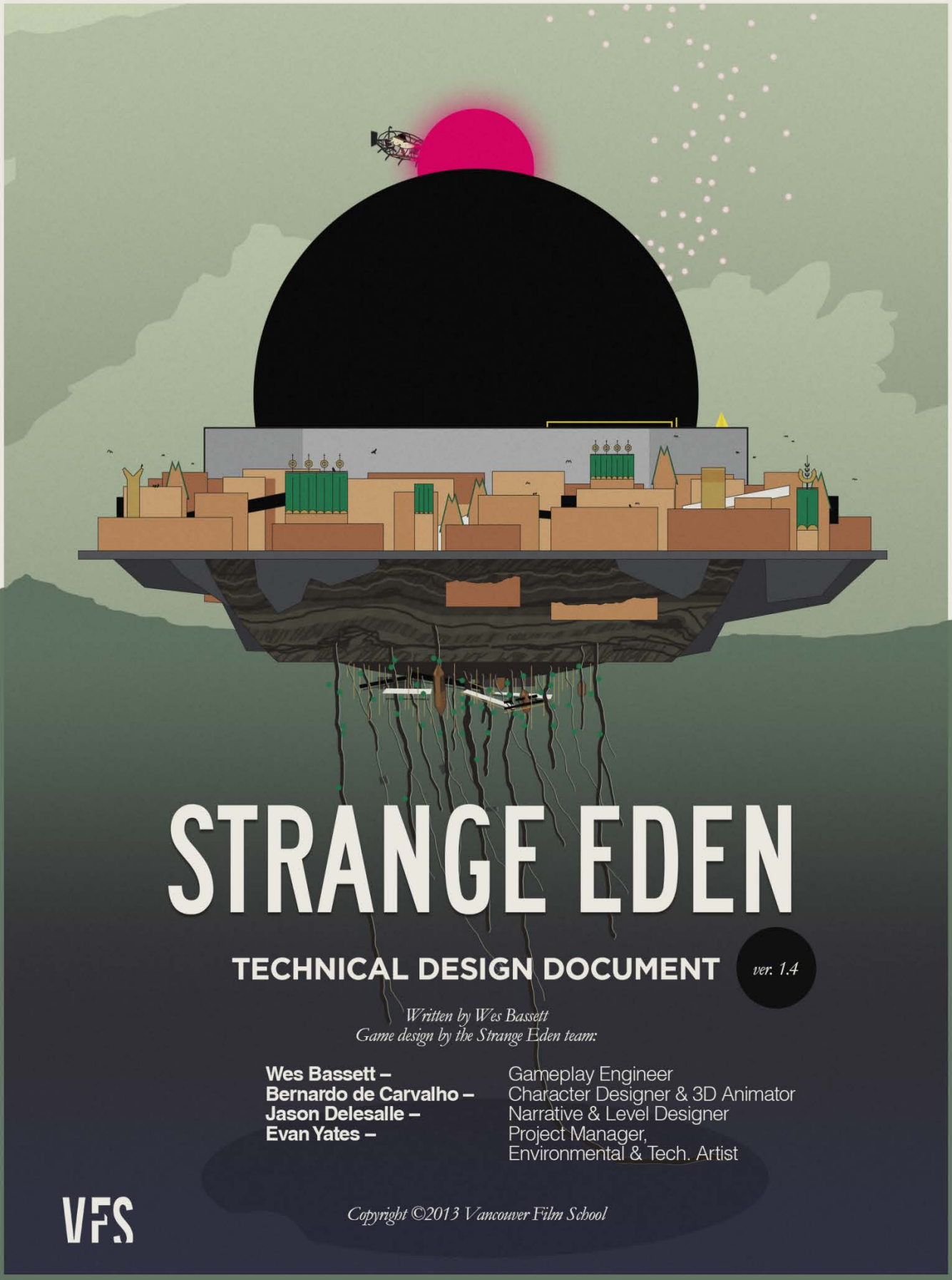

# STRANGE EDEN

## TECHNICAL DESIGN DOCUMENT

*ver. 1.4*


*Written by Wes Bassett*
*Game design by the Strange Eden team:*

**Wes Bassett –** Gameplay Engineer
**Bernardo de Carvalho –** Character Designer & 3D Animator
**Jason Delesalle –** Narrative & Level Designer
**Evan Yates –** Project Manager, Environmental & Tech. Artist


