Game Development Document

The game that I decided to create is brick breaker, where I followed the tutorial steps to code the game.

So, I created a new project with unity by setting the game as 2D and saving it in the correct path. Then I modified the background of the main camera by setting it to black instead of blue. Then I created a 2D sprite and from the inspector, I selected the sprite and clicked on UISprite. Then I set the size of the sprite to the appropriate size needed for the paddle. After that, I changed the colour of the sprite to yellow and moved it to the bottom of the screen and then set the name of the sprite to Paddle.

I made sure that the resolution of my screen was Standalone (1024 by 768). Then I created a new folder in the Assets and named it Scripts. After that, I created a C# script and named it Paddle. To make the paddle move horizontally, I opened the script in Visual Studio Code and modified the code to move horizontally. In the Update function, I added a variable to move horizontally. Then, I tested the game and tried moving it by pressing the left and right arrows.

From the speed variable I increased the speed of the paddle to 7. Then in the script, I added two variables (right and left), so as to prevent the paddle from moving out of the screen. In the script, I also wrote an if condition to calculate the size of the sides of the screen. Afterwards, going to the main screen, I set the variables leftScreenEdge and rightScreenEdge to -5.6 on the leftScreenEdge and a 5.6 on the rightScreenEdge. I had also

added a polygon collider 2D so that the ball can bounce in any direction.

For the ball, I added another sprite to the game and from the sprite option I chose the default knob shape. Then I named the sprite 'Ball' and also scaled the sprite to make it into a proper ball. I also added the rigidbody 2D. Then, I tweaked the elements of the sprite, I set the gravity of the ball and linear drag to 0. I also added a circle collider 2D and pulled the radius in to a 0.08.

After I had set the ball with colliders and everything, I created a C# script for the ball in the Script Folder and named it Ball Script. Then, I added a variable for the rigidbody 2D. After that, I connected the variable with the component (rigidbody 2D). I then gave the ball a force (speed) on the y-axis to initiate movement. I then linked the Ball Script with the ball sprite to show that it is working.

After creating the ball sprite, I created empty components for the sides of the screen and for the top and bottom edges. Hence, the ball wouldn't disappear out of the screen and positioned them accordingly. Also, I created a physics material 2D from the Assets folder and tweaked its properties by setting the friction to 0 and bounciness to 1. Then, I linked the physics material with the ball so that the ball can bounce.

With regards to the bottomScreenEdge component, I had tweaked the settings by setting it as isTriggered. Afterwards, I tagged it by creating a new tag and named it Bottom. Once that was done, I linked the Bottom tag with the bottomScreenEdge component. After that, I had added a new function OnTriggerEnter2D. This function will run every time the ball hits a collider and check to see if it hits the bottom by comparing

the tag. If the ball hits the bottom collider, it will display a message.

The next step is to indicate when the ball should or shouldn't be on the paddle. So, I created another variable in the Ball Script called 'inPlay'. If it returns true, the ball is moving around whilst if it returns false, the ball should be on the paddle and follow the paddle around. In the Update function we need to check in every frame if the ball is in play or not. I added another variable for the paddle to check whether the ball is on the paddle or not, then in the main game, I clicked on the Ball component and from the inspector, I found the paddle variable and linked the paddle gameobject with the variable.

So, if the ball is not in play, we are going to set the ball and the paddle in the same position. I had also created a child object for the paddle by adding an empty gameobject (Ball Position) so that the ball wouldn't be inside the paddle. So then, I clicked on the ball gameobject and found the paddle variable, which I then arranged to Ball Position. Then, in the OnTriggerEnter function, I added the inPlay variable as equal to false. I had also set the rigidbody's velocity to a Vector2.zero.

We need a button to launch the ball, so in the update function, we are going to check to see if the spacebar has been pressed. So, we use the Input.GetButtonDown("Jump") to indicate the spacebar that it needs to launch the ball and we set the inPlay variable as true. It is possible for the ball to go back and forth which could bore the user, so we rotate the left and right screen edges so that the ball wouldn't go back and forth that easily.

For the bricks, I created them with photoshop with a 64 by 32 pixels and then added them in the Sprites Folder, which I added

in the Assets Folder. Then I set the pixels per unit, which is shown in the inspector when clicking on the brick sprite, and change it to 64, then I hit Apply. Then I added a brick in the main game and renamed it to Brick as the gameobject. After that, I changed the background colour of the brick to a more prefered colour, from the sprite renderer.

Then I added a box collider 2D to the brick and added a new tag, which I then linked to the gameobject. The next step was making the brick as a prefab as we are going to use more than one brick of the same sprite. I added the Prefab folder in the Assets folder and changed the position of the brick to a -5 on the x-axis. Then I duplicated the same brick to add more bricks in the game. To duplicate the same brick, I went to Edit and used the Snap Settings, then to move the brick, I used the CTRL + the left click and moved the brick a unit to the right. I repeated the same procedure until I got 10 bricks. To add another row, I duplicated the whole row and I used the Snap Settings by using the CTRL + drag to 0.5 units down. Then, selected all the duplicates of the brick and made them the children of the original brick.

In the Ball Script, I added a new function OnCollisionEnter2D. So this is used whenever the ball hits the brick whenever the brick doesn't have a trigger function. So the first thing that we need to check if the ball actually hit the brick or no. If the ball hit the brick then the brick needs to be destroyed. To give it a cool effect, I added a particle system to brick to make it look like a part of the brick is being thrown away. I also tweaked the particle system by clicking through Emission and the particles over time, I changed to a 0. Also, I added a burst, then I went to the Shape and changed it to a Sphere, the radius I changed it to a 0.1, in Duration, I changed it to 2, in Start LifeTime I turned it

down to a 1. I also added a randomness between 2 constants 0.5 and 1.5, I turned the speed down with a random between 2 constants some with 0.5 and 1. Then I clicked on the Renderer and I changed from DefaultParticle to DefaultSprite shape. Then from the start size, I changed it to random between 2 constants 0.1 and 0.3. Then from the gravity modifier and changed it to 0.2.

