Flip ‘N Fall

1. GAME OVERVIEW

1.1. Two-Sentence Pitch

"Captain Toad: Treasure Tracker meets both Super Monkey Ball and Super Mario Galaxy in 3-D Puzzle Platformer. Captain Cube must manipulate the stage environments and gravity itself to reach the end of each stage and stop the evil space shark, Andromegalodon."

1.2. Two-Minute Pitch

Captain Cube, the main game character, is a dauntless space explorer that finds himself trapped in a galactic dungeon created by the evil space shark, Andromegalodon. In order to escape, Captain Cube must rely on his wits and his platforming abilities to escape the dungeon in which he is trapped. In order to do so, Captain Cube must not only brave traditional perilous platforming challenges, but must also rotate the entire stage and alter the direction of gravity to reach the end of each level.

Ultimately, Flip 'N Fall is a 3-D platformer, similar to Super Mario Odyssey or Super Mario Galaxy. However, because of the presence of additional features, such as stage rotation and gravity manipulation, that help to embellish the simple premise, each stage can be viewed in its entirety all at once as a diorama, similar to Captain Toad: Treasure Tracker. It is this dioramic view of each stage that Captain Cube will rely on to determine how to rotate the stage and in which direction gravity should be going.

The game will be built in the Unity 3-D game engine in order both to better develop our skills within the engine as well as to explore unique 3-D level design, which can be quickly accomplished using Unity.

2. GAMEPLAY OVERVIEW

2.1. Level Structure Overview

At the beginning of the game, the player will be greeted by a title screen that displays buttons with the options to start the game, to receive instructions on how to play the game, and to quit the game. On clicking on the option to start the game, the first stage begins, playing each subsequent stage each time the player beats a stage.

Each stage begins with the player spawning at a predetermined spawn point in the stage. The player be viewing the entire stage, including the player on the stage, at once. This means that the player must identify the exit door on the stage and begin planning on how to navigate the stage to reach the exit. After running, jumping, rotating the stage, manipulating gravity and finally reaching the goal door, the level ends and the next level begins. More details on the number of levels and environmental interactables will be presented in the ENVIRONMENT AND LEVEL DESIGN OVERVIEW section.

2.2. Basic Input Layout

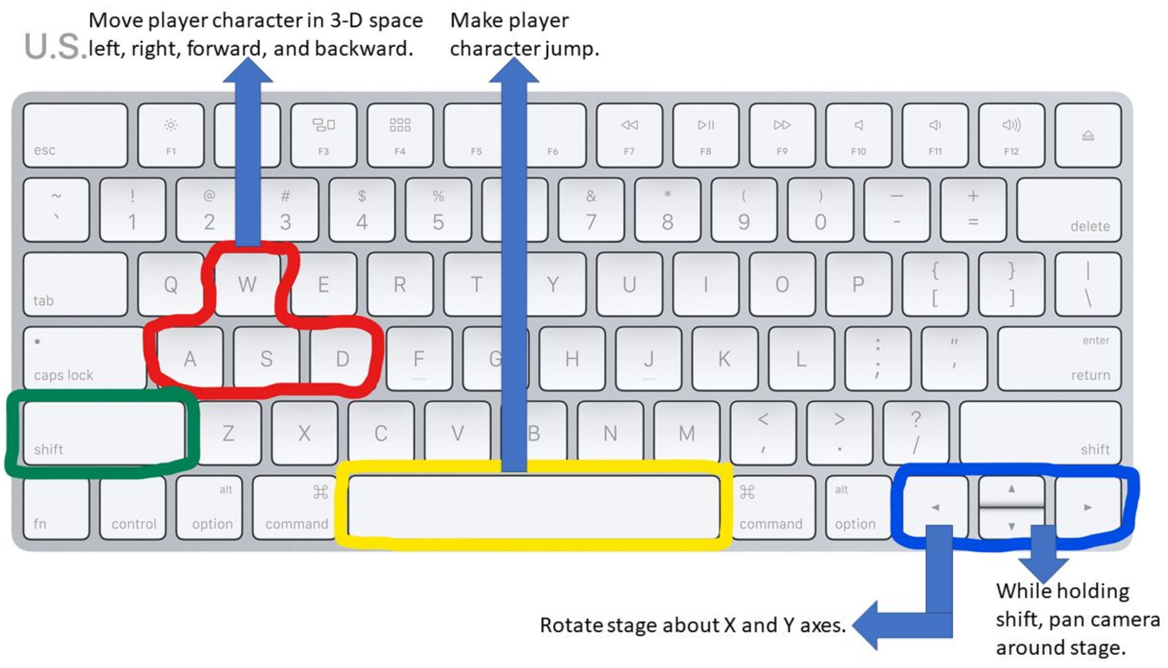

U.S.
Move player character in 3-D space left, right, forward, and backward.
Make player character jump.
Rotate stage about X and Y axes.
While holding shift, pan camera around stage.

Figure 1

Other keys: esc - returns to main menu

When jumping, the player will move 8 units upward in the Y direction.

When moving using WASD, the magnitude of the vector in which the player is moving will be 10 units.

When rotating the stage, the angle about the X and/or Y axes will be incremented/decremented by 1/10th of a degree every frame.
UPDATE: This gameplay feature has been cut. Stage rotation will instead occur as a result of an interaction with a stage element. The rate of rotation will remain the same.

When panning the camera around the stage, the camera's speed will be 10 units per second (subject to testing).
UPDATE: Because the rotation gameplay feature has been cut, holding shift is no longer necessary to pan the camera around the stage.

Press and hold shift to bring up a reference page that lists the environmental interactables and describes how each interactable behaves.

2.3. Player Character Controls

As partially described by the Basic Input Layout subsection, the player character will be controlled using WASD. To move forward, the player holds W. To move backwards, the player holds S. To move left or right, the player holds A or D, respectively.

Key combinations can also cause the player to move at a diagonal, e.g. holding W and D at the same time moves the player in a northeastward direction. If two keys that move the player in opposing directions are held at the same time, the player will not move, e.g. holding A and D at the same time will prevent the player from moving.

Friction will also cause the player to slow to a stop when no movement keys are being held (or if keys with opposite directions are held).

The player can also use the space bar key to jump in the air. The player will only be able to jump while grounded. However, the player will be given the ability to move while in the air, again by using the movement keys WASD.

The player will also be able to interact with various environmental interactables simply by moving into them. These interactables will be discussed in more detail later.

2.4. Camera Controls

The player will also be able to manipulate the camera to view the diorama-shaped level at other angles. Essentially, the camera would behave as if on the surface of a sphere whose center is fixed in the very center of the stage. When the player inputs a directional key, the camera would travel along that surface and point towards the fixed center of the sphere.

The player will be able to control the camera in this way to prevent the player character, environmental hazards, and objectives from becoming hidden by various stage geometry.

As indicated previously, the speed at which the camera would travel along the surface in a direction given by the player is 10 units per second.

2.5. Health/Lives System

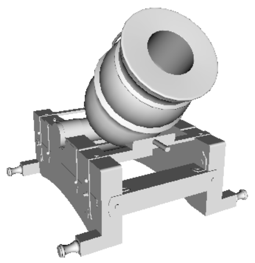
The player character will have a health meter that indicates that the player has 10 Hit Points (HP). As the player loses Hit Points, the meter will drop. Once the meter is completely depleted, the player dies, loses a life, and respawns at the beginning of the level (because of the small sizes of each stage, checkpoints will not be necessary). The player will respawn with full health at the spawn point of the stage (It is worth noting that the stage will also be set to its original rotation to prevent an unavoidable death loop).

