Design Document for:

Wednesday, December 19, 2018

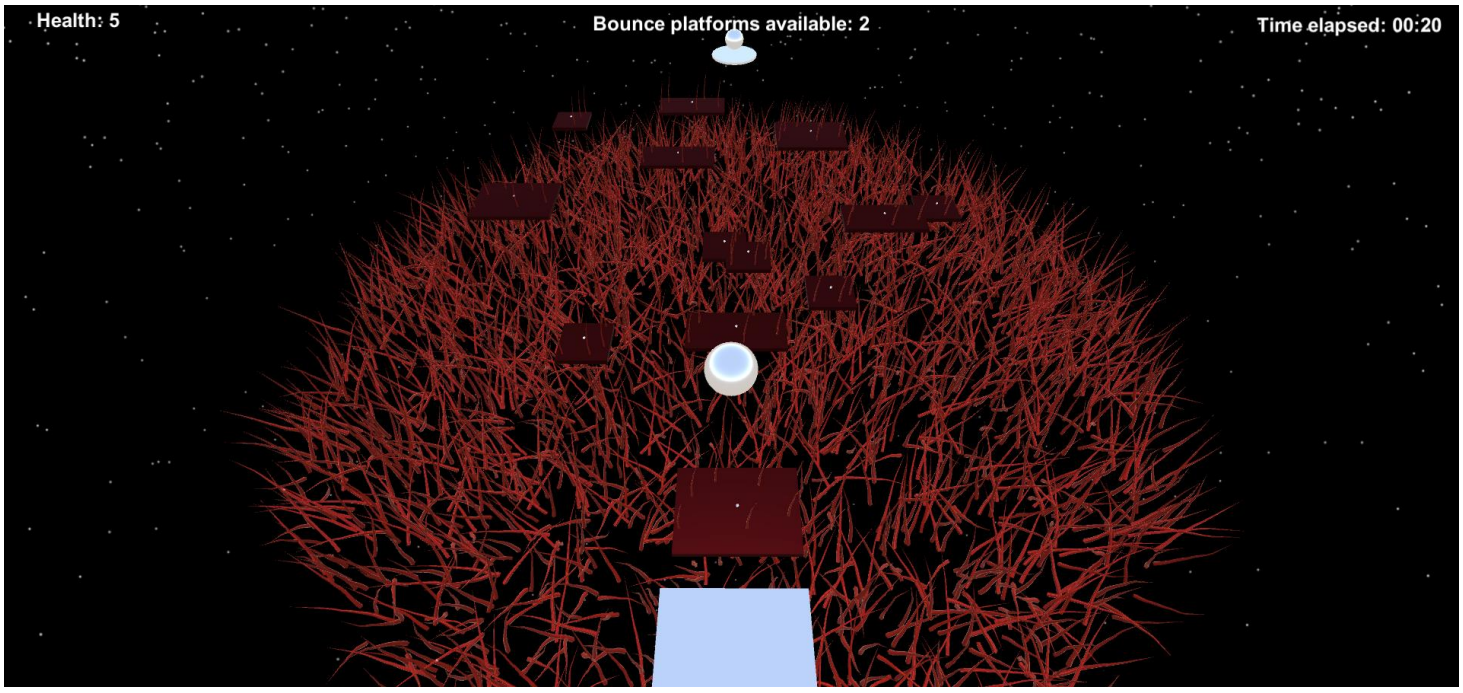


Table of Contents

Game Overview

Philosophy:

This game intended to use little to no external assets, instead creating the world with basic shapes and tools.

Common Questions:

What is the game?

This game is a 3D platformer made in the Unity engine. It contains 360-degree player and camera movement, challenging platforming, creative freedom in choosing how you complete the levels with a soundtrack and sound effects.

Why create this game?

This game is created for the purpose of continuous assessment of a Games Engines module in a Games Design and Computing Degree and possibly as an entry to Games Fleadh 2019.

Where does the game take place?

The game takes places in strange voids of unknown space.

What do I control?

You control a small ball of energy that is trying to escape from the void and return to the matter stream.

How many characters do I control?