VFS

# 1.0 Design History

## 1.1 Initial Version

1. First implementation of the TDD
2. Information added from sections 1.0 until the end of 3.2

## 1.2 Second Iteration

1. Added sections 4.0 through 10.4
2. Look back over section 8.0 (8.2 in particular) as well as 7.2 and the appendices (section 10) with group before handing in
3. Update format, title and table of contents

## 1.3 Third Iteration

1. Updated 8.2, 7.2 and section 10
2. Updated formats and title.
3. Spelling / grammar changes throughout

## 1.4 Fourth Iteration

1. Added images to the document. See Table of Figures.

## Table of Contents

## Table of Figures

# 2.0 Game Overview

## 2.1 High Concept

Strange Eden is a third person puzzle platformer that is set on a foreign city that resembles a futuristic style of 1910. The main character is the wife of the first flight engineer, chasing her husband to the city in the sky to find out why he disappeared years ago. The main character utilizes a polarized flight suit that allows her to either attach or repel herself to similarly or oppositely polarized surfaces as well as fall at a slower altitude.

## 2.2 General Technical Overview

Strange Eden will utilize Unity 4.2 for PC download. It will support both Keyboard / Mouse functionality as well as gamepad support.

The engine was chosen because of its flexibility, ease of use, great online documentation, open community for both online support as well as in house support from instructors and mentors.

The Unity interface is intuitive and simple for our Level Designer, Artist and Environment artist to have autonomy when working within the engine. It also allows for a more streamlined pipeline where the programmer does not need to have to import all assets and is able to be managed by each team member individually.

The strongest benefit of Unity is its ability to have pre-built 3D collision and camera functionality that are strong enough to give Strange Eden a great baseline to start the project and to utilize that functionality throughout development.

# 3.0 Feature Set

## 3.1 General Features

### 3.1.1 Polarity (Sliding and Bouncing)
**Time estimation for the following features: 180 hours for Programming, 40 hours for Design**

The main mechanic of the game is being able to slide along surfaces as well as bouncing to gain height, speed and distance. The main character either has symbolic features of blue or red and the environment they are moving through is either colored blue, red or grey. The character will slide a long a surface if they are opposite colors to the surface, IE blue character touching red surface. The character will bounce off of the surface if they are the same color, IE blue character touching blue surface. If the character, regardless of if they are blue or red, touches a grey surface, no special interactions occur except for gravity and collision.

Sliding works by moving the characters forward acceleration by a minimum speed of 5x character walk speed and accelerates them by an additional 1.1x every second. The player still has full control of left and right movement, but they are forced to move forwards at all times while sliding. This speed cannot exceed 8x character walk speed. Characters cannot jump while sliding, they can only bounce off the surface.

Bouncing works by moving the character up relative to the world, as well as away from the surface, based on the characters position relative to the angle of the surface. This ensures that the character will always go up and away from the surface when they bounce from it.

If a player switches from one polarity to another, while in contact with a surface (for example, red character and blue surface, currently sliding across it) the change will be instant and would switch the effects of the polarity immediately.

This is the core feature of the game and requires the most time and dedication. Prototypes of these mechanics are required and extra hours may be necessary (estimated 40 hours) to make sure that the effects are tight and they feel intuitive and smooth.

### 3.1.2 Movement and Gravity
**Time estimation for the following features: 75 hours for Programming, 20 hours for Design**

The complimentary mechanic to the polarity is basic movement and the effects of gravity to the player.

When on a grey, flat surface the character will have a basic, slow movement, contrary to the speed of how they move while sliding, bouncing and falling. It works with basic WASD or left

joystick movement. Spacebar or Right or Left trigger will work as the jump while on the ground or will activate a reduced fall speed while in the air. Mouse / Right joystick will control camera movement and right / left bumper or left and right mouse click will control the two different polarities.