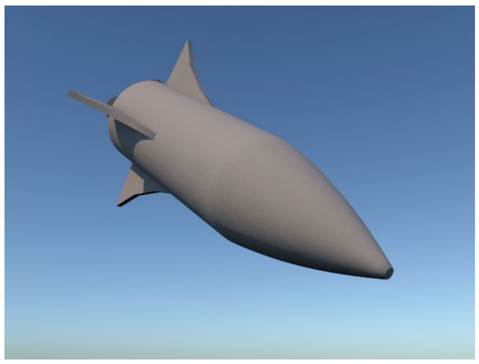
If the player is hurt, an I-frame state is activated, reducing the alpha of the player to 0.5 and preventing the player from taking additional damage for 1 second, giving the player character the chance to move away from the entity causing the player character to take damage. Once the time elapses, the player character’s original alpha is restored and can again take damage.

If the player loses all their extra lives and dies, a game over screen appears and sends the player back to the main menu after 1.5 seconds.

3. ENEMY OVERVIEW

3.1. Cannon Enemy





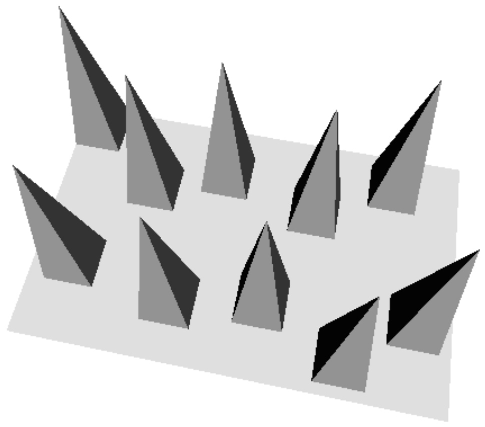
The main enemy that the player will encounter throughout each stage is a cannon enemy. These enemies, which are fixed to certain locations on the stage, will target and fire bullets at the player character once the player character reaches the cannon's trigger field. If the cannon is not facing the player when the player enters the field, the cannon will turn to face the player. This angle incrementation/decrementation will be described as 1/10th of a degree every frame.

Once facing the player character, the cannon will instantiate bullets at a steady rate, around 3 bullets per second, in the direction of the player character. The bullets will travel at a rate of 5 units per second and will not be under the influence of gravity. When the bullets hit the player, hit stage geometry, or travel out-of-bounds, the bullets will be destroyed (out-of-bounds detection will be handled by the death planes, which will be explained in section 4.1).

When hit by a bullet, the player character's HP will be reduced by 2 points.

The player will be unable to destroy these cannons. They can only be avoided or moved during stage rotations.

3.2. Spikes



A stationary obstacle that, when in contact with the player, will deplete the player's health by 2 HP. The spikes cannot be destroyed, meaning that the player must jump or use gravity to get around these objects. The spikes also act as environmental hazards that are attached to the stage. This means that, as the stage rotates, the spikes move with the stage.

If the player is standing on the spikes and has taken damage, the I-frame state will activate. The player can run on the spikes during this state so that the player has a chance to stop taking more damage.

3.3. Gravity Blocks

These blocks will be influenced by the current direction of gravity, just as the player character is influenced. As the direction of gravity changes, the blocks will fall in the direction of the new gravity vector.

These blocks are able to crush the player. If a block manages to sandwich a player character between itself and a section of kinematic stage environment, then the player will lose 5 HP. During the player’s I-frame state, the block can still collide with the player. However, the player will not take damage until I-frame state ends.

4. ENVIRONMENT AND LEVEL DESIGN OVERVIEW

4.1. Basic Layout of Diorama Stage/General Stage Elements

Each stage, at its core, is made up of primitive 3D objects that are made children of an empty game object at the center of the stage. This allows the developers to create a script that can easily rotate the stage by directly manipulating the rotation of the empty game object, causing the entire stage to be rotated with it. Multiple stage elements will be made children to this game object in a similar way to allow those stage elements to be moved with the stage as well.

There is a spawn point where the player will spawn, either when the player first enters the level, or after the player has used an extra life to continue playing. This spawn point will be made a child of the empty game object.

There is a goal point, represented by a portal. Walking into the portal causes the scene manager to load the next level in the game.

Deliberately placed throughout the stage are blocks that are color coded. These blocks are gravity distortion blocks. The way they distort the gravity is dependent on the color of the blocks. Below is a legend of how each color distorts the gravity:

- Black - Reverses the current gravity vector
- Blue - Rotates the current gravity vector about the X axis in a clockwise direction by 90°
- Orange - Rotates the current gravity vector about the X axis in a counterclockwise direction by 90°
- Purple - Rotates the current gravity vector about the Y axis in a clockwise direction by 90°
- Yellow - Rotates the current gravity vector about the Y axis in a counterclockwise direction by 90°

These blocks reverse the gravity as described above when a player collides with one of them (each of which is a trigger). They do not behave under the influence of gravity and again are made children of the empty object so that they can be moved when the rest of the stage is moved.

When the player collides with one of these blocks, the block's alpha is reduced to 0.5, which acts as a visual indicator to the player that gravity cannot be changed again using this block. Only after the player exits the trigger collider of the block will the block's alpha return to normal and the player can use that block to manipulate gravity.

In addition to gravity manipulation blocks, each level will also be populated by spheres that can rotate the stage in a variety of ways based on the color of each sphere. Below is a legend of how each color distorts gravity:

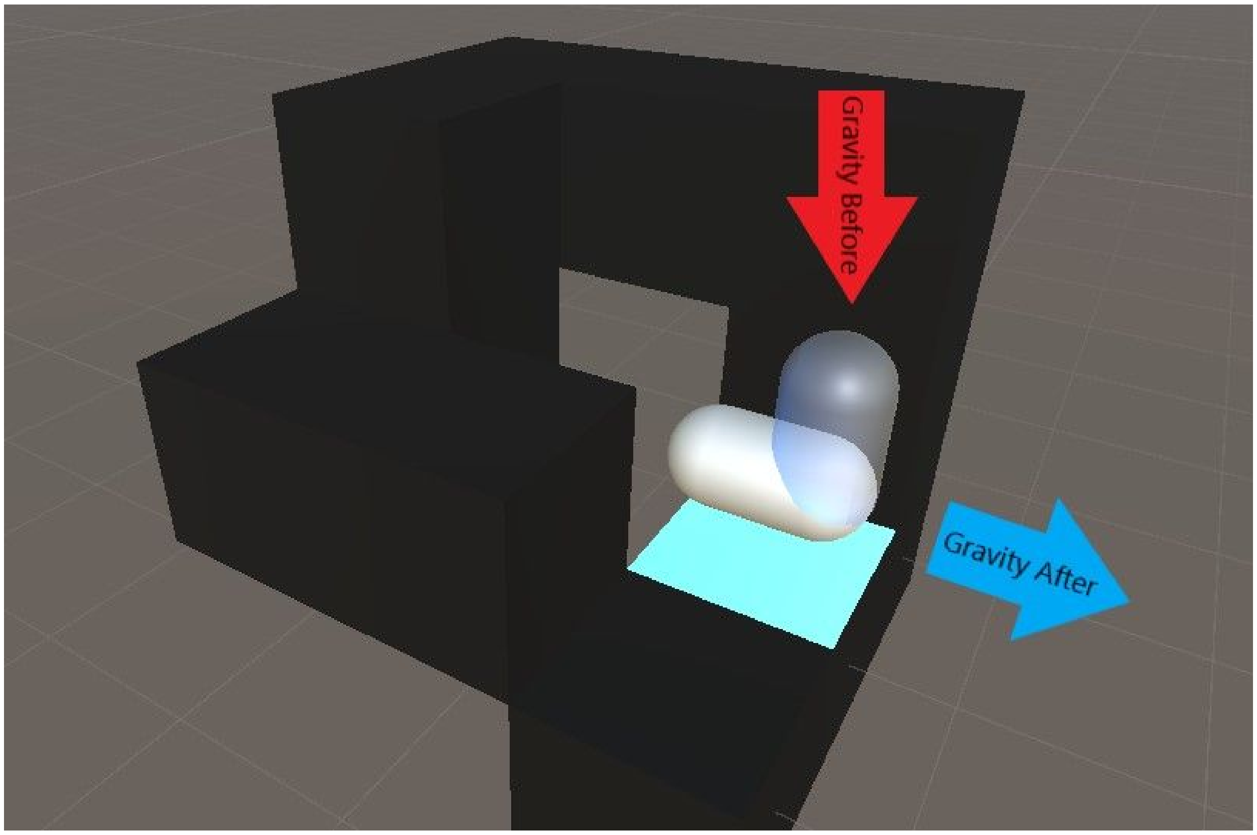
- Blue - Rotates the entire stage about the X axis clockwise by 90°
- Orange - Rotates the entire stage about the X axis counterclockwise by 90°
- Purple - Rotates the entire stage about the Y axis clockwise by 90°
- Yellow - Rotates the entire stage about the Y axis counterclockwise direction 90°