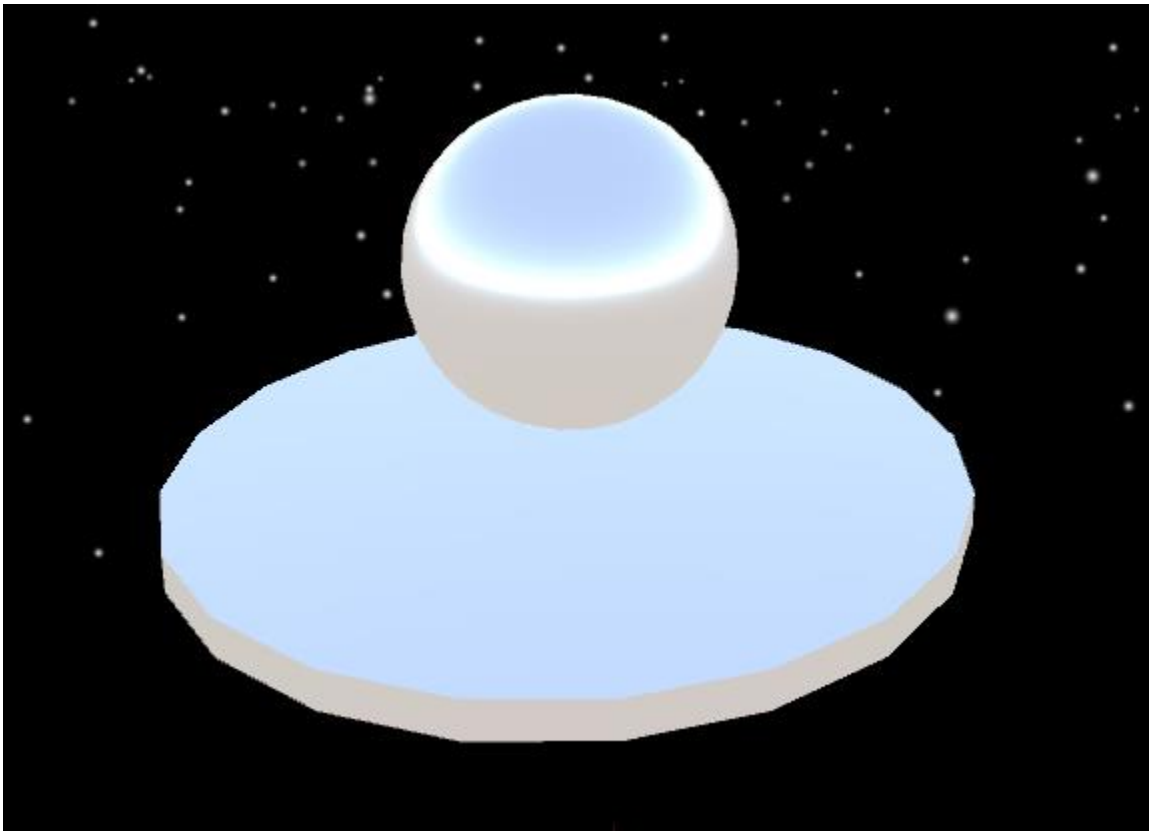
STARLOST is a single player game in which you control one character.

There are two ways to control the character, either using Keyboard and Mouse or with an Xbox 360/One controller.

Controls:	Keyboard/Mouse:	Controller
Jump:	Space	A button or Right Bumper
Generate Platform:	E	Left Bumper
Player Movement:	WASD	Left Analog Stick
Camera Movement:	Mouse	Right Analog Stick