Basic movement constitutes a fixed camera based on right joystick / mouse movement and then character movement with full 360 degree control relative to the camera's direction always being forward.

The character will fall at normal gravity until they hit the spacebar mid-air. Doing so will cause the player to descend at half speed, and gain full control of their descent. They will be able to move in any direction 360 degrees while slow falling, but are locked to just falling with minimal control otherwise.

Prototyping will be quick to get basics working and requires little time to implement a poor version of it that is usable for the final product. Contingency is to just require the basic functionality and invest additional time taken from this into Polarity if required.

### 3.1.3 Turrets
**Time estimation for the following features: 120 hours for Programming, 20 hours for Design**

Turrets are the most demanding of the mechanics, not being a main core feature but requiring a deep time investment for both programming as well as design to make it feel like a strong feature.

The turrets are invulnerable objects against surfaces that will shoot orbs of varying size causing varying effects. One potential shot of the turrets is to shoot small blobs that attach to the player when they touch and slows the player down by 5% and increases fall speed by 5% for each one attached. The player can switch polarity to have them knocked off.

The other alternative is to have large spheres or random types of debris that are roughly half the size of the player that interact the same as polarized surfaces, lending for the potential to aid or hinder the progress of the player by bouncing them in an unfavorable or favorable direction (or sliding).

Prototyping is required to finalize which version of the turret is the final choice. It will also be important to find out if it adds to the gameplay or seems like more of a bother then a benefit.

## 3.2 Technical Implementation Plan

### 3.2.1 Polarity (Sliding and Bouncing)

1. Basic movement functionality with a Boolean for positive and negative polarity that changes the visual appearance of the players unit.
2. Prototype differing effects or debug log's with the polarity and touching other surfaces tagged as positive or negative.
3. Prototype a bounce effect that acts as a higher jump, only in the players up and not as a reflective bounce.
4. Prototype basic sliding, affecting the players forward motion so that they can turn while sliding.
5. Update bouncing to take the difference between the surface's normal local up and compare that to the player's forward and make bouncing the middle of those 2, rather than always going up in the world.
6. Change sliding so that the player adapts their position relative to the surface normal to allow the player character to slide up ramps, along walls and upside down.
7. Look at some kind of transition surface that doesn't impact the player's movement, but acts as a transition state between 2 different polarized surfaces requiring quick switching to complete.
8. Work with animator to create visual feedback.

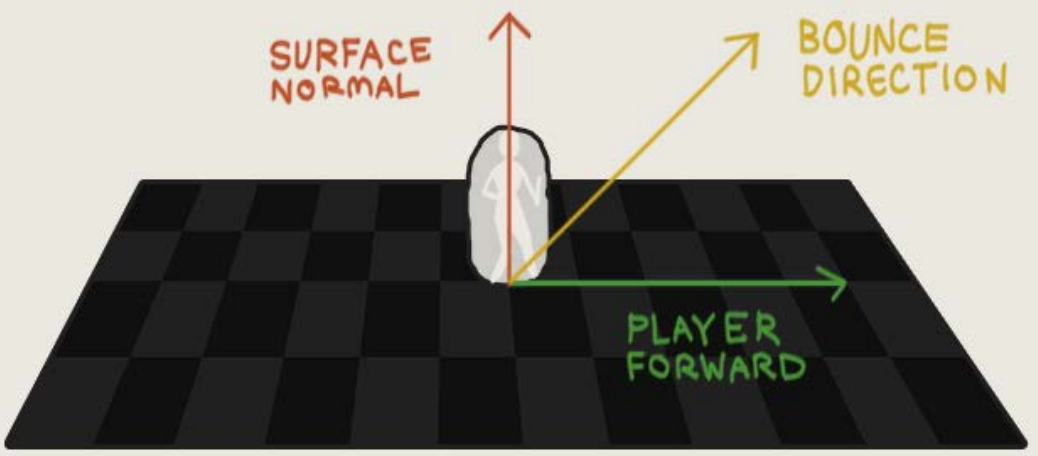


Figure 1: Bounce Direction Formula

### 3.2.2 Graceful Descent

1. Get regular gravity to work so that the player falls with limited movement aside from down.
2. Allow for a key press to activate a slowed fall.
3. Allow for more freedom of movement while slow falling.
4. Make sure the player keeps their forward momentum / falling arc.

