Concept

In this project, I will be developing an original game that will derive from two game genres, maze and puzzle-solving. The game will contain several levels there the player will have to manoeuvre through a series of corridors encountering puzzles and scoring points.

This nature of the game requires creative thinking and imagination to both play and develop. And as such should provide a chance to work with (or tweak) existing game mechanics and make (and explore) new once.

The game targets younger auditions, mainly children who look for a game with fun challenging and rewarding puzzles.

Concept Visuals

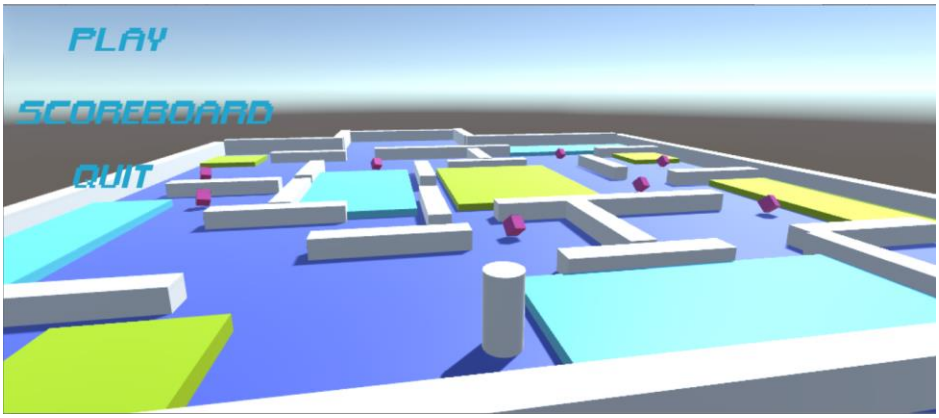

Figure 1: menu destine A

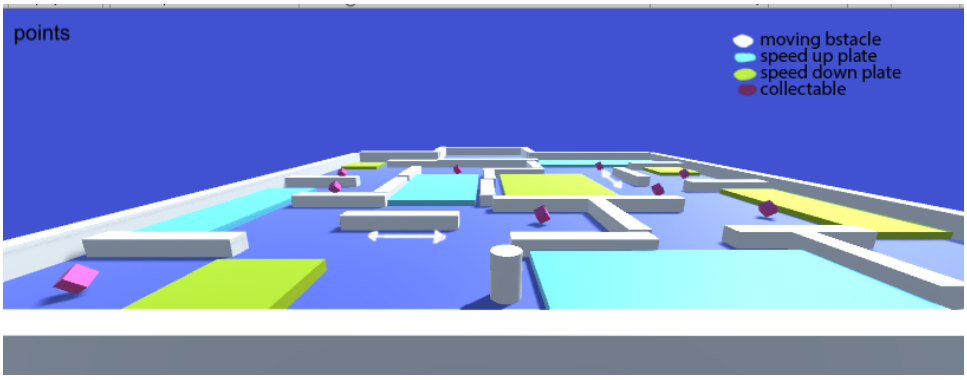

Figure 2: level destine A

Figure 3: level destine B

Requirements

Program behaviour requirements

*The main character will need to move in the x and z axis using the arrow or the WASD keys. He will also rotate around his centre to allow movement in all angels

* A camera will need to fallow and rotate around the character, in a way that can provide the player with a 360° view the sprawling area

* Collectables - when the player collides with one it will disappear and the player will gain a point.

* Scoreboard - the scoreboard will be a place for players to compare their scores. Points will be displayed in descending order.

*Sand pails - They will slow down the player.

*Ice blocks - As long as the player is on one, he gains a speed boost

*When collecting all the maxim amount of points for level 1 a final collectable will appear and a camera will focus on it and zoom out. Notifying the player where he should be heading next. Up on coalition with the player and the final collectable the floor under the player will be removed. Letting the cylinder/player fall on to the next level.

*Shooting - When the player collects 11 points, he will unlock the ability to shoot.

*Destructible walls - Will be walls that the player will be able to shoot down.

*Plate triggered restrictions - those will be walls that will be lowered when a player completes a puzzle. The puzzle might be as simple as moving a cube from point A to point B.

*Control swap -- by shooting a cylinder the player can switch controls between his current character and the target.