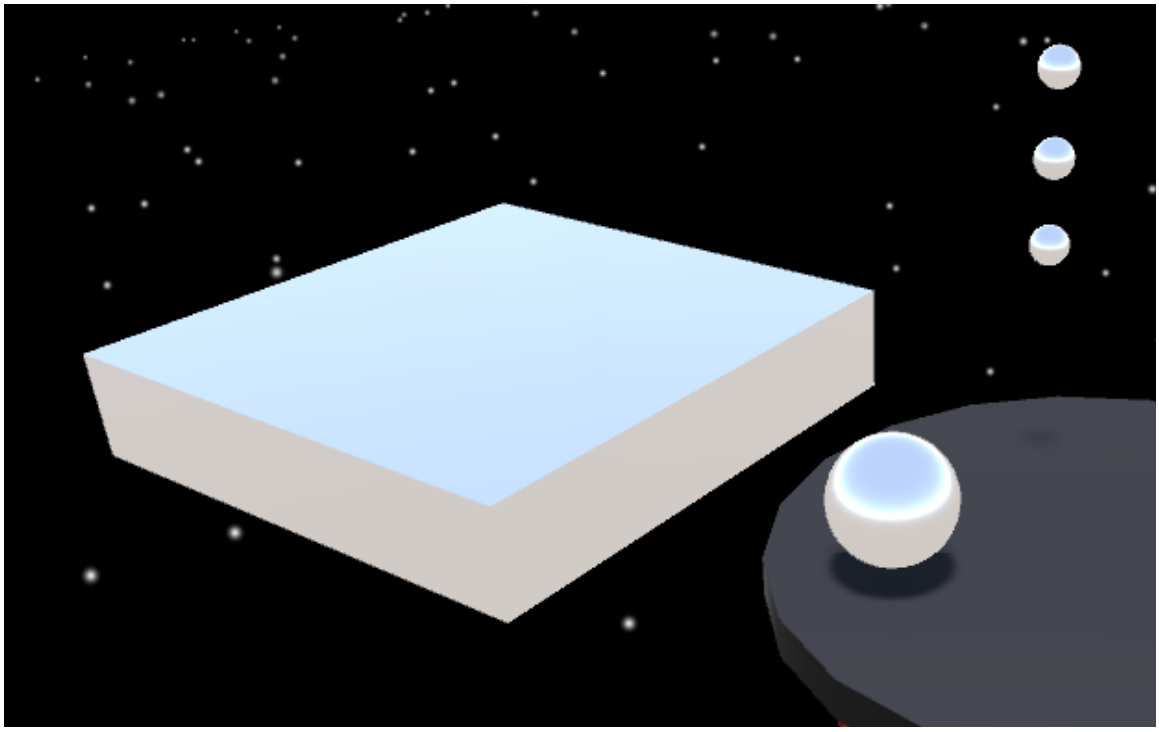
What is the main focus?

The main focus in this game world is to collect three large energy cores to progress through the areas and escape from the void.

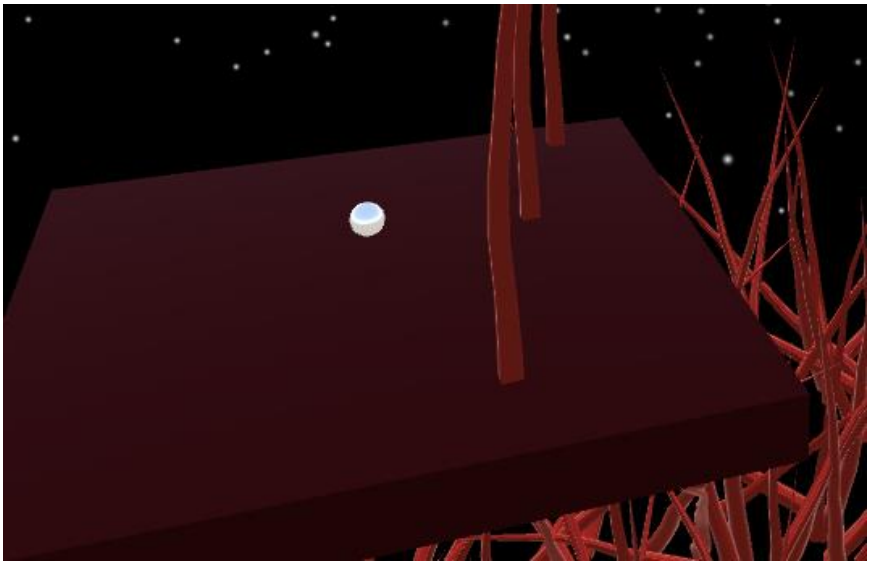


What's different?

To aid the player in traversing the level they are able to pick up and use smaller energy cores. These smaller energy cores allow the player to generate platforms underneath themselves that will bounce them up high into the air.



I feel that this is innovate in that it allows the player to construct the level before them as they play the game, making choices and applying some strategy in how they decide where to place the platforms and how to go about picking up the smaller energy cores.



Feature Set

General Features

Main Menu.
    Ability to play or exit the game.
Story and control overview.
    Shows the backstory and how to control the character.
Three levels of increasing difficulty.
    Level 1 is a straight forward path.
    Level 2 is straight and slightly elevated.
    Level 3 is a climb up to the goal.
Player jumping and movement.
    Full 3D movement and air control.
Camera control.
    Full 3D camera control.
Looping game system.
    When the end is reached the last part of the story is shown and the game restarts.

