### Technical Design Document Template

1.0 Revision History

<As you revise the document, list what was changed and when it was changed>

|  |  |
| --- | --- |
| Version | Description |
| 1.0 | Initial document |
| 1.1 | Adding Diagrams |
| 1.5 | Redesigned AI Paddle Navigation |

2.0 Development Environment

2.1 Game Engine

Unity 2019.3.6f1

2.2 IDE

Visual Studio 2019

2.3 Source Control procedures

Push changes w/ GitKraken when finished

2.4 Third Party Libraries

2.5 Other Software

3.0 Game Overview

3.1 Technical Goals

AI that can complete the obstacle course with modifiable precision

3.2 Game Objects and Logic

Player, AI, Jump Platforms, Skinny Bridge, Pusher, Paddle, Trap Door, Wrecking Ball, Goal

3.3 Game Flow

Player starts game and sets AI difficulty, after which the player will be able to use wasd space and mouse to move and look around, achieving victory by getting to the end before the AI and loses if the AI gets there first

4.0 Mechanics

Move with wasd

Jump with space

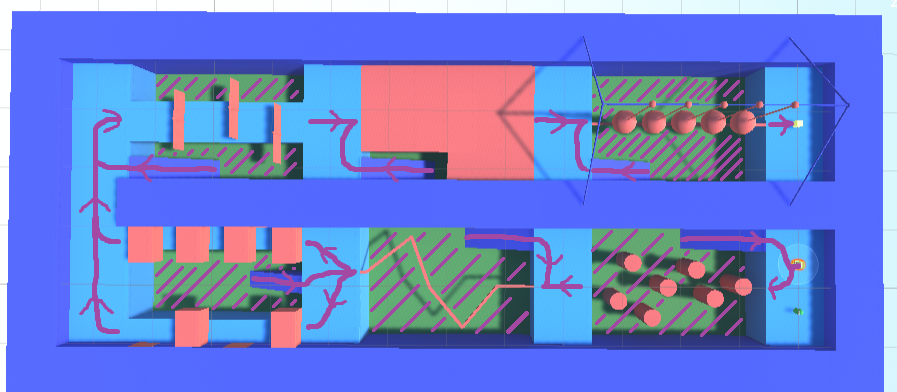
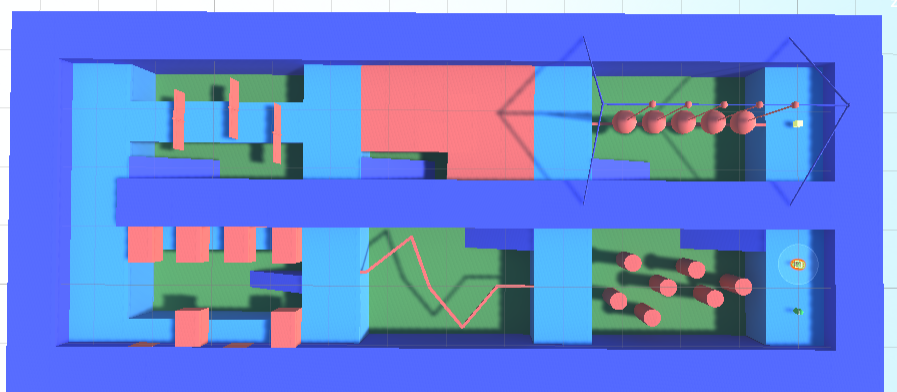
Look with mouse

5.0 Graphics

First Person 3D

6.0 Artificial Intelligence

I plan to use Unity’s built-in navmesh to travel to the start of each puzzle then utilize custom AI to navigate each obstacle as will be detailed below. Whether to utilize navmesh and custom AI will be as unique clauses for each obstacle with the general idea being get to node that begins custom AI using navmesh then deactivate navmesh agent to transition into custom then return back to navmesh on completion or failure of obstacle. Failure will likely be determined with three factors, the AI is on the navmesh, the AI is grounded, and the AI’s y position is below -1. The AI will have three difficulties: Beginner, Intermediate, and Expert. Beginner will have half to three quarters the player’s speed, high chance of error on precision-based movement, and takes the slower path. Expert will have the same movement speed as the player, very low chance of error on precision-based movement and takes the fastest path. Intermediate will be in-between the two and there will be obstacle specific factors that the AI difficulty will play into.



6.1 Jump Obstacle

Calculate path across using Dijkstra based on AI difficulty, lowest takes longest path, highest takes shortest path. Then Set the AI’s forward direction to face the next node and set the velocity of the AI’s Rigidbody to the vector specified in the node plus a random offset to give the AI a chance to fail. Success will be determined by making it through to the end of the path.

Graphical user interface

Description automatically generated

6.2 Skinny Bridge

Have the AI seek steer towards the corners with a wander offset. Success will be determined by making it through to the end of the path.

Chart, line chart

Description automatically generated

6.3 Pusher

Pusher will have to different paths an easy and a hard path. Beginner AI will always take the easier path, and the expert will always take the harder path, intermediate will have a chance to take either.

6.3.1 Easy Path

AI will navigate to the first node in the path, then wait until pusher is out of the way before moving forward to the next node and repeats until the AI makes it to the end of the path.

6.3.2 Hard Path

AI will navigate to the first node then turn 45 degrees to the right and wait for the first pusher to reach it’s max extension then travels forward until it makes it to the end of the pushers marking the success when the position.z exceeds a certain threshold.

A screenshot of a computer

Description automatically generated with low confidence

6.4 Paddle

Use the navmesh to calculate the path every frame then with said path have the agent Seek Steer towards corner[1] until through using rigidbody velocity

A picture containing text

Description automatically generated

6.5 Trap Door

Path will be calculated using Dijkstra the AI will then seek steer the path, if the AI falls through a trap door, the AI will have a percent chance to remember the tile they fell through: 50% for beginner, 75% for Intermediate, 100% for Expert. That tile if remembered will be marked as impassable in the Dijkstra calculation.

A picture containing text, sky, clock

Description automatically generated

6.6 Wrecking Ball

Functionally the same as Skinny Bridge, only its a straight line and there are wrecking balls to knock off the AI.

A picture containing sky, light

Description automatically generated

After making it through the wrecking ball the AI will just navigate to the goal and collect it to win.

7.0 Physics

Colliders and RigidBody, Gravity and such is custom coded in all instances

9.0 Game Flow

9.1 ‘Mission’ / ‘Level’ structure

Everything done in one scene, so I can use the map itself as a background

9.2 Objectives

Get to goal

11.0 Interface

11.1 Menu

Play, Difficulty, Quit

11.2 Camera

Perspective, start as over top menu, then move to first-person, then finally back to over top to show victor

11.3 Controls

Keyboard and Mouse

14.0 Asset List

Scripts, materials, scenes, animations

16.0 Technical Risks

The AI does have a small risk of escaping the computer and hunting the player down however that will be minimized in the code so it should only happen to 1 in a thousand players