I changed the Particle's colours by using the randomize between 2 colours having a light colour and the other with a darker colour. I also used the Rotation by Speed and changed it to a random between 2 constants to a -90 and 90 and renamed the particle system to Explosion. Then I added the Explosion as a Prefab and turned off the looping.

Then, in the Ball Script, I added a variable which can hold a reference to the Explosion Prefab. Before destroying the brick, I added a new line of code, I put the explosion in the correct place and also giving it a rotation. Then from the main game, I clicked on the Ball gameobject and found the Explosion variable and linked the Explosion particle system. When playing, I realized that the Explosion cloning is happening. So I went to the scriptand added a new variable for the position and rotation for the particle system and destroying the clone of the particle system throughout the gameplay.

For the UI Elements, I created the Canvas in the main game and in the canvas, I added a Text Box and moved it to the left side top corner. Then for the score, changed the colour to cyan colour, font size to 35 and did a vertical center to the text in the box, then I named the gameobject Score Text and for the anchor points, I clicked on the rectangle transform and picked the left top. Then I duplicated the Score Text to make the Lives

Text and moved it to the top right corner of the canvas. Then for the lives, changed the alignment to right align and for the anchor points, I clicked on the rectangle transform and picked the right top.

Then I created an empty gameobject, called it Game Manger and created a new C# script and named it GameManager. Then I linked the GameManager script with the gameobject and ready to run. So in the script, I added 2 variables for the lives and the score. Then I added another using to read the UI elements and also added 2 variables for the Text. Then, in the main game and Game Manager gameobject, I added 3 for the lives and then linked the texts UI with the texts variables.

In the Start function, I initialised the Text Boxes. In the Ball Script, I added another variable and named it gm (GameManager). So, when I go into the main game, the gm variable shows up, linking the GameManager object. I then added a new function, in the Game Manager Script, UpdateLives and added an int parameter (changeInLives). So in this function, I had set the lives to set an increasing value or a decreasing value, according to the situation of the game and also updating the text for the lives.

Then in the Ball Script, in the OnTriggerEnter2D function, we are going to the game manager (gm variable).UpdateLives(-1). This means its losing a live. Then in the Game Manager Script, I added a function called UpdateScore, adding an int parameter for the points. I had set the score to set an increasing value or a decreasing value, according to the situation of the game and also updating the text for the score. Then, I created another script for the bricks and attached it to the brick prefab. Then, I modified the Brick Script and added a variable for the points. In

the main game, we can see that in the brick prefab we can find the points variable and I changed the number to 10.

Then in the ball script, before we destroy the brick itself, we add the gm.UpdateScore(other.gameObject.GetComponent<Brick>().points); This is used to access how many points does this brick have. In the GameManager script and in the UpdateLives function, we add another line of code. If lives is less than or equal to 0 than we are out of lives and we set lives to 0. Then we add a new function for Game Over, which we want to freeze everything, the ability for the player to control the paddle and everything and also pop up a panel where the user can play again or quit the game.

In the beginning of the game, I added a variable for the gameOver. Then, in the Ball Script in the Update function, I added an if function for when the game over variable is true, it will stop the game. So then in the GameOver function found in Game Manager script, I had set the gameOver variable to true. In the Paddle Script, I had set another variable for the game manager and in the Update function, if the game over variable is set to true then it will stop the game by giving it a return. Then in the main game, I linked the game manager object with the game manager variable found in the paddle gameobject. In the Canvas, I added a panel for the game over section.

For the Panel, I modified the anchor points to set it in the middle of the screen.I renamed the panel to Game Over Panel and added a Text, which I set the name to Game Over Text. I modified the text of the Panel to Game Over, centered the text both vertically and horizontally, set the font size to 90, and change the text colour to red. Then in the Panel, I added a

button for the Play Again, modified the text to Play Again, changed the font to 30, changed the colour of the button, and changed the colour of the text. I duplicated the button for the quit button, then all I changed was the text of the button and positioning.

I made the panel invisible so that in the beginning of the game it wouldn't show up. In the GameManager Script, when the game over is set to true, we also want the panel to show up. So that the panel can show up, we need a new variable for the panel. Then in the main game, in the game manager object, we need to link the panel gameobject with the variable by dragging it. Going back to the script, we set the panel to active.

Back to the script, we have to add another public function naming it PlayAgain. Hence, we need it to reload our game and for it to work, we need to add another using for the scene management to reload our scene. In the main game, I went to the File section and chose the build settings. From there I clicked on the Add Open Scenes button to get the current scene. In the main game, I went to onClickEvent to add an event and dragged the Game Manager object to the Event to choose which function I want, then from the Game Manager script I chose the PlayAgain function. Hence, when I click the Play Again button, the scene will reload itself.

The same idea goes for the Quit function. The procedure is as follows, adding the function and adding the Application.Quit(); line of code. To test it out, I added a debug line of code stating Quit. In the main game, I went to onClickEvent to add an event and dragged the Game Manager object to the Event to choose which function I want, then from the Game Manager script I

chose the Quit function. Hence, when I click the Quit button, the Quit statement will show up.

Back to the script, I added a new variable named numberOfBricks to check how many bricks I have in the game. In the Start function, we added the new variable declaring it to find objects with the same tag named brick. We then added a new function, naming it, UpdateNumberOfBricks. So that as soon as the ball hits the brick, the number of bricks will decrease and also check how many bricks are left in the scene. In the ball script, when we hit a brick we have to update the number of bricks.