|  |
| --- |
| EEL Encorporated |
| Amazein Snek |
| Technical Design Document |

|  |
| --- |
| Joel Goodchild and Edward Ladyzhenskii  9-5-2017 |

# Game Concept:

Amazein Snek is an endless puzzle game that pits the player against yourself in a challenging world filled with twists, turns and your tail. The goal of Amazein Snek is to collect as many cubes as possible before running out of space for your ever growing tail. The player will find it difficult to traverse the map with a lazy tail that lags behind the player.

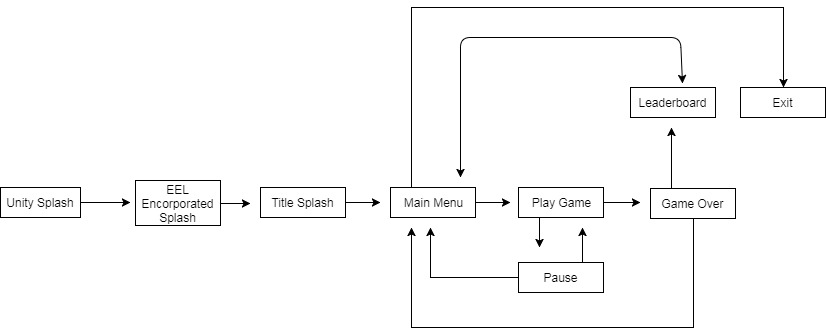
# Features:

* Basic player movement
  + Left, Right, Forward. No backwards, but you can turn around and retrace your tracks.
  + Only able to turn once Snek has some velocity.
* Lazy tail
  + Tail that grows with every cube, following Snek, making it difficult for the player to make its way around corners.
* Cube collection
  + Speed boost with every cube consumed.
* Deceleration
* Cannot touch walls, or game will end
* In level tutorial posted on floor

# Technical Risks:

* Tail trailing may cause some issues
  + Balancing angles to make it playable will be difficult.
* Player speeding up
  + Must speed up gradually, not too fast, and not too slow, may be hard to balance.
* Using the FollowMe asset may provide some difficulties
  + As we have had little experience with the library, it may be difficult to implement.

# Game Flow:



# Game Objects:

Snek – Snek will have a player control, similar to that of the car controller, as it cannot turn without already having speed. Using a custom player controller, Snek will include a list storing each of its cubes. If Snek collides with a cube, that cube will disappear and add length and speed to Snek. If Snek collides with a wall, the end game scene will trigger, blowing up Snek.

Progression Cubes (Placeholder name) – Progression cubes will spawn on random spawn points, which will be chosen from an array of empty game objects with locations that are not too difficult for the player to access. The cubes will be hovering and rotating to make them more desirable for the player, and the spawn point in which they are located will be depicted with a particle effect.

# Input Method:

PC –

* W/Up Arrow = Move Forward
* A/Left Arrow = Move Left
* D/Right Arrow = Move Right
* ESC = Pause Menu open
* Mouse Location and Left click = UI control

PS4 –

* Left Analogue Stick = All movement and UI selection
* X = UI Selection
* Square = Back (When in menus)
* Start = Pause Menu open

PS Vita –

* Left Analogue Stick = All movement and UI selection
* X = UI Selection
* Square = Back (When in menus)
* Touch = UI button selections
* Start = Pause Menu open

Android –

* On screen joystick = All movement
* Touch = UI button selections and Pause menu open (with pause button)

Nvidia Shield –

* Left Analogue Stick = All movement and UI selection
* A = UI Selection
* B = Back (When in menus)
* Start = Pause Menu open

# User Interface:

# 

# Deliverables:

Amazein Snek will be a fully functioning 3D puzzle game, fully supporting a range of platforms, including PC, PS4, PS Vita and Nvidia Shield. Amazein Snek will be a challenging yet fun game that people won’t want to stop playing. It will have a full progression system with the cube collectibles being what Snek has to collect, and will include a leaderboard, so that the player will desire to beat their best score, as well as their friends.

# System Requirements:

PC –

* OS: Windows XP SP2+, Mac OS X 10.9+, Ubuntu 12.04+, SteamOS+.
* Graphics card: DX9 (shader model 3.0) or DX11 with feature level 9.3 capabilities.
* CPU: SSE2 instruction set support.

Android and Shield –

* Android: OS 4.1 or later; ARMv7 (Cortex) CPU with NEON support or Atom CPU; OpenGL ES 2.0 or later.

PlayStation 4 or PlayStation Vita console.

# Third Party Tools:

* Unity 2017
* FollowMe Asset pack
* Autodesk Maya 2017

# Coding Conventions:

* CamelCase
* Variables and function names will always begin with a capital letter, with the first letter of any new words being capitalized also.
* Braces will be place on their own line below functions or wherever else they are required.

# Source Control:

* We will be using SVN and GitHub to control the project. Each time there is a commit, there will be a comment stating who has added it, and what has been added in the update. Commits will only contain the Assets and Project Settings files.

# Team Members:

* Joel Goodchild
  + Level design and character movement
* Edward Ladyzhenskii
  + Model creation and world mechanics (physics etc.)