Gameplay

The core gameplay revolves around traversing from platform to platform, collecting pickups that allow the player to spawn platforms under themselves. These platforms, when collided with, will bounce the player up high into the air.

The intention was to create a game that is part platformer, part creative and part puzzling. The player can take as long as they want amassing pickups to be able to spawn lots of platforms, exploring as they go along.

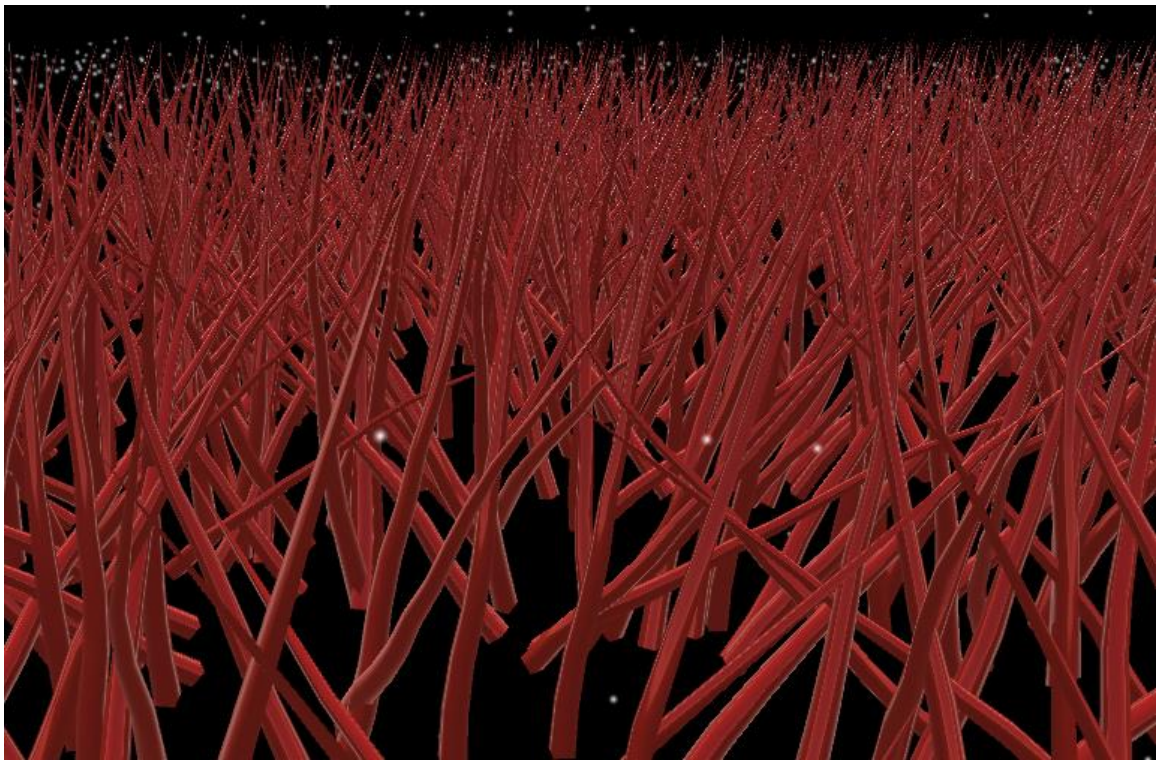
The Game World

Overview

The game world is a strange and expansive void to which the player is teleported to.

Void Tendrils:

Acting as the death plane in the game, the red tendrils attempt to impart some sense of wonder and danger to the player.