Similarly to the gravity manipulation blocks, when the player collides with one of these blocks, the block's alpha is reduced to 0.5, which acts as a visual indicator to the player that gravity cannot be changed again using this block. Only after the player exits the trigger collider of the block will the block's alpha return to normal and the player can use that block to rotate the stage.

It is worth mentioning here that when the player character has interacted with one of these stage rotation blocks, the player will be frozen in mid-air until the stage has completed its rotation. Once completed, the player character will again be allowed to fall in the direction of the current gravity vector. Since this behavior is part of the player state, more details on this can be found in section 6.2.

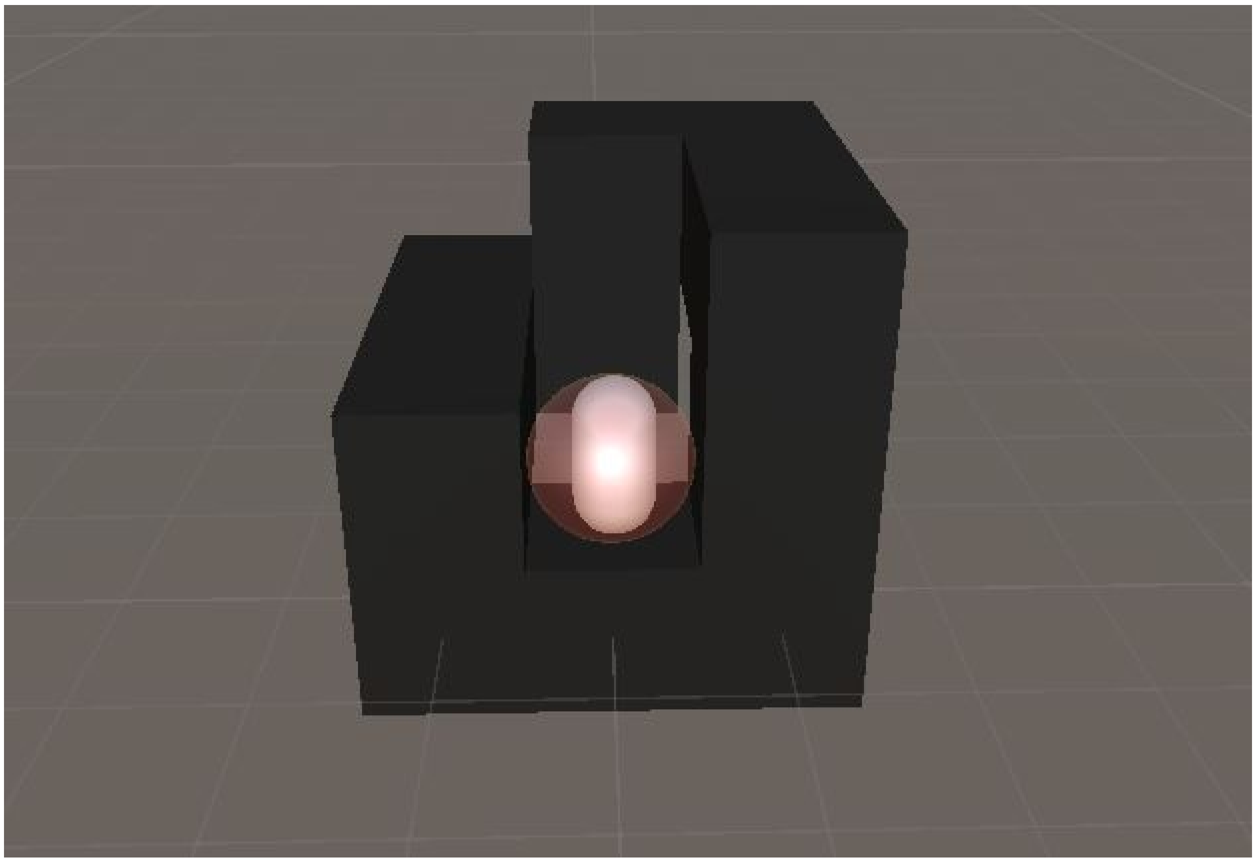
Finally, surrounding each level is a set of four invisible rectangular prisms that contain each stage. These act as death planes that cause the player to lose all HP when the player collides with them.

4.2. Demonstration of Gravity Manipulation Interactables

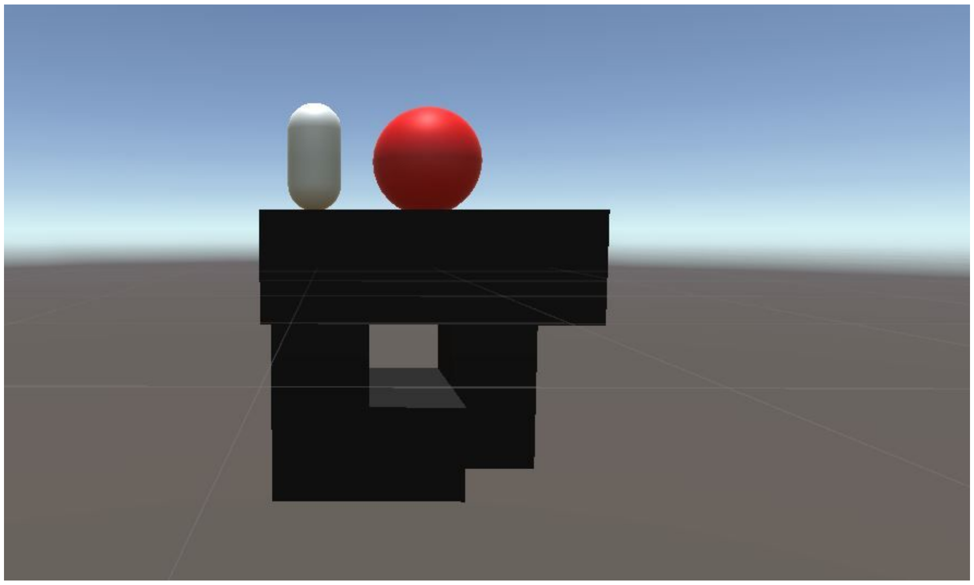


Above are images demonstrating the effects of interacting with the gravity manipulation blocks.

