TECHNICAL DESIGN DOCUMENT: MAZE GAME

A Maze Game

Change History

Change Log

Date	Author	Changes
26/08/2022	Blake	Created Document
07/09/2022	Luke	Added Main Menu Content

Roles

Name	Role(s) / Tasks
Blake	**Designer / Programmer** Level Design UI
Jason	**Programmer** Movement
Luke	**Programmer** UI

Contents

Project Overview

Game Concept

Simplistic maze game with two moveable characters, when one character is moved the other will move in the opposite direction. With the goal to have both characters escape a maze.

Technical Goals

To create a simple maze game making the player think about two possibilities with every action having an opposite reaction with stable gameplay.

System Requirements

Platform	Requirements
PC	https://docs.unity3d.com/2020.1/Documentation/Manual/system-requirements.html#mobile
Web	https://docs.unity3d.com/2020.1/Documentation/Manual/system-requirements.html#mobile
Mobile	https://docs.unity3d.com/2020.1/Documentation/Manual/system-requirements.html#mobile

Technical Risks

- Characters getting stuck in the environment

Third Party Tools

Unity
Visual Studio
Gameplay

The Environment

The game consists of one playable area that is split down to the middle to separate the characters. Each character will have a small playable area filled with obstacles blocking the path to the exit. Switches will be located throughout the level, when a switch is hit it will open up a path for the other character to progress.

Game Structure / Game Objects

Character Information [Scriptable Object]

A scriptable object used to store the information for each character.

Determines if they are the active character and will change their direction and material based on this.

Character Controller [Component]

A component is put on each character to update movement and rotation.

Runs a check on start that will change the material depending if they are the active character or not. Regularly updates the characters rotation and position to face their target. Runs checks to ensure the character is a distance away from the target, to prevent clipping as well as denies the player the ability to move the character until they have reached their destination.

Controller [Component]

A component used to determine the direction and rotation of each character.

Includes a list of the characters in the scene and does raycasts based on player input for each of them. Afterwards sends the information of their targets to each character for them to move to.

Physics (raycast)

The player's input using the WASD and arrow keys on the keyboard(swipe or UI elements for mobile), determines the direction the raycast will be sent relative to each character's environment. For the purpose of flexible programming, the script controlling both of the characters will not using Input.GetKeyDown(KeyCode.{Key}) will not be used. Instead the direction will be controlled using Input.GetAxisRaw("Horizontal" / "Vertical") to modify float values for up, down, left and right.

Level of Detail

The game will not include heavy levels of detail, having its focus being on puzzle solving but will feature pre-built assets that give colour to the game.

As the player progresses, the levels will become more complex. Incorporating more mechanics and changing the level design to introduce more challenge.

Since the game will not be graphic intensive, we have the option of including multiple levels in a single scene without performance dropping significantly.

In Game Controls

Using variables to detect player input instead of set keys being detected we can simplify creating controls that work for different platforms. The direction the characters will move and rotate to is determined by vertical and horizontal variables that are modified using Unity's built-in keywords "Vertical" and "Horizontal".

Desktop / WebGL

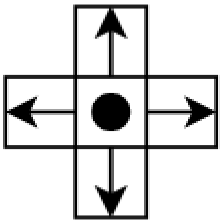
Mouse and Keyboard

The previously mentioned key words primarily detect input using a keyboards W,A,S,D keys, the arrows key as well as a controllers joystick input direction.

Mobile (Android / iOS)

While in the early stages of development, the mobile controls will consist of manipulating the vertical and horizontal variables using buttons that are placed on the UI.

Once other important mechanics have been implemented, the mobile controls will be shifted to using touch screen swiping to control the characters movements.



Menus

Main Menu

Title - RGB slowly changing glow colours controlled by code in SETTINGS script. Include a nice font with some material effects for presentation.

Background - Animated colourful cubes moving on a loop behind the menu. These cubes are controlled by “LeanTween” Animations in the EASING MANGER script. Created in a 3D space and set the camera up to view from top down. Included a blur shader and glow effects for presentation.