5. Allow for full fluid movement while slow falling so that the player has more control for movement.
6. Work with the animator to create visual feedback.

### 3.2.3 Turrets

1. Create orbs that float in the air until the player hits them.
2. Have the orbs stick to the player.
3. Have it so that if the player switches polarity, it knocks the gooey balls off.
4. Get the orbs to affect the player's speed and falling speed when stuck to the player. Works additively the more that are attached.
5. Create turrets that shoot the gooey balls in a spray pattern, not aiming at anything in particular.
6. Look at having the turrets aim at the player, in the player's general direction or where the player is going to be.
7. Create turret movement and some indicator of how the turret is shooting so the player can react accordingly.
8. Work with animator to create turret movement and visual feedback.

**NOTE:** All of the technical implementation above assumes a minimum of 30 FPS at all times.

# 4.0 Camera

## 4.1 Overview

The camera system is a 3rd person free movement camera controlled by the player. Player input along the Y axis will create left and right movement that affects the player and camera equally, but movement along the X axis will create up and down movement that is camera only, allowing the player to look up and down for the next ledge. The 3rd person camera is also of varying distance behind the player, as the player can scroll in and out with the mouse wheel to bring it closer or further away. The player is also unable to unlock the camera from controlling the player's X axis movement by holding either Shift or Left Trigger

If the player goes on a surface that makes them turn to a different angle aside from straight up and down, the camera will keep the same rotation of right side up, so as not to create potentially unwanted disorienting effects.