4.3. Demonstration of Stage Rotation Interactables



Above is the initial view of when the character enters the sphere.



Above is the view after the stage has rotated and the player has moved out of the sphere.

4.4. Andromegalodon Boss Stage

In addition to the three normal stages, the final stage will be a boss stage featuring the boss Andromegalodon.

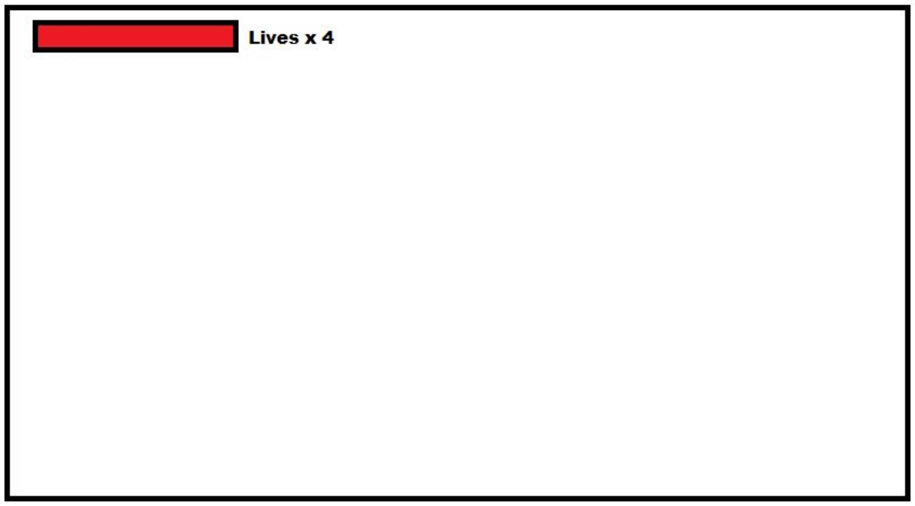
Andromegalodon will not actually fight the player, but will instead swim menacingly in the background.

The boss stage acts as a final platforming challenge where the stage is rotating by itself. This means that there will be no stage rotation interactables in the final fight. Instead, the stage will be populated with the enemies described previously as well as gravity manipulation blocks that can either help or prevent the player from remaining on the rotating stage. There will be a timer that lasts for two minutes and appears in the top right of the GUI. Once the timer is depleted, the level ends and the final victory screen appears.

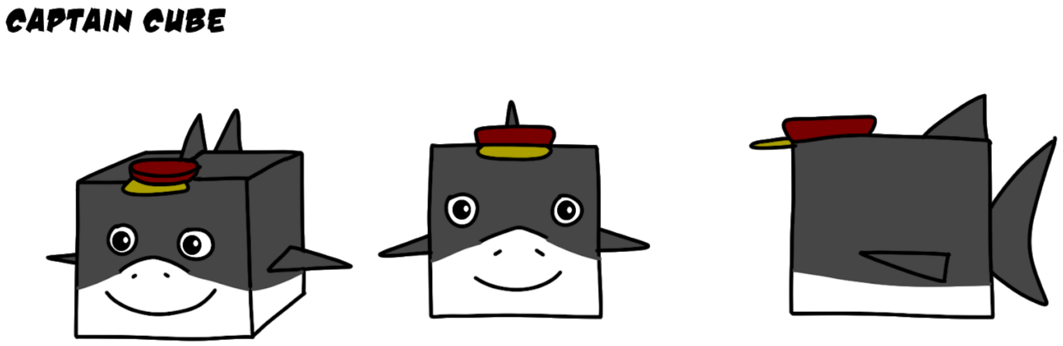
5. ART DIRECTION DOCUMENTATION

5.1. GUI

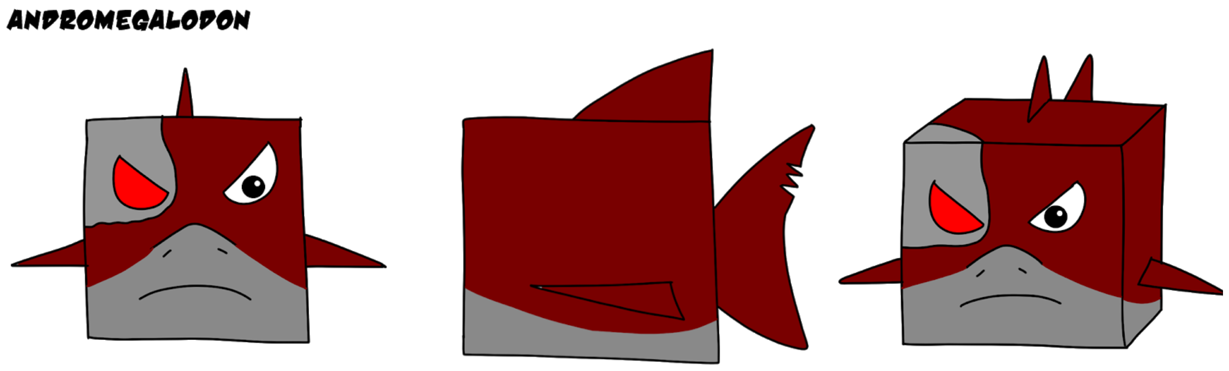
The user interface will be very minimal, featuring the health system of the player as well as the number of lives the player has left. Both the health system and number of lives will be featured at the top of the screen. The health system/meter will be split into 10 sections/bars. Each time a player takes damage a bar of health will be depleted. The depleted bar of health will have a darker visual, different from the lighter visual that the bar of health will have when it is full. The number of lives the player has left will be displayed at the top of the screen alongside the player’s health. It will be displayed with a multiplication value followed by a number (e.x. “Lives x 4”).