Menu Options - All menu options are layed out using the FLEXIBLE GRID LAYOUT script and assigned all the options to the TAB GROUP script which is used to control the panels that appear when each option is highlighted. Each option has the TAB BUTTON script which takes in the point event data from the mouse and controls a simple tweening animation to give the user more feedback when they are selecting an option. There is a MENU NAVIGATION script to control the functionality when the user selects an option.

Panels - The panels are layed out with the FLEXIBLE GRID LAYOUT script and each panel is assigned in an array in the PANELS GROUP script which controls what panel appears when menu options are highlighted or selected. This script also controls a small tweening animation to display each panel when required.

Settings - All the settings options are controlled in the SETTINGS UI script. These features include background music volume control, a mute toggle, a full screen toggle, basic graphics settings and screen resolution options. All these options have their own function in the script. Each selected panel contains a back button to close the panel if the user wants to.

Options Menu

The options menu consists of a main menu button, a reset button and a restart button.

Main Menu- the main menu button return the player to the starting screen when pressed and which is located in a different scene.

Reset- Pressing the reset button will reset the characters positions to the current checkpoint

Restart- When the restart button is pressed, it will completely reset all the characters and cameras positions back to their starting positions, reset the coin count and the timer.

In-Game Hud

The in game hud consists of a timer, coin counter, options button and a display showing which character is currently moving correctly.

On mobile the hud also consists of the controls and a button to switch which character is active.

List of Assets

Standard Assets Pack.

TextMeshPro.

LeanTween

Schedule

Milestone	Features	Due Date	Contributors
1. Proof of concept	Character Movement UI mockup Basic Mechanics	26/08/2022	Jason Luke Blake
2. Prototype	First level UI implemented	2/09/2022	Jason Blake Luke
3. Alpha	WebGL build PC build	9/09/2022	Jason Blake Luke
4. Beta	Mobile controls Mobile build	16/09/2022	Jason Blake Luke

Strategies for monitoring production progress

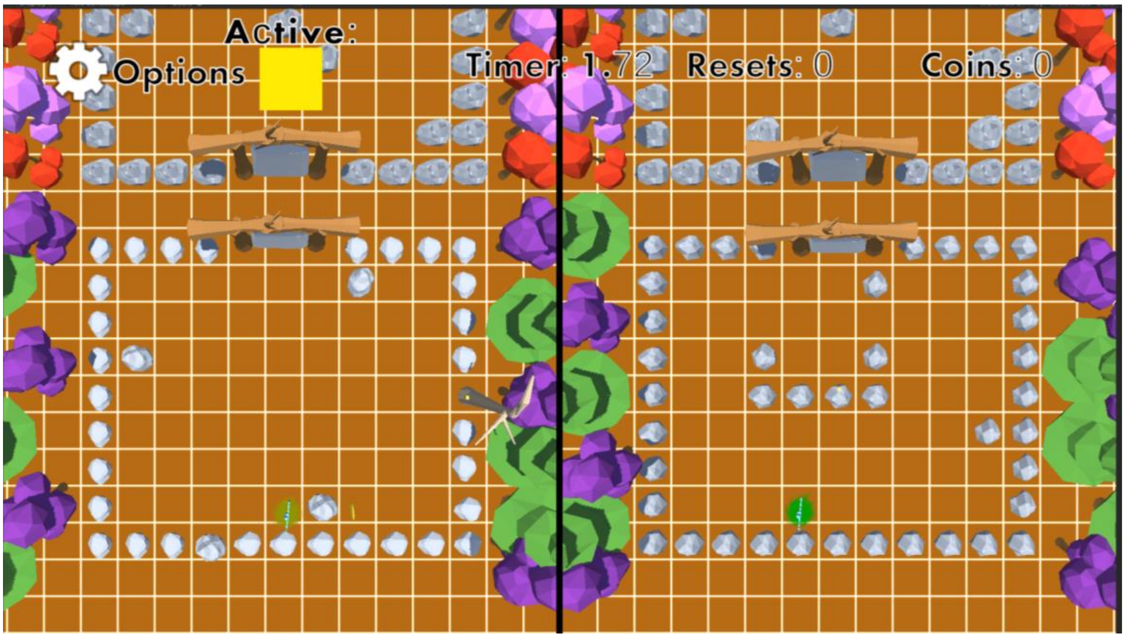
Regular team meetings.