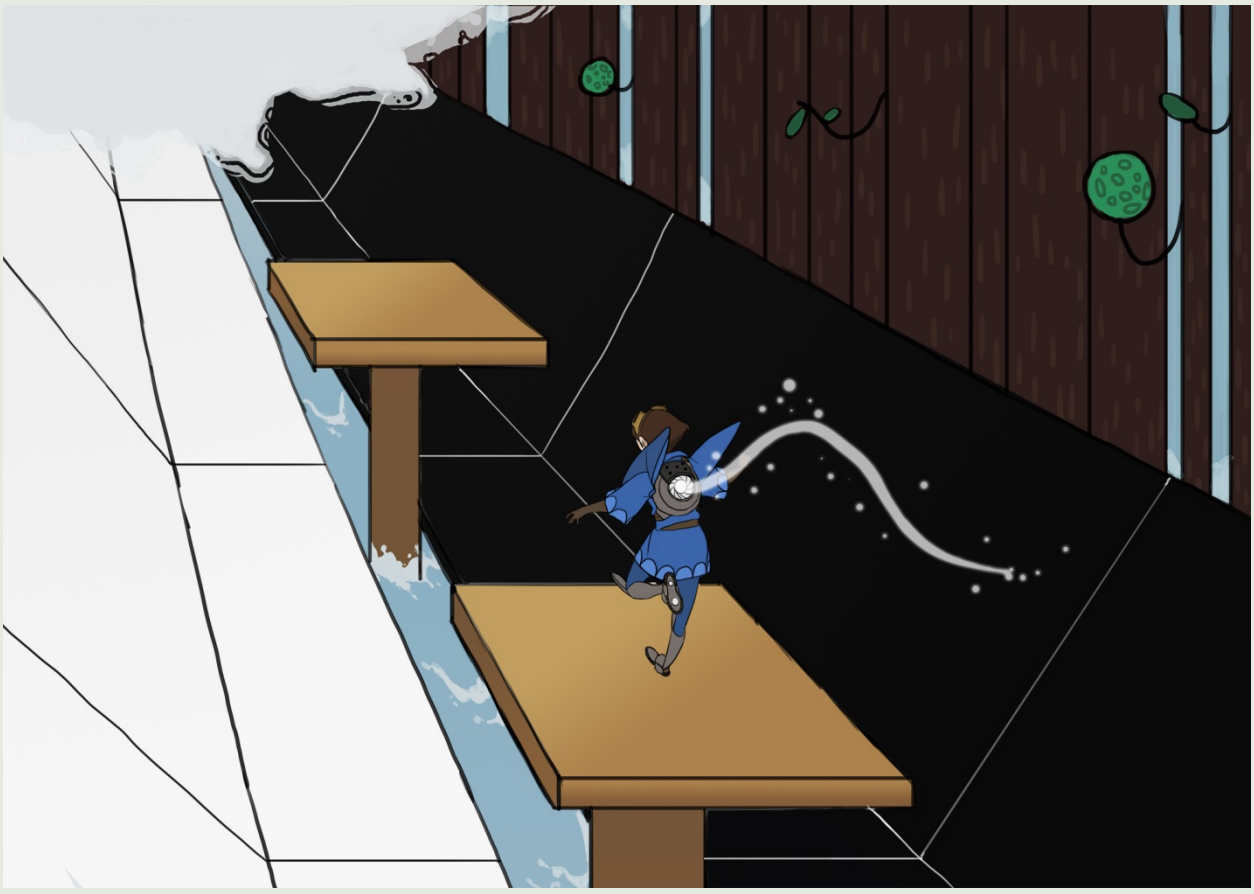

**Figure 2: In-Game Camera Example**

## 4.2 Technical Implementation Plan

**Time Estimation for the following implementation: Programming: 60 hours, Design: 20 hours**

1. Attach camera to player so it follows him around from behind.
2. Change the camera to function so that it can look up and down.
3. Add scrolling functionality so that it can zoom in and out.
4. Have the camera separate from the player so that the camera is looking around and player's horizontal movement is controlled by the camera
5. Have the ability for when the player is standing still or a button is pressed, that the camera no longer directs player movement and is instead separate to look around the world and rotate around the player (to see the full character)
6. Have it so that when the player goes upside down, the camera also goes upside down but both twerk so that it's not disorienting
7. Get the camera to NOT rotate with the player on the different surfaces but stay always up and down relative to the world so that the player is controlling the crazy tricks but is instead looks cool for the character to be doing the actions.

