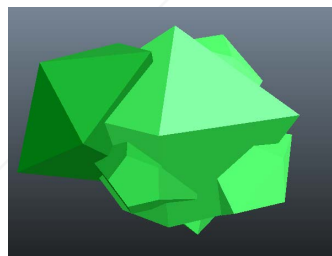
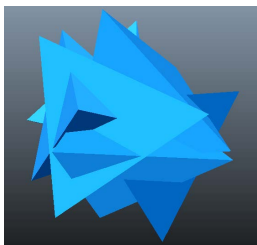
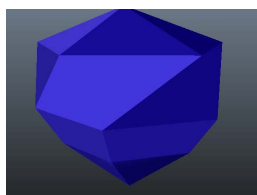
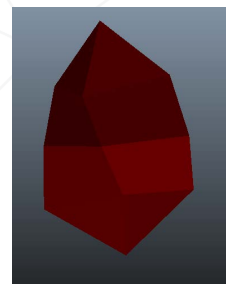
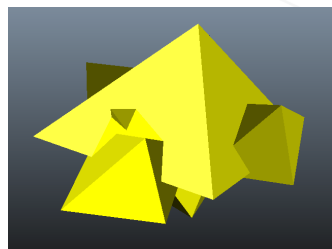
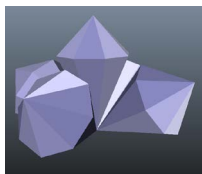
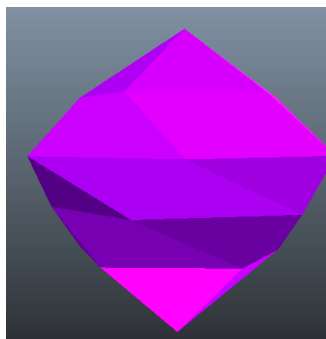


ROCK

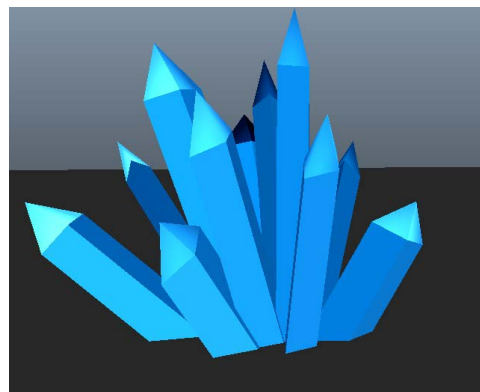
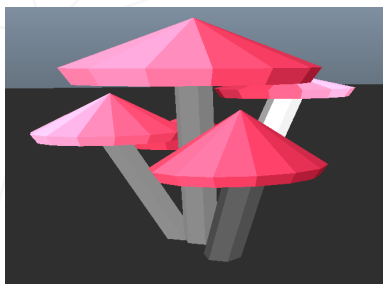
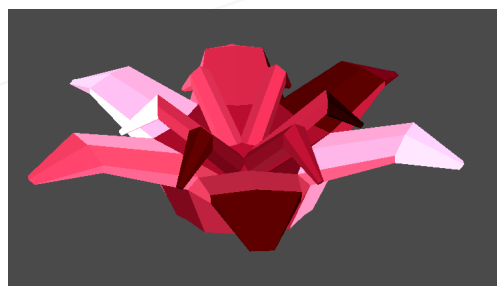
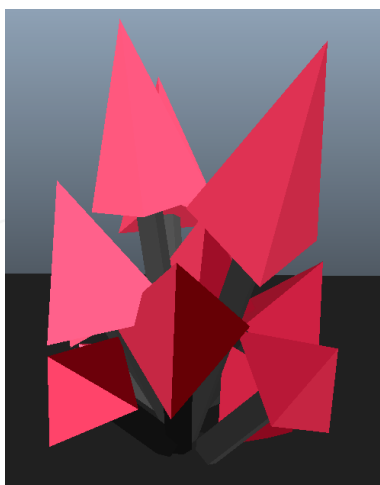
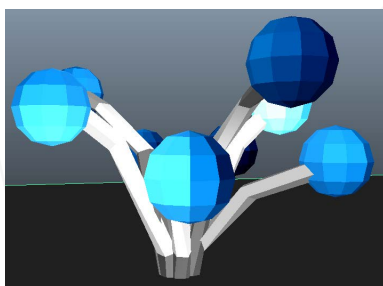