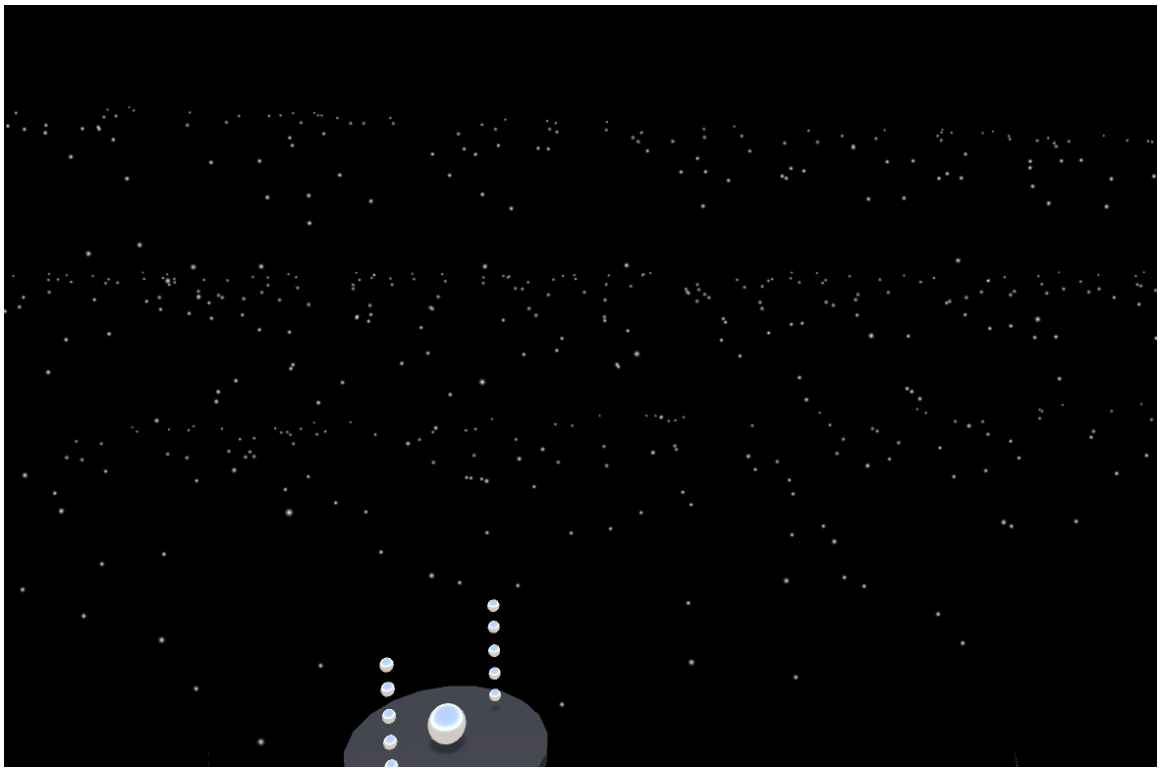
They are created from Unity's built in "Tree" 3D object. Their position and rotation are randomized inside a circle at when they spawn in at the start of a level.

```
void placeTrees()
{
    float yRotationMin = -100;
    float yRotationMax = 100;

    // spawn a number of trees
    for (int i = 0; i < number; i++)
    {
        float yRotation = UnityEngine.Random.Range(yRotationMin, yRotationMax);
        float zRotation = UnityEngine.Random.Range(yRotationMin / 2, yRotationMax / 2);
        float xRotation = 0;

        // Spawn the "trees", add them to a list for later rotation.
        trees.Add(Instantiate(tree, generateInsideCirlce(), Quaternion.Euler(xRotation, yRotation, zRotation)));
    }

}
```

Stars:

Acting as the backdrop and context for the world are stars that move out from the center of the play area. These “stars” are generated particle effects that are configured with a start lifetime, an emission rate, a shape and a size over lifetime.

There are four separate generators that hold differing x and z rotations between 10 and -10.