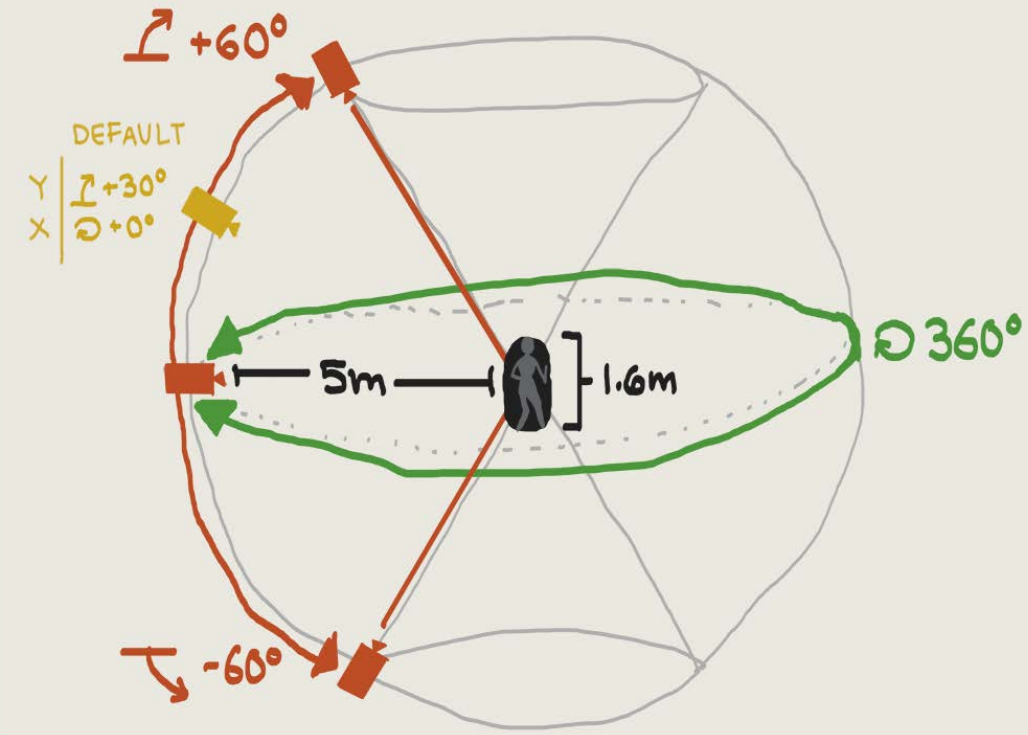


Figure 3: Camera Control

# 5.0 The Game World

## 5.1 Overview

The game world consists of one giant city representative of our “Strange Eden” that is one consistent level that the player travels from the bottom to the top of, going around multiple challenges and obstacles until they reach the top. The city is broken up into 4 different environments that have their own distinct appearances and minor differences in player interaction.

**Time Estimation for Game World: 50 Hours of Programming**

### 5.1.1 Boundaries:

The boundaries of each environment is different whether it is within a large dome, preventing the player from leaving or potentially having an endless emptiness below the world that falling into will kill the player and respawn them. Leaving the world results in death and respawning.

### 5.1.2 Polarized Surfaces:

The world is littered with black and white surfaces that appear everywhere and leads the player as their path. These surfaces react accordingly to the player based on the state of their polarity relative to the surface whether it results in sliding or bouncing.

### 5.1.3 Scale:

The city is atop a floating spherical piece of land with tentacles coming out from beneath it. The spherical diameter is 500 meters, relative to the player who is 1.62 meters.

## 5.2 Technical Implementation Plan

1. Create cubes and use tags to represent the positive and negative polarized surfaces to interact with the player.
2. Import Level Design and align in the world
3. Import Environmental design utilizing prefabs and textures

# 6.0 Game Characters

## 6.1 Overview

**Time Estimation for Programming: 20 hours**

Strange Eden consists of 2 characters of our heroine and our secondary character who acts as the player's motivation and objective.

## 6.2 Main Character

**Ida Dumont:** The player character who is an inventor that created her polarized suit, landing on Strange Eden in search of her Husband, August Dumont.

## 6.3 Secondary Character

**August Dumont:** August Dumont is Ida's husband and was the first person to head off to Strange Eden and since then has gone missing.

August Dumont does not have any AI. When the player see's August, he is either unconscious on the ground or floating / laying peacefully in a human sized holding pod. In all possible situations planned for the secondary character, none of them result in specially created behaviour outside of animations.

## 6.4 Miscellaneous

**Crows:** Crows are not a single character but they are the only entity that has their own dedicated programming on it. The crows are a reactionary particle that perches on structures throughout the world, occasionally making noise.

Figure 4: Crows

**Crow Behaviour:** The crows are 2D shapes that face towards the player and sits perched on objects throughout the game. They occasionally crow once or twice every 15 seconds when the player is 20 meters away from the crow. When the player becomes 7 meters or closer to the crows perched location, the crow will crow and fly off, circling in the sky until the player leaves that 7 meter radius of its perched location before returning to it.

There is also circling crows that circle around in the sky, out of the players reach, circling like vultures.

The crows are all 2D art plane's that are always facing towards the player so that it doesn't go invisible when the player goes beside it. This does not mean that the crow or the texture of it is directly facing the player.

**Technical Implementation: Estimated 40 hours**

1. Create a unit that circles around in the sky.
2. Implement a perching collision check and node system.
3. Implement player proximity to destroy the crow child.
4. Make the crow move to a node that starts circling
5. Work with animator to create a smooth flight pattern for the crows based on the nodes.

# 7.0 User Interface & Controls

## 7.1 Overview

In game user interface is non-existent aside from clearly communicating polarity to the player through the characters outfit. Beyond that, functionality for a pause menu is necessary.

Controller info for the game is built for PC and keyboard + mouse functionality first, with functionality for the gamepad being second, but equally as important so that players can enjoy the game with either control method of their choice.