*Flying- towards the end the player will be granted the ability to fly.

*Gravity - as the player will move in the y-axis gravity needs to be implemented. Also, gravity will be used as a way to progress through levels, I.E. by removing the floor and letting the player dorp down to the next level.

*Messages - the player will be notified when a new ability is unlocked

* Navigation between menus and scenes - a menu screen will be the first thing the player will encounter and interact with. From that screen, he could either proceed to the scene with the game, access the scoreboard or quit the application. At the end of the game, a scene with a second menu will be displayed that will provide the options to play the game again (restarting the previous scene) or returning to the main menu.

Assets Requirements

* Two characters that the player will be able to control. These can be cylinders with a character controller and a collider

* Walls - walls will need to be added to establish level boundaries and author obstacles will be added to create paths the player can walk in, to help sell the maze feel. Different wall designs will be needed for the destructible walls and the once connected to a trigger.

* Collectables - these can be small cubes scattered around the level.

* Bullets - these will be balls that will fly off the player in the direction the player is facing.

* Sand pails and ice blocks - these will most likely be resized cubes with deferent colours.

* Rings - when the player unlocks the ability to fly rings will be used to mark the direction the player will have to go in. The rings can be taken from the standard assets package.

* Notifications - when a new ability is unlocked the player will be notified. The notification could be as simple as text on the screen or it could contain an image of what the ability is alongside a short description complemented by some visual effects.

Analysis and design

Functionality

The game will implement a lot of features and mechanics. Some features, like the sand pails and the ice blocks, should not cause much trouble. Once I know who the player will move throughout the game, however, others will require more patience and creative thinking. Like the shooting for example. A projectile, probably with a rigid body, will have to be instantiated with a certain velocity and direction. The bullets will also have to be destroyed at some point to prevent the game from taking more resources than it should be. And that's even before we can start working on how the environment will react to those projectiles. Such as the destructible walls, the trigger cubes and the other playable character.

The *destructible walls* will have to withstand a couple of hits before they vanish.

The *trigger cubes* will have to move when shot. and when they are placed in a particular area that should trigger a door opening event.

The other object the player will be able to take control over will be the trickiest to implement, as to do so I will have to make sure that only one of them moves at a time. The swap of control between the two is trigger when the one is shot. I will also have to figure out how the camera will react to this change. An easy solution will be to attach separate cameras to both objects and have CameraA be

active when the player is controlling ObjectA and when he switches to ObjectsB, CameraB will be turned on, CameraA - off. I will also have to make sure the swap is reversible.

The Collectables will be removed from the screen when the player collides with them and the number of points the player has will be incremented by one.

The transition from level to level by removing the floor the player stands on can be done by positioning/layering the levels one over the other and ensuring that the player will fall by adding a constant downward force, to simulate gravity.