## LEAF



## BUSH





# APPENDIX D: COLOUR PALETTES

## DESTROYER



## PROTECTOR



## ILLUSIONIST

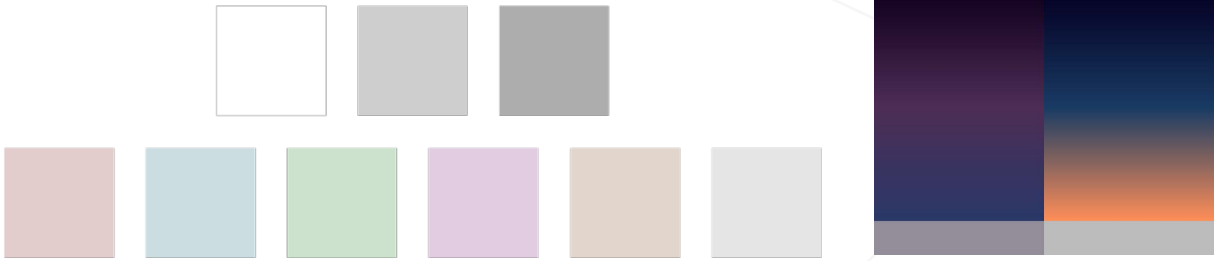


## ARCHITECT





ENTITY



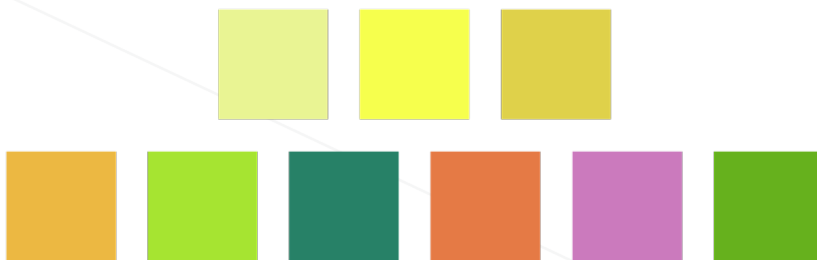
RED



ORANGE



YELLOW





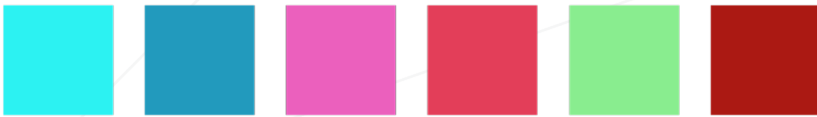
GREEN



BLUE



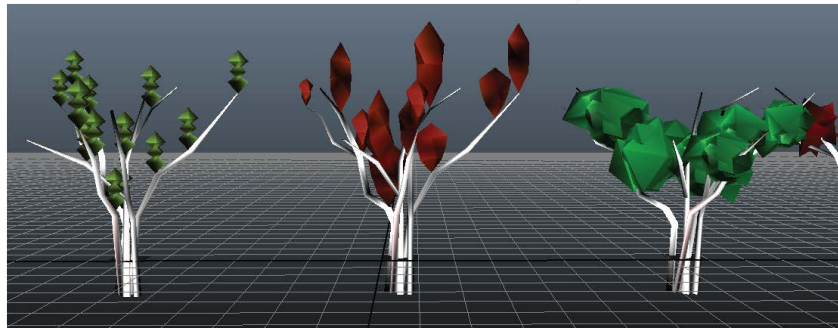
PURPLE



# APPENDIX E: TREE CUSTOMIZATION EXAMPLE

Below is an example of the customization rules for one of the trees in Another End. For all tree customization rules visit: [http://anotherend.com/wiki/index.php/Asset\\_Format](http://anotherend.com/wiki/index.php/Asset_Format)

Note: Colours of tree and leaves are not final.



## TRUNK

- Has 5 trunks, can use 3-5 of them
- All trunks must appear within a 1x1 meter square on the ground
- Each trunk can be rotated between 0-360 degrees around the Y axis

NOTE: if they appear all in the same spot they would be pointing in different directions, meaning they could all have a rotation of 0 and wouldn't look weird (although tested pointing them in all the same direction and didn't look bad either)

## LEAVES

- Leaves 1,2
  - Scale: 0.5-2.0 (all)
  - Rotation: 0-360 only in Y axis
  - Translation: should be placed above branches
- Leaves 3,4
  - Scale: 1.0-2.0 (all)
  - Rotation: 0-360 any or all axis
  - Translation: translated at least 4 units up (Y) randomly placed (X,Z)





- Leaves 5,6
  - Scale: one large leaf scaled at 6 in center of tree, can be more scaled at 1 outer area of branches
  - Rotation: 0-360 any or all axis
  - Translation: one large leaf translated up 4 units(Y), others randomly placed outside of that
- Leaf 7
  - Not used for this tree
  - leaves 8,9,10
  - Scale: 1.0-2.0 (all)
  - Rotation: 0-360 any or all axis
  - Translation: placed randomly in branches (X,Z) and at least 4 units up (Y)