Team chat platform such as Teams or Discord.

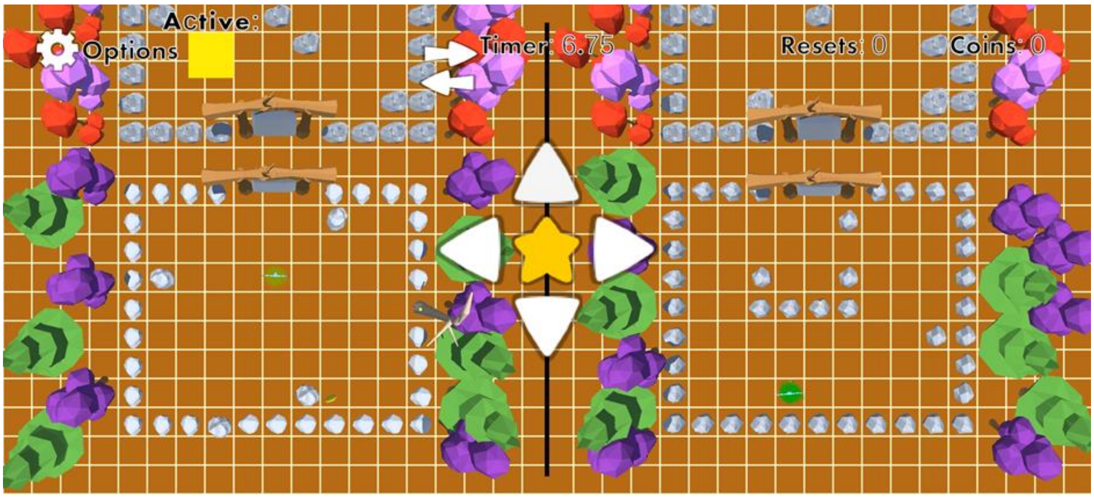
Project management tool, such as Trello and Emails.

Builds

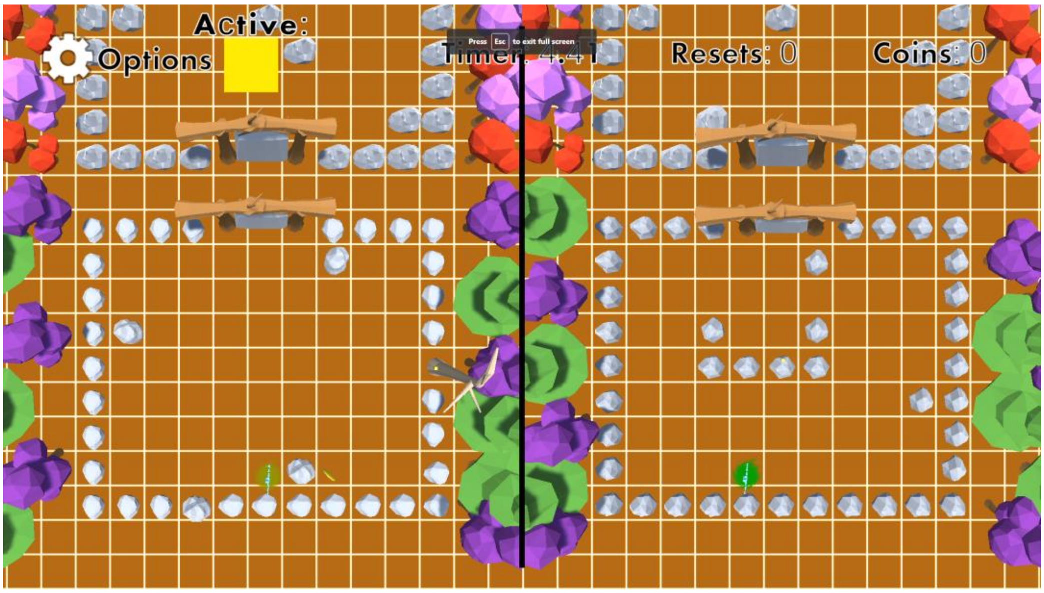
PC:



Mobile:



Web:


Active:
Options
Press Esc to exit full screen
Timer: 4.41
Resets: 0
Coins: 0