Activity Flow Diagrams

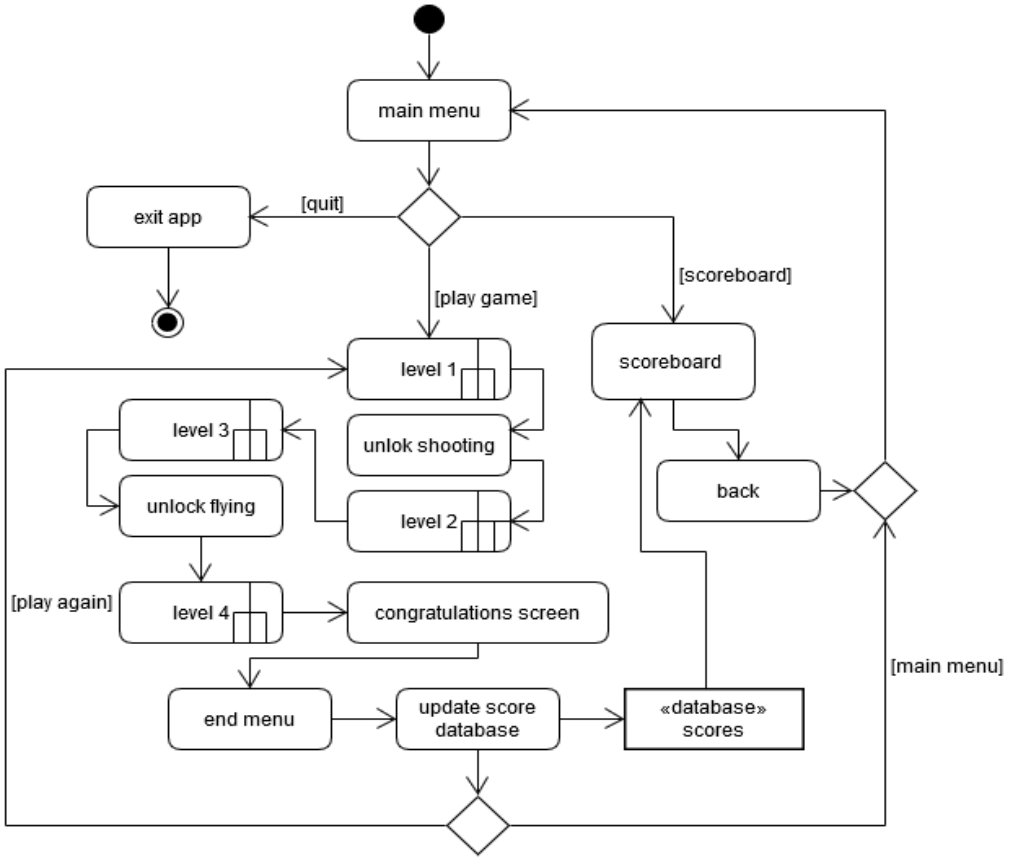

Figure 4: main UML

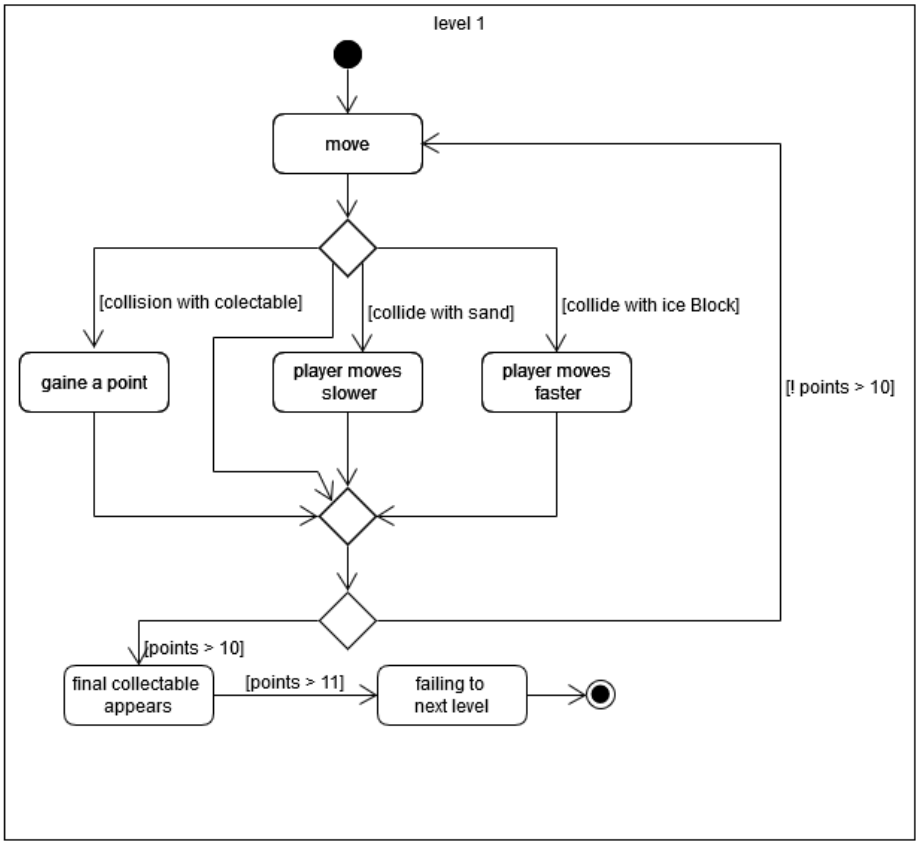

Figure5: Leve1

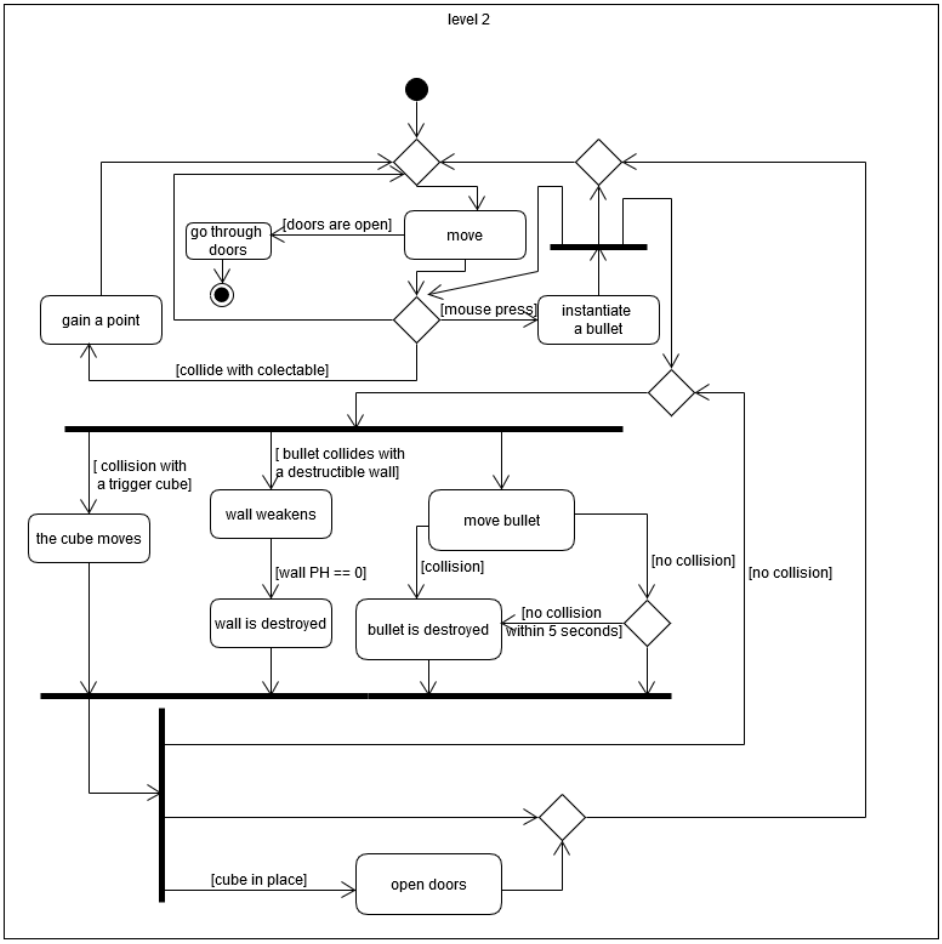

Figure6: Leve2

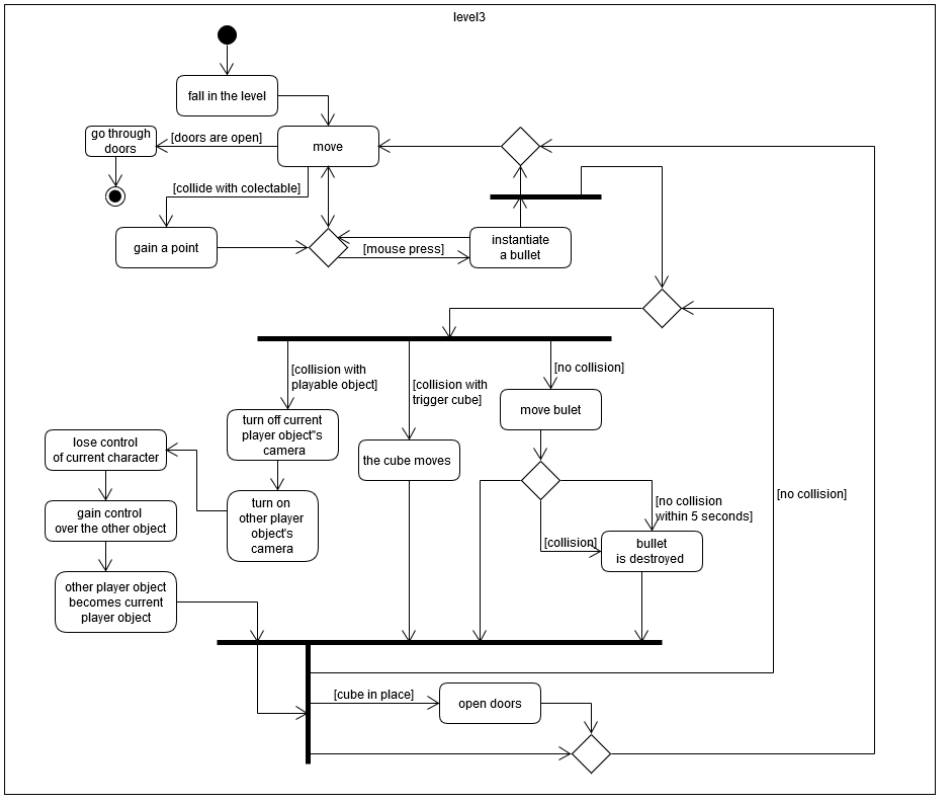

Figure 7: Leve3

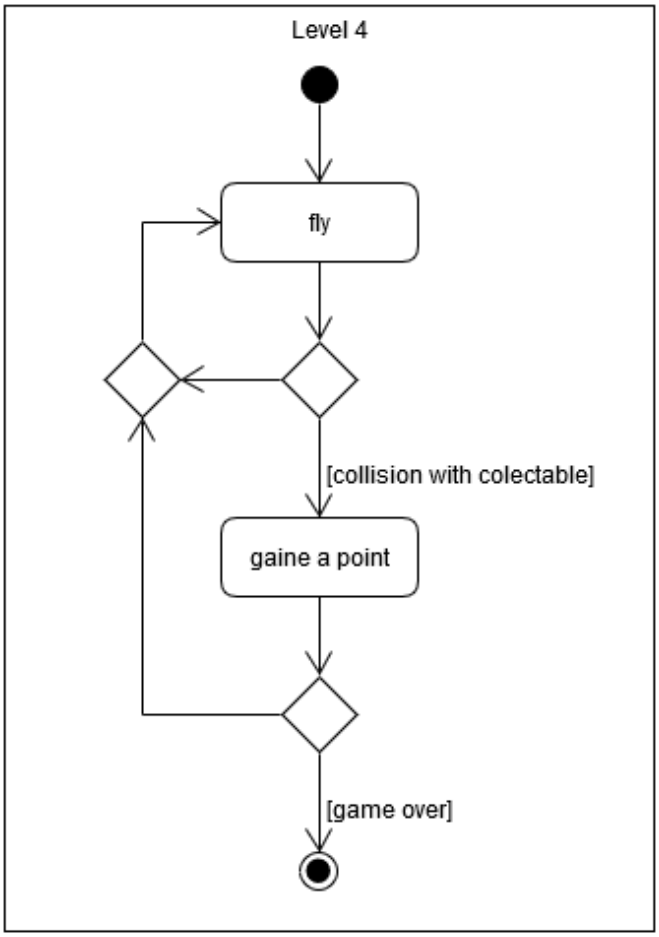

Figure8 : Leve4

Implementation

First, I made the menu with play, scoreboard and Quit buttons and connected them to on click events and functions. The play button loads the next scene in the project using the play() function in the UIButtons script. The scoreboard button brings a different menu that should display the player's scores over time in descending order, however, due to time constraints this feature was not implemented. The quit button uses the quit() function form the same script to close the application.

With the mine menu ready I started working on *controls* and the *walls*, *collectables and the* sand piles prefabs for level one. To allow the player to move I added an object with a character controller to which I attached a camera and cylinder so that we can see where the player is.

When working on the level boundaries I realized that if some walls move from side to side, predictably, they will block and unblock certain paths and that will introduce a timing element to the game, so that was added to the game.