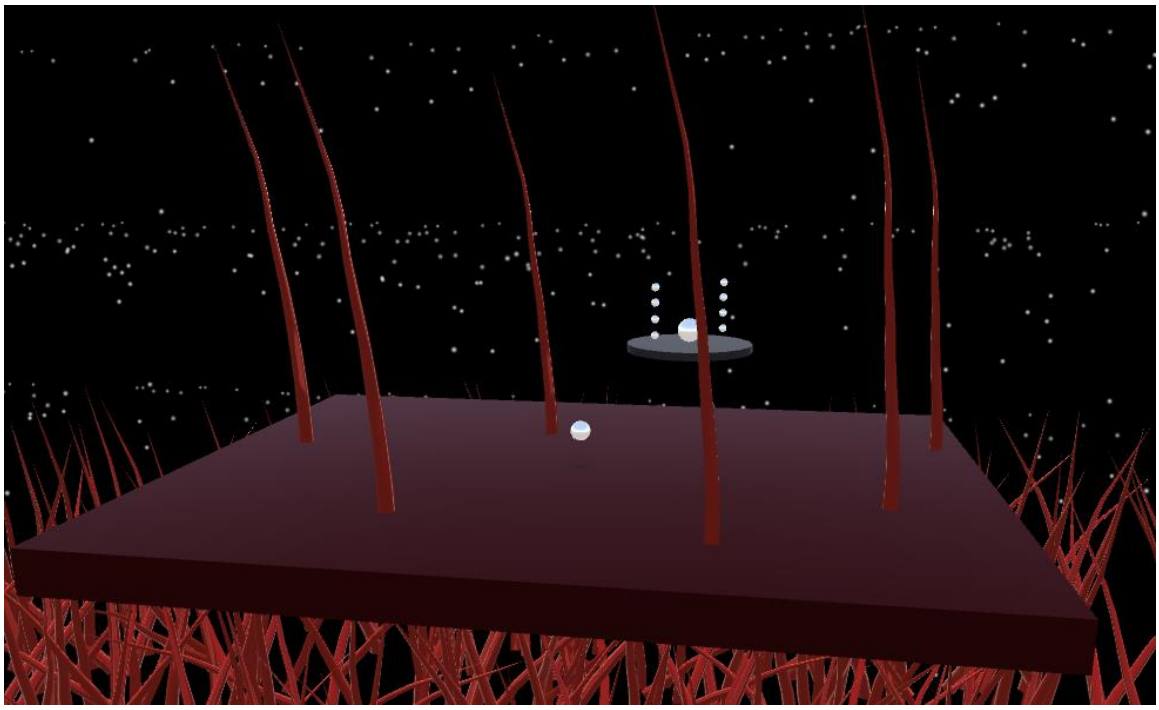


Platforms:

The main interactable physical entity of the world are the platforms that the player uses to get from the start to the goal.

These platforms have four size variants, and each has three different types of movement available with different speed of movement and time of movement.

The “Movement Amount” is how much distance the platforms move in a second. The “Time To Move” is the total time the platform should move for and the “Move ID” dictates on which axis the platform moves, either up and down, left and right or forward and backward.



The starting platform, also seen above in figure x is where the player lands when they initially begin the level and fall down.

This platform, with pickups, is programmed to always spawn below the player, no matter where they move while in free fall. If the current level isn’t the last level, more starting pickups are available.

The platform is spawned by taking the players position with a -10 y value when they have fallen to just above the death plane of tendrils.

The pickups are then spawned relative to where the platform is going to spawn.

```
// spawn the first platform as the player falls
public void spawnFirstPlatform()
{
    // calc the platform position to spawn appropriately with player.
    Vector3 platformPosition = new Vector3(player.transform.position.x, player.transform.position.y-10, player.transform.position.z);

    // store the inital pickup positions;
    float seperationValue = 2;
    startingPickUpPositionOne = new Vector3(platformPosition.x + seperationValue, platformPosition.y + 1, platformPosition.z + seperationValue);
    startingPickUpPositionTwo = new Vector3(platformPosition.x - seperationValue, platformPosition.y + 1, platformPosition.z - seperationValue);

    Quaternion noRotation = Quaternion.Euler(0, 0, 0);

    // spawn the platforms and pickups.
    Instantiate(firstPlatform, platformPosition, noRotation);
    Instantiate(pickup, startingPickUpPositionOne, noRotation);
    Instantiate(pickup, startingPickUpPositionTwo, noRotation);


    // Spawn some more if we aren't on the last level.
    if (!isLastLevel)
    {
        startingPickUpPositionOne.y += 1;
        startingPickUpPositionTwo.y += 1;

        Instantiate(pickup, startingPickUpPositionOne, noRotation);
        Instantiate(pickup, startingPickUpPositionTwo, noRotation);
    }
}
```

Camera

Overview

The camera is controlled with the mouse or the right analog stick of the controller. The camera can look in all directions and gives the player the ability to see how they can navigate the world.

Game Engine

Overview

The game is built using the Unity 3D engine. This engine provides lighting and rendering, collision detection, world building, audio, default assets such as shapes and particle effects as well as many other features useful for a developer.

Game Characters

Overview

The main playable character is a ball of energy that the player controls. He is lost in space and must traverse the three levels to escape.

Enemies and Monsters

The tendrils that are placed on platforms serve as the game's enemy. If the player comes in contact with these the player is knocked back and 1 point of health is taken away.

User Interface

Overview

The user interface is designed to provide some information to the player in the form of their health, the number of platforms they can spawn and the time that has passed since the start of the level.