## 7.2 The Controller

| In Game Command | Keyboard / Mouse | Gamepad |
|---|---|---|
| Move Forward | W | Left Analog Stick Axis Up |
| Move Left | A | Left Analog Stick Axis Left |
| Move Right | D | Left Analog Stick Axis Right |
| Move Backwards | S | Left Analog Stick Axis Down |
| Camera Rotation Y | Mouse Axis Up / Down | Right Analog Stick Axis Up / Down |
| Camera Rotation and Player Rotation X | Mouse Axis Left / Right | Right Analog Stick Axis Left / Right |
| Free Camera Rotation Unlock | Left or Right Mouse Button | Left Trigger |
| Polarity Toggle (primary) | Left Shift | ------- |
| Polarity Set Positive (secondary) | E | Right Bumper |
| Polarity Set Negative (secondary) | Q | Left Bumper |
| Jump + Slow Fall | Space Bar | A |

### 7.2.1 Mouse and Keyboard


Figure 5: Mouse and Keyboard Controls

### 7.2.2 Gamepad / Xbox 360 Controller

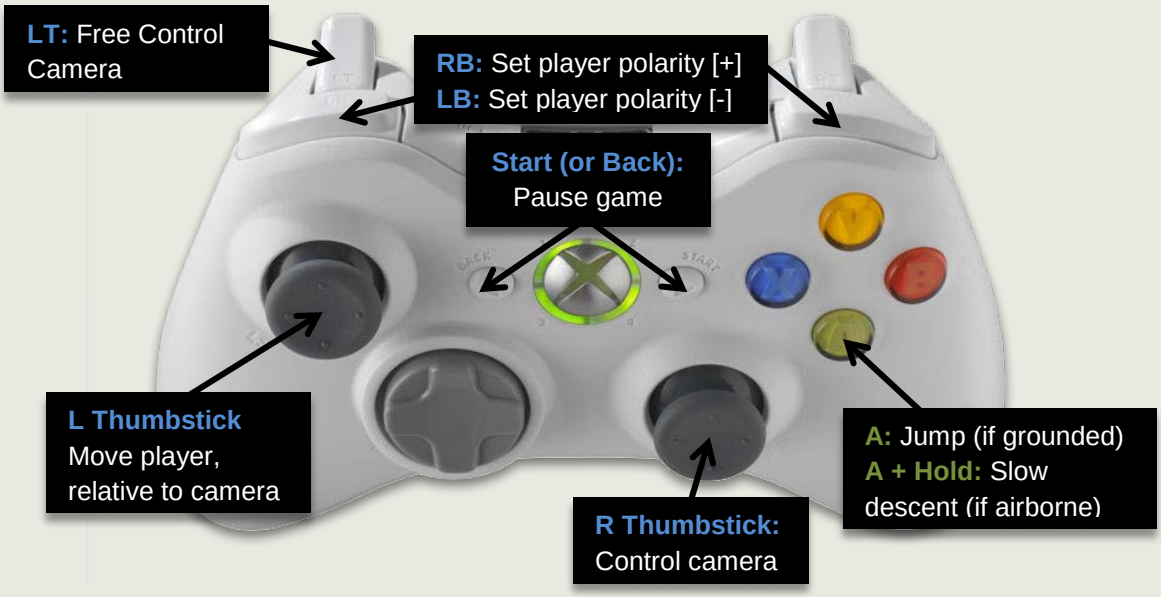


Figure 6: Xbox 360 / Gamepad Controller

# 8.0 Audio

## 8.1 Overview

The goal for audio in Strange Eden is to have a music score that starts off low and mysterious but builds in intensity as the player hits intervals throughout the game. On top of that, we are looking to have strong audio feedback to match with sliding, bouncing, polarity shifting and overall movement in the game. It's important that all audio feedback delivers the same intentions as the art and level design intentions. We will also be using Audio Toolkit.

## 8.2 Format

We will use WAV for our audio and utilize Unity's audio compression.