Object	Component	My activity	Comments
player1	character controller	Used to move the player	
	collector	Used to handle collisions with collectables, sand, ice and power-ups	
	shooting	contains the shooting mechanic and also handle the player change event	
Main camera	transform	Follows and rotates around player 1	A child of player 1
aim (empty objects)	transform	Determents the spawn point and direction of the bullets	A child of player1 Reference 4 points to what the idea of Instantiating a prefab and applying a force to it was taken. Collision effects were implemented from scratch
cylinder	Movement (script)	Allows the player to move and fly and also its direction and speed I.E when colliding with ice or sand	A child of player 1. References3 points to what was used to create the basic movement.
	Meshrenderer	shows where the player is	
projectile	Box collider	Used to detect collisions	
UICanvas			

msgManager	MsgManager(script)	displays the unlockable and achievements	
camManager	CamComtrol(script)	Handles the direct camera changes	
ground	Material	Made in unity	
	Mesh collider	Used to prevent the player from falling through it	
fackGround		Disappears when player should proceed to the next level	
	Material	Made in unity	
pickUps(empty Object)		Each pickUps object contains the pickups for the level it is in	
pickUp	transform	Used to rotate the object	A child of pickUps
	Box collider	Used to detect collisions with player	
	Material	Made in unity	
walls (empty object)		Contains the	
W_R	Box collider	The right wall of the level	A child of walls
W_L	Box collider	The left wall of the level	A child of walls
W_B	Box collider	The back wall of the level	A child of walls
W_F	Box collider	The front wall of the level	A child of walls
maze (empty object)		Each maze object contains the moving and non-moving walls for the level it is in	
movingWalls		Each maze object contains the moving walls for the level it is in	A child of maze
nonMovingWalls		Each maze object contains the non-moving waslls for the level it is in	A child of maze
verticalWall	Box collider	Prevents the player from walking through it	
	transform		

	MoveVerticlWall (script)	Moves the walls back and forth	
sand	Material	Made in unity	
	Box collider	Used to detect collisions with player	
ice			same as sand
primaryJumpRamp			Taken from the Standard Assets Asserts
iceWall	Box collider		
	Material	Made in unity	
	iceWall (Script)		
doorMechanism(empty object)		Each doorMechanism object contains the doors, TriggerPlare and triggerCube objects for the level it is in	
triggerCube	Box collider	Used to prevent the player from walking through it	
	Material	Made in unity	
	Rigidbody	Used so that the cube can be moved using projectiles	
	OpenDoor (Script)	Moves the doors down when a collision between the triggerCube and triggerPlate is detected	
triggerPlate	Box collider	Used to detect collisions with the TriggerCube	
	Material	Made in unity	
triggerDoor	Box collider	Used to prevent the player from walking through it	
	Material	Made in unity	
player2			same as player1

ring	Mesh collider	Used to prevent the player from walking through it	Taken from the Standard Assets Asserts
GameOverTrigger (empty object)	Box collider	Used to detect collision with player.	
	End (script)	Used to display a game over massage and change to endMenu scene	

Evaluation

What was done

*Menu system

*Player controls

*Camera controls

*Points system

*Obstacles that change the speed of the player

*A second playable character, which the player can swap characters with while playing the game

*Scene management

*Prefabs

*Instantiation of prefabs at run time

*Resource management and optimization

Issues

*When shooting both payable objects instantiate a bullet. This behaviour is undesired and could cause trouble. Especially if the two objects are facing each other.

* Some times when shooting the control will change between the two cylinders when this is not supposed to happen.

What wasn't implemented

* The scoreboard, however this can be easily done by following the two tutorials below

- How to make a HIGH SCORE in Unity - available at - https://www.youtube.com/watch?v=vZU51tbgMXk

- SAVE & LOAD SYSTEM in Unity - available at - https://www.youtube.com/watch?v=XOjd_qU2Ido&t=259s

Personal statement

The work submitted is my own and no other people have worked on this project. Some of the assets used were taken from unity's asset store and have been stated in the implementation table. Coding tutorials used for the completion of this project are listed below in the references section. All other work that is to be considered a product of the author's (my) imagination and skill.

References

if gameobject is active - Available at:

https://answers.unity.com/questions/44137/if-gameobject-is-active.html

How to access a script inside of a gameobject from other script- available at https://forum.unity.com/threads/how-to-access-a-script-inside-of-a-gameobject-from-other-script.474559/

FIRST PERSON MOVEMENT in Unity - FPS Controller- available at https://www.youtube.com/watch?v=_QajrabyTJc&t=704s

TOP DOWN SHOOTING in Unity - available at https://www.youtube.com/watch?v=LNLVOjbrQj4&t=936s