5.2. Concept Art for Captain Cube Character

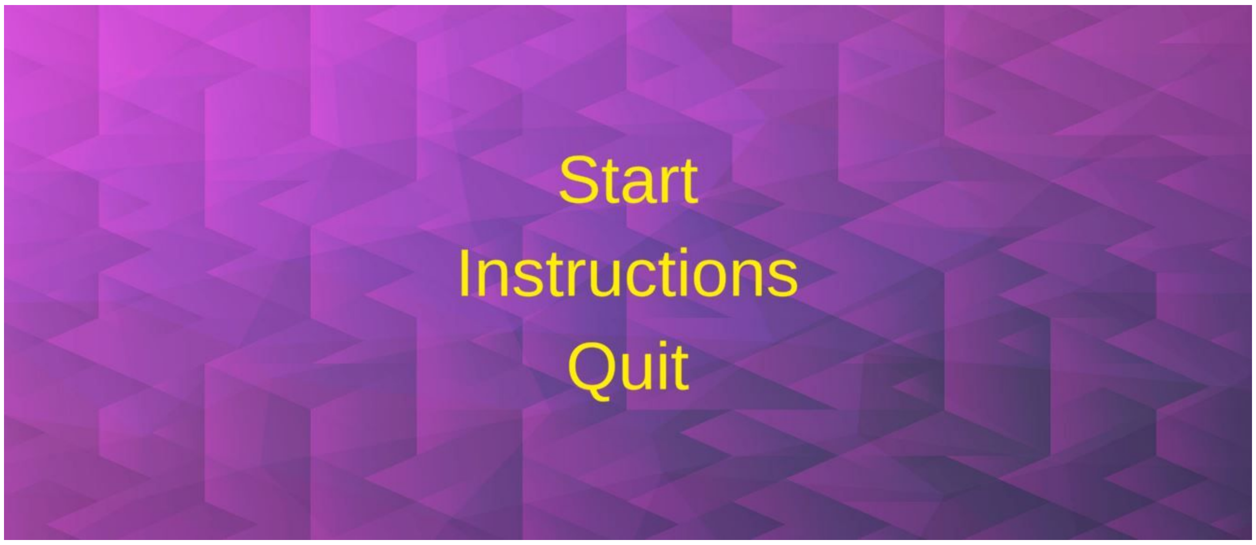


5.3. Concept Art for Andromegalodon

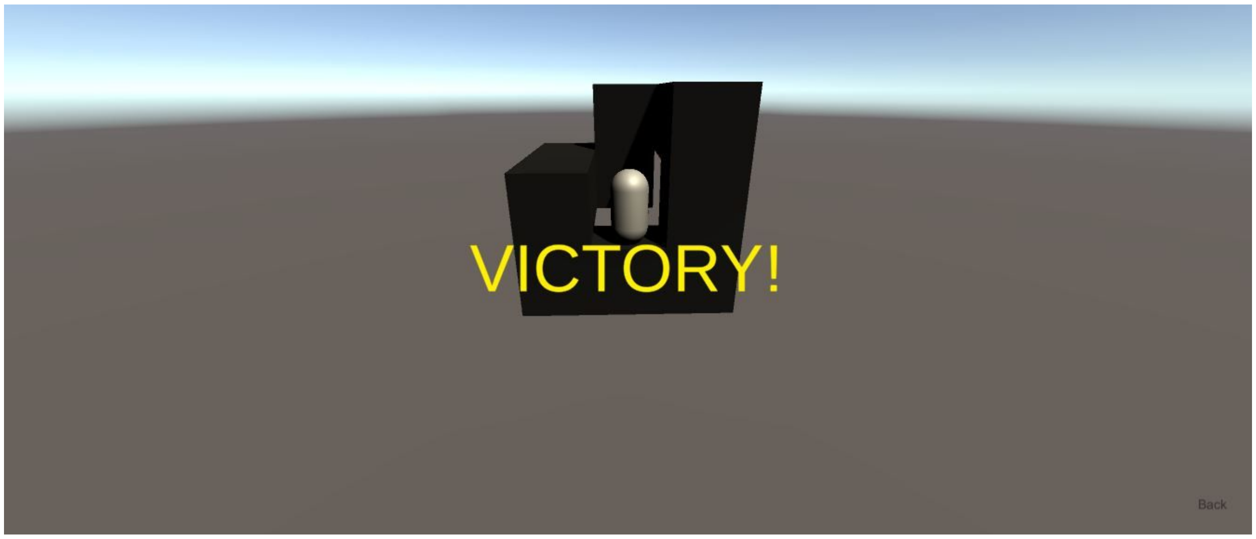


5.4. Designs of Title, Victory, Game Over, and Help Screens

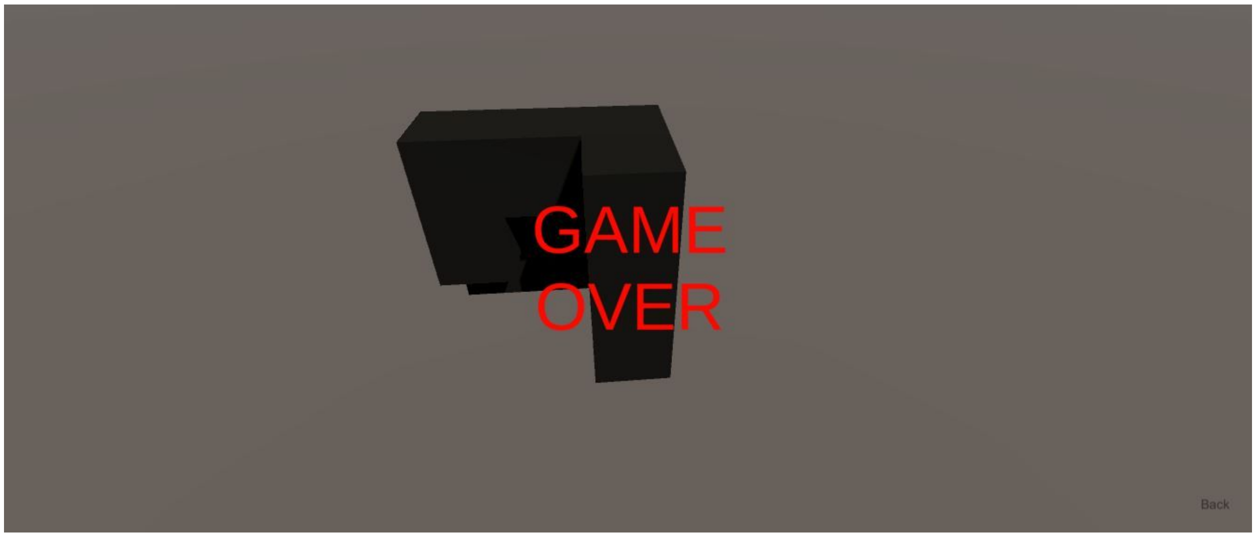
5.4.1. Title Screen



5.4.2. Victory Overlay



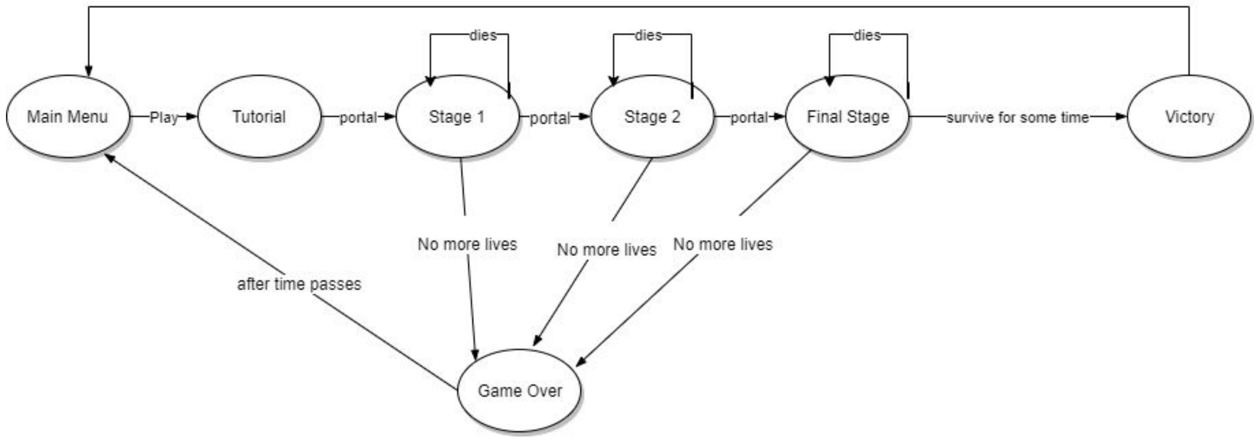
5.4.3. Game Over Overlay



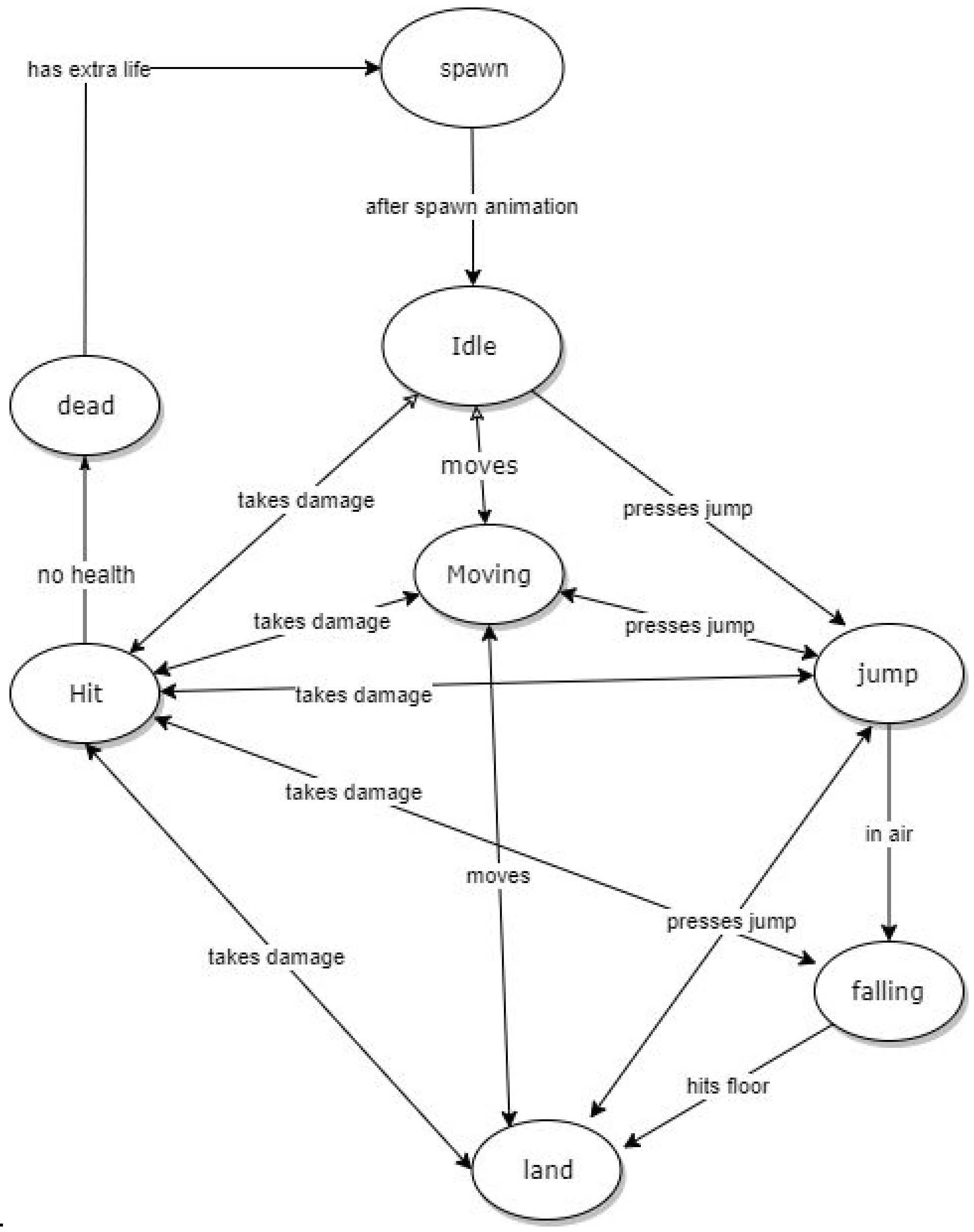
5.4.4. Instructions Screen

6. GAME STATE MACHINES

6.1. Overall Game States

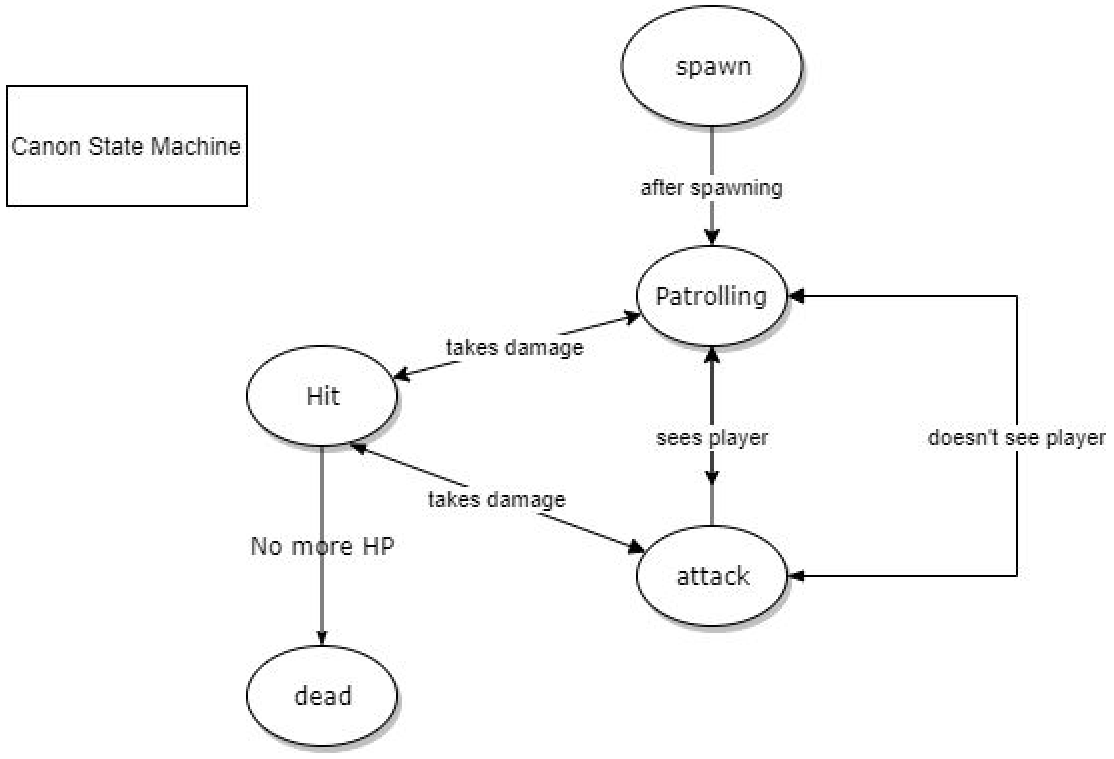


6.2. Player State Machine

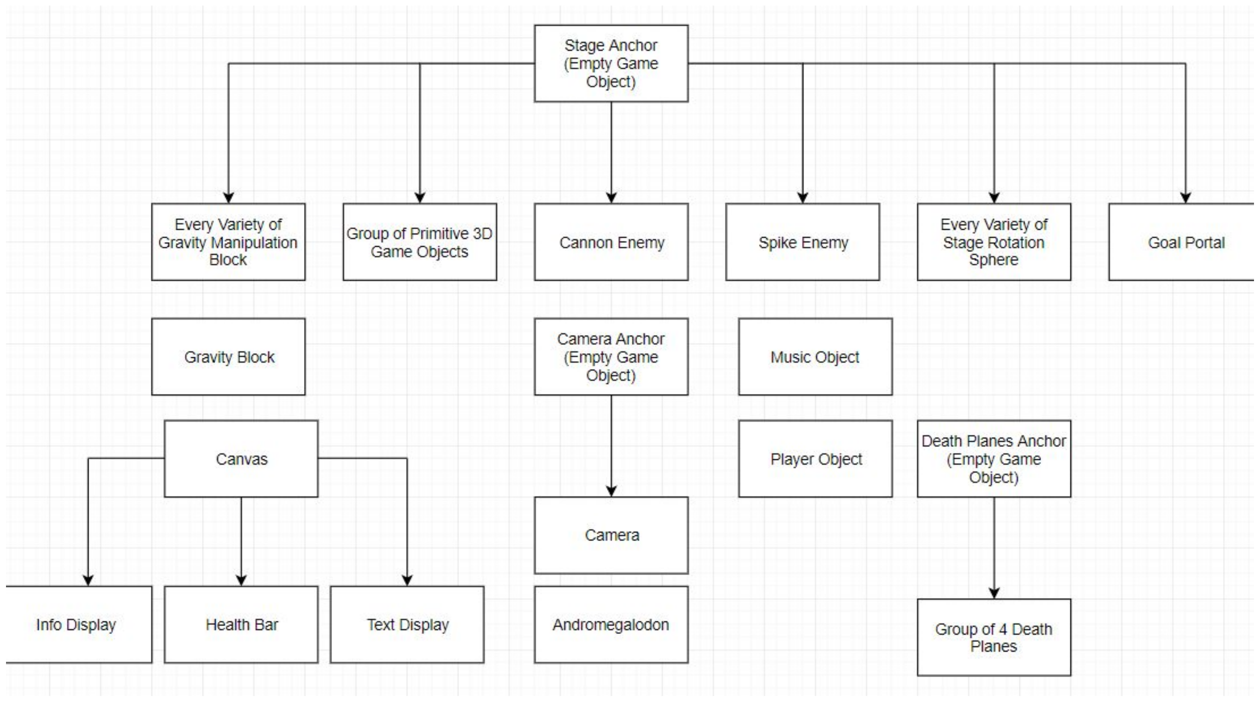


.

6.3. Enemy State Machines



7. LIST OF EXPECTED GAME OBJECTS/SCRIPTS


Stage Anchor (Empty Game Object)
Every Variety of Gravity Manipulation Block
Group of Primitive 3D Game Objects
Cannon Enemy
Spike Enemy
Every Variety of Stage Rotation Sphere
Goal Portal
Gravity Block
Camera Anchor (Empty Game Object)
Music Object
Canvas
Player Object
Death Planes Anchor (Empty Game Object)
Camera
Info Display
Health Bar
Text Display
Andromegalodon
Group of 4 Death Planes

Above is a general inheritance tree listing the expected game objects and their children found in the main game. The tree excludes the Title, (which will be a canvas in a separate scene with buttons allowing the player to Play the game, see the game instructions, and quit the game), the Game Over screen, (which will be another canvas in a