On level 1 there is a small hint at the bottom of the screen prompting the player to spawn a platform. This hint goes away after the player has spawned their first platform.

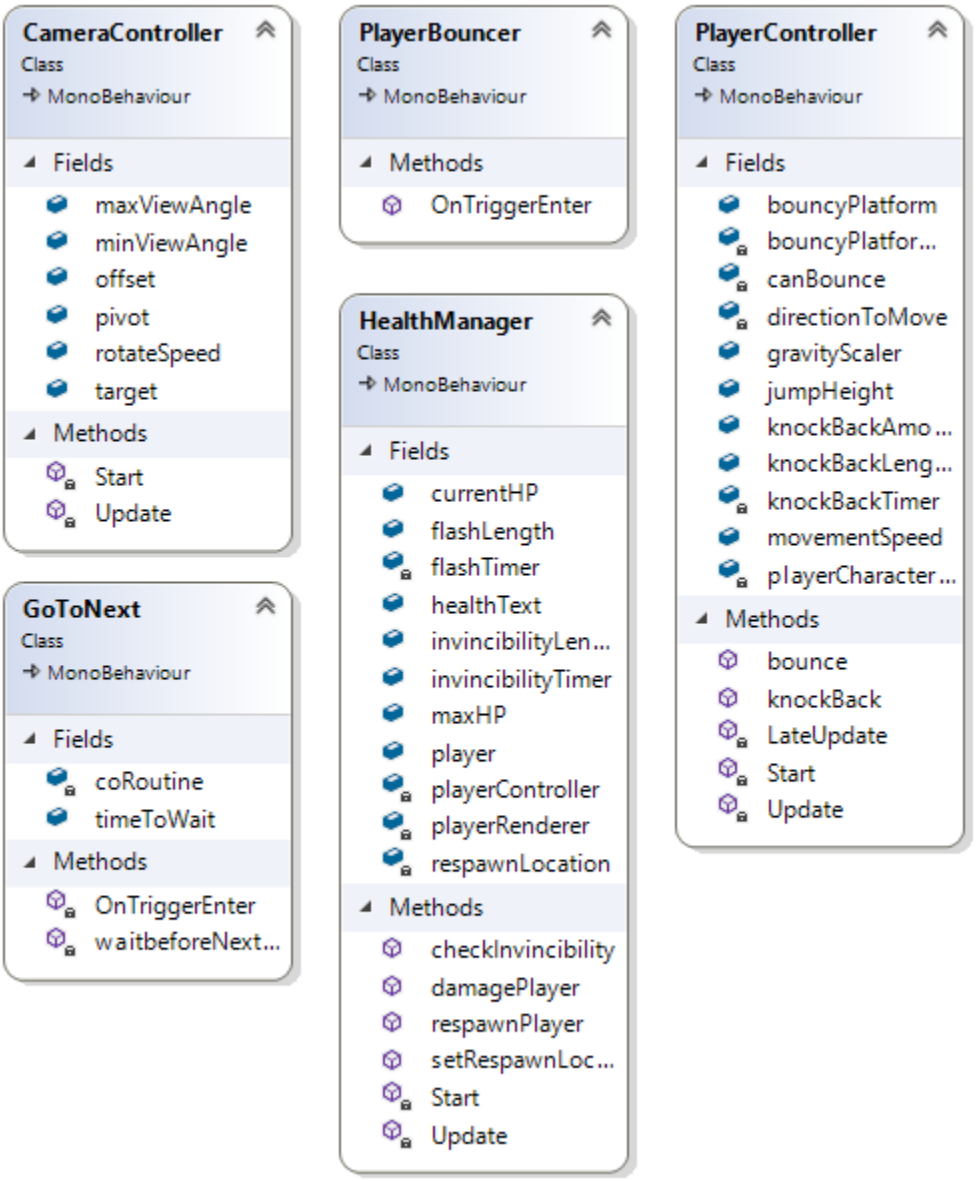
Musical Scores and Sound Effects

Audio.

As a background track there is a somber space themed digital instrumental.

For the jump and bounce sound effects, there is a classical retro arcade jump sound. For the pickup sound there is also a retro arcade powerup noise. These assets were sourced from https://opengameart.org/content/512-sound-effects-8-bit-style.

Class Diagram A.



Class Diagram B.