## 8.3 Sourcing

For the sourcing of the audio, we are looking to pitch to the audio campus and pick up sound designers that know how to balance audio, create unique sound effects and potentially a menu song. We are also looking to source a friend of the group who is known for creating his own music tracks in the past and owns his own little mini audio studio to create all the music with. There is the risk with both, but a bigger risk with our friend as he lives in Florida, USA and makes it difficult to keep up a constant communication as well as transfer of game prototype, art and audio to share ideas.

## 8.4 Sound Detail

The detail of our sound is broken up into musical layers and individual sound effects.

### 8.4.1 Music

The music of Strange Eden is broken up into 8 layers for each of the 4 environmental areas for the player to progress through. As the player reaches each challenge for each area, certain layers will fade out and others will fade in to create the feeling of a constantly building score that builds up along with the progression of the game.

The music plan is of a somber mysterious undertones that have a slightly upbeat (130+) Beats Per Minute with subtle percussion that underlines everything. We have talked about having synthetic music with the chorus done in a more traditional instrument such as a classical guitar and harmonica.

### 8.4.2 Sound Effects

The sound effects of Strange Eden are planned to be reactionary to the player's constant progression of movement, communicating energy and free movement. In a strange world, the sounds should be foreign but in some way comforting to how the player is able to move through the world.

The main sound effects needed are for Sliding on a Surface, Bouncing on a Surface, switching polarities, crows and running water. These are the building blocks for our environment and gameplay.

The Main menu can utilize the same sound effect for polarity switching on button presses.

# 9.0 Testing

## 9.1 Overview

We plan to have fresh eyes testing done by both Jason and Wes starting after week 3 of production. By that time, core features will be nailed down and basic level design implementations will be in game so that we can get opinions on what people think of the game as a whole. We should also hopefully have a 3D modeled character as well as environmental art.

We will test by first seeking out players from the term 3 and 4's, so that way we get more Game Design experienced testers within the still early pre-alpha phases of the game. Once we move to Alpha and Beta, we plan on having the term 1 and 2's get in on the testing as well as outside of school such as family and friends.

We will track bugs with both Pivotal Tracker, post it notes and note tracking that will be visible to everyone in the group as well as mentor's and instructors. We will be recording paper feedback both of personal notes of playtests as well as playtester experience organized in a folder that will be photocopied for backups.

We plan to utilize heat maps, player trails or both to find out what routes players are predominately taking as well as where they are dying or restarting from to make sure that there aren't large problems with the level design.

# 10.0 Appendices

## 10.1 General Information

Additional Assets:

- Audio Toolkit
- 2D Toolkit
- N-GUI
- Strumpy

## 10.2 Rendering Systems

Strange Eden will use fully utilize Unity's systems and assets for all features and rendering of the game. We are going to also use 2D tool kit to help with crow rendering and look to reduce stress of far off levels by using simple textures when too far away for the player to see.

## 10.3 Time Estimate

For the technical programmed aspects of the game, including the design of them, we have an anticipated 550 hours for base features and an additional 120 hours for stretch goals if we are ahead of schedule.

## 10.4 Prototype Notes

So far with the progression of the prototype, we look to potentially be on time or ahead based on future anticipated changes to basic mechanics of the game. It is functional and is a successful proof of concept as people have reported it to be a fun game thus far.

Unfortunately button toggle for polarity has hit unforeseen technical complications and will take a few extra hours to figure out the problem and fix it so one button can be used instead of 2 buttons for mouse and keyboard polarity. Initial feedback has seen that to be a positive goal.

Sliding works, but looking to have it so that the player moves according to the surface. Tried to have in time for prototype, but caused more problems than solutions offered. Put off until development, but basic functionality still commented in code. Otherwise, basic functionality of it is a success.

Bouncing works, but requires more time to make accurate bouncing mechanics work in directions other than straight up relative to the world.

Weird glitch has caused forward movement to freeze the player in the air when slow falling against a wall. Although it was unintentional, we have talked about potentially programming it to be a feature to grab onto surfaces, but not be able to climb them. Will require testing to proof of concept properly but is put on the back of the list as stretch goal if it works.

The prototype has proved to be difficult to play. Making controlled descent easier for basic players.