separate scene that displays the text "Game Over"), and the Victory Screen, (which is another canvas in a separate scene that displays the text "You Won!"). By children, we don't mean it in the strictly Object-Oriented sense, but in the way Unity handles children by allowing multiple objects to share the transform of a parent object.

We will also list the following scripts that are likely necessary to implement each game feature. These are:

- Player Character Movement Script
- Camera Rotation Script
- Player Health System Script
- Enemy Health System Script
- Cannon Enemy Behavior Script
- Gravity Manipulation Script
- Stage Rotation Script
- Andromegalodon Model Movement Script
- Level Transition Script
- Level Transition Timer Script

8. TIMELINE/MILESTONES

1. Basic controls
    a. Player controls: WASD movements, Jump
    b. Camera controls: Third-Person, Panning
2. Implementation of mechanics
    a. Gravity
    b. Rotating environment: interactable objects
3. Prototype Levels
    a. Tutorial Level
    b. Main Level
    c. Boss Level
4. Enemy Designs
    a. Cannons
    b. Spikes
    c. Gravity Blocks
5. Early Back-End
    a. Play/Rules/Quit Screen
    b. Pause Screen
6. Boss Design
    a. Andromegalodon
7. Polish

9. TEAM BIO/EXPECTED CONTRIBUTIONS

Team Members

Josef Reif

Josef, most often referred to as Sef, will be working on the stage rotation and gravity manipulation mechanics, as well as various quality of life changes. Brand new to game design, Sef has experience working with Nav Mesh components to create both enemy movement as well as player movement. He will be graduating with a degree in Informatics with a focus area in Game Design.

Jacob Hreshchyshyn

Jacob, the team lead for this final project, will be overseeing the development process and will be working primarily on consolidating each individual component implemented by the other developers. He will also be assisting in level design. Having experience working with

GameMaker Studio 2 in the Spring 2019 semester and Unity in the Fall 2019, Jacob's expertise lies in level design. Jacob is a Sophomore majoring in Software Engineering and is pursuing ASU's Game Development Certificate.

Sammy Clayden

Samuel Clayden, or Sammy as he prefers to be called, will be taking care of the concept art for the characters, implement the designs for the GUI, and the title, victory, and game-over screens. Also a newcomer in the game design world, Sammy's strongest field seems to be GUI elements such as health and lives.

Tyler Taing

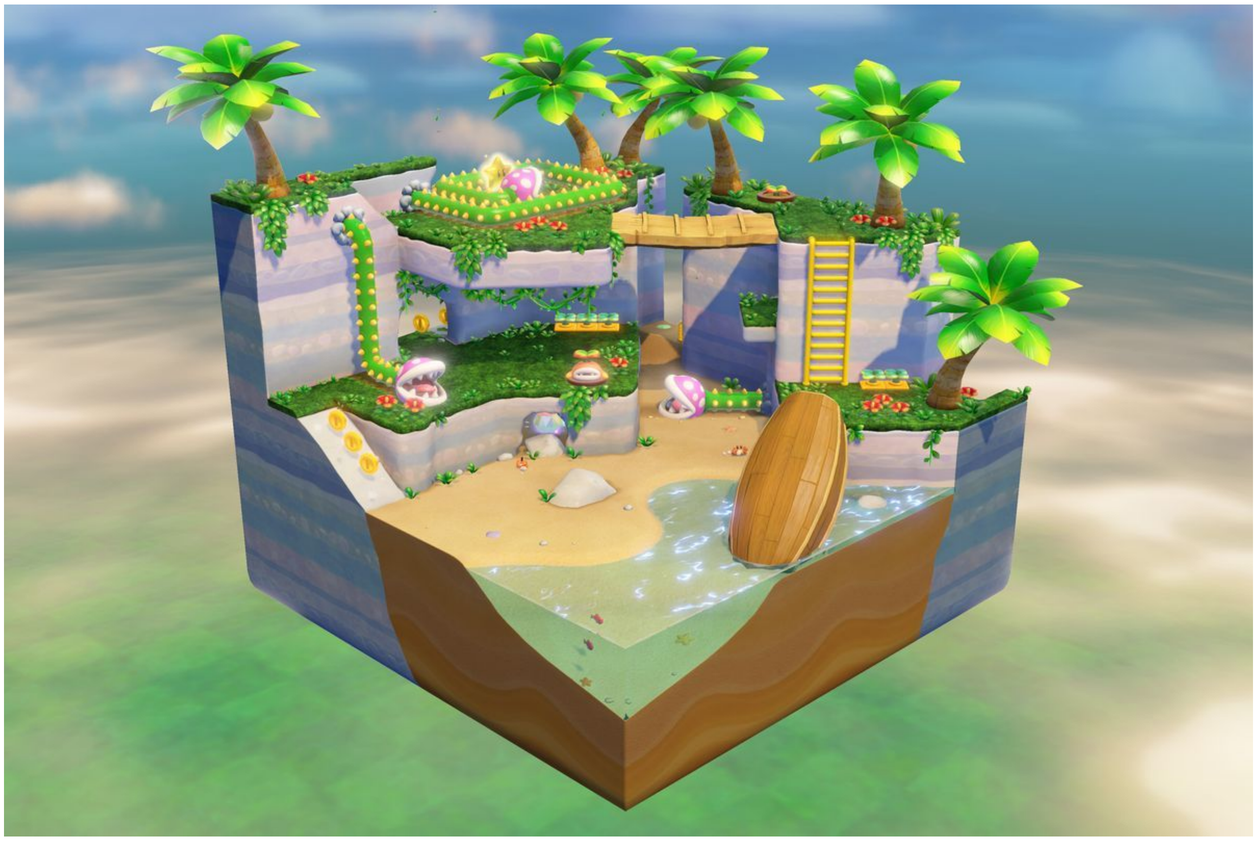
Tyler will be working on animation, modeling and some game mechanics such as enemy behavior and attack animations. Having previous experience with Unity and GameMaker Studio 2, Tyler specializes in animations and modeling and is aiming for a Game Development Certificate from ASU.

Wei Chieh Huang

Wei Chieh Huang, or you could call him Joseph, will be working on assisting in level design as well as quality assurance. His previous class experience includes working with raycasts to implement dynamic shooting mechanics as well as working to implement health systems. Joseph is majoring in Computer Science and aiming for Game Development Certificate from ASU.

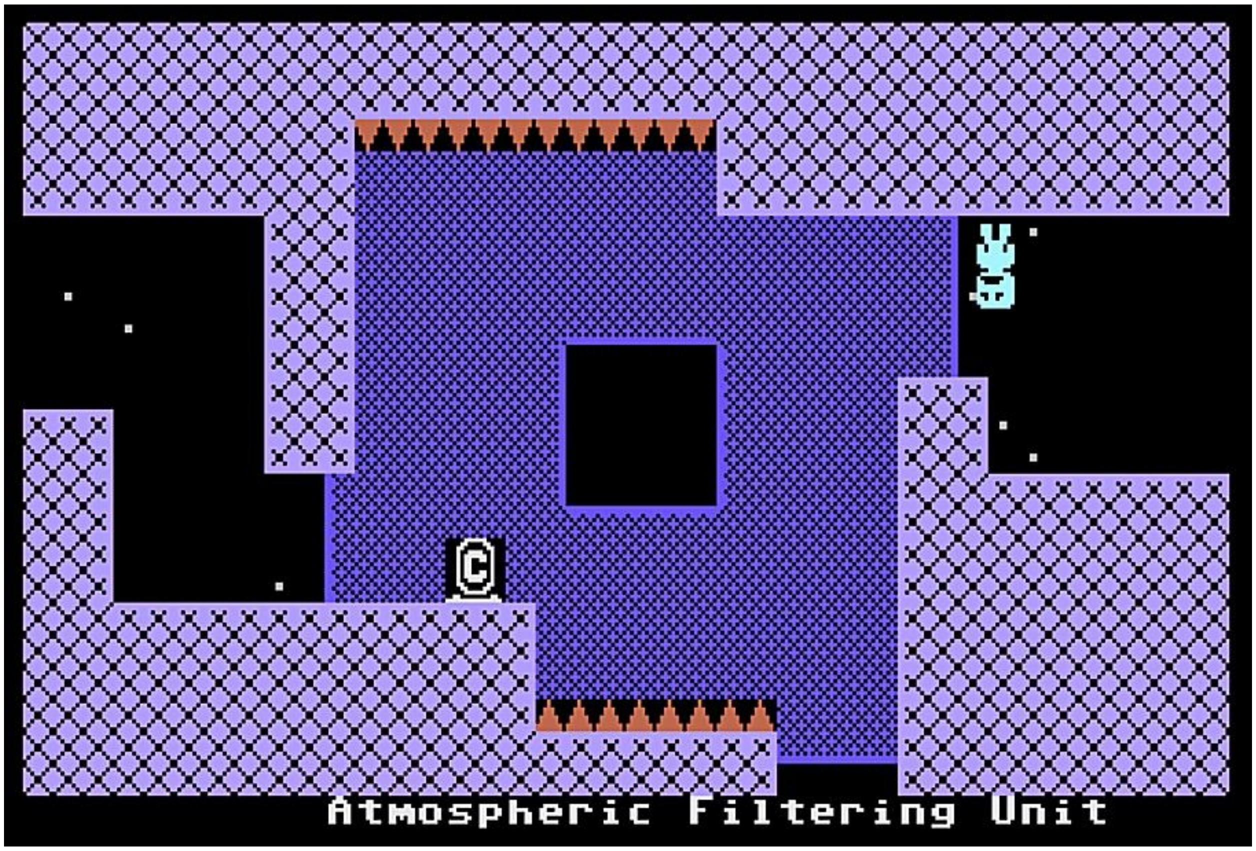
10. APPENDIX

10.1. Reference Games

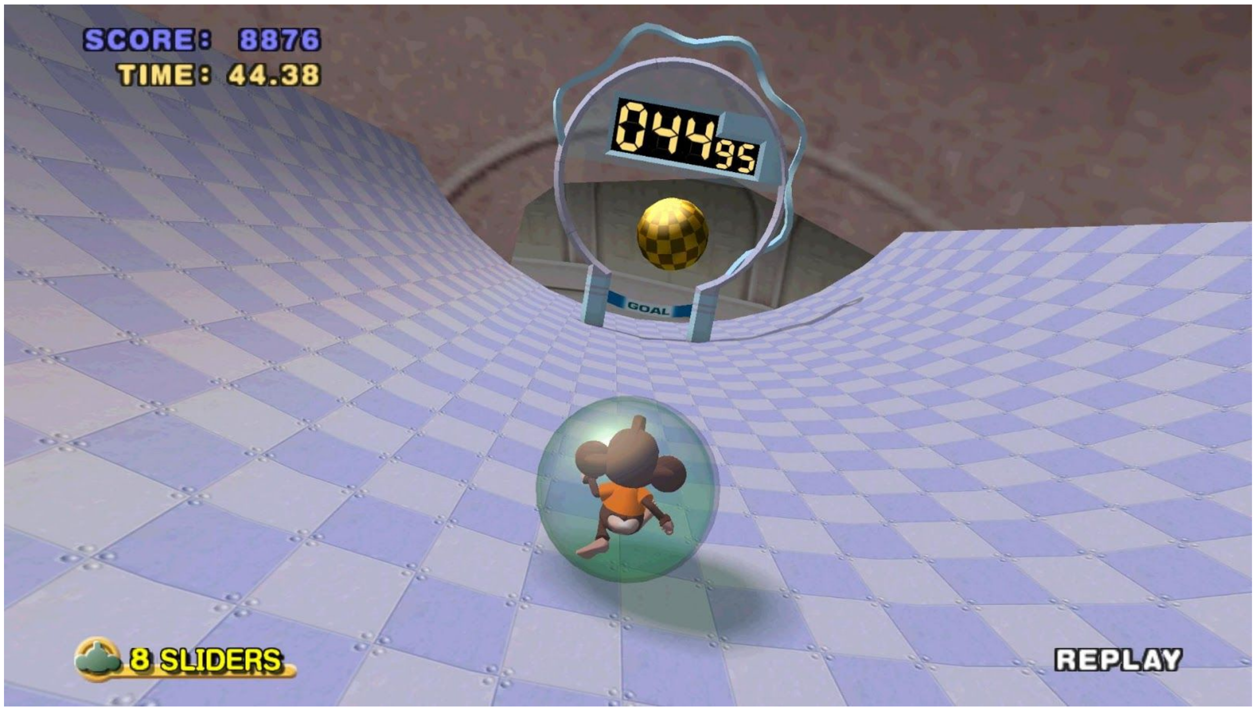


Captain Toad: Treasure Tracker - Diorama-shaped levels allowing the player to view the entire level at once.

Super Mario Galaxy - Gravity mechanics and inspired level design.



VVVVVV - Gravity mechanics, inspired level design, and great music.



Super Monkey Ball - Tilting stages.