# 4 Implementation Chapter

## isGrounded() Function

This function returns a Boolean variable, it uses the variables “groundCheck.position”, which is the position of the empty game object that is placed just under the player’s collider, “radOfCircle”, which is the radius of the circle, and the layer mask called “groundMask”. The ground mask is applied to all objects that are in the layer “Ground”. The Boolean returns true if the ground mask overlaps with the circle under the player’s collider and false if it doesn’t.

 

Text

Description automatically generated

The author also draws a Gizmo which can be seen in the editor but not in game. This makes it easier to see where the ground is intersecting with the player.

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

## Flip() Function

This function will change the direction of the object it is used on the author changes the scale of the object’s X value by multiplying it by -1 which just flips the direction the object is facing.

Text

Description automatically generated

This is how the flip function is used In the “Update()” function. This checks if the object is not facing right, and the direction value is greater than 0 then flip the object to the other direction and vice versa.

Text

Description automatically generated

## Player Movement

The author uses Unity’s new Input System for keyboard and mouse the player will use the “A” and “D” key to move direction, and the “Space” key to Jump.

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

In the Events tab the author has their “PlayerScript” script that is attached to the player. They use the “Move()”, “Jump()”, “Dash(), and “Fire()” functions. For example, when the space bar is pressed the “Jump()” function checks if the action was performed and if the player has a jump available, it will make the player jump. The Author will go into more detail in the individual Functions later in this section.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

The Author created the “Move()” function which gets the direction the player is trying to move the direction is a value of 1, -1 or 0. If the player is not moving the value is 0. If the player is moving the value is 1 or -1 depending on the direction. This function also starts the run animation which happens when speed is not 0 however the jump animation take priority over the run animation. For example, if the player jumps and is moving to the side it will not play the run animation because the jump animation is being played.

The Author created the “Fire()” function which first checks

1. if the player can attack and if they can, it will use the “Attack()” function which activates the “attack1” trigger, this plays the attack animation, it is called attack1 because the author wants there to be two swings of the attack if the player presses attack in quick succession, this might not be implemented because of time constraints.
2. Next the “Attack()” function will put all enemies that are in the attack range when the button is pressed into an array called “hitEnemies”
3. it will run the array through a foreach loop that will deal damage to all the enemies hit.
4. “Fire()” will then put the attack on a “Cooldown” where the player cannot attack again for about a second.

The Author created the “Jump()” function which will first check

1. if the player can wall jump, which is something the player can only do when they are touching a wall. if true this will start a “Coroutine” called “WallBounce()” which will play the jump animation and set the players gravity scale to 0 so the wall jump isn’t affected by outside forces. It then pushes the player away from the wall and up into the air as if they jumped off the wall. The players gravity scale is then reset back to its original value.
2. Next if the player is not “isWallSliding” (the variable used to check if the player is sliding on the wall) and their max number of jumps is greater than the current number of jumps used. The variable “jumpForce” is applied to the player which makes the player jump. This is also where the Coyote Timer is used which will be discussed later under “Coyote Time Jump”.
3. Next “Jump()” will check if the “Jump” key was let go of or cancelled while the player was moving up (y velocity greater than 0) when this happens the y velocity is multiplied by 0.5 to make the player not instantly stop going up but look like they are slowing down, this allows the player to jump at different heights depending on how long they hold the “Jump” key for.

The Author created the “Dash()” function which first checks

1. If the player can dash with the Boolean “canDash”
2. If true it will start the “Coroutine” “Dash()”
3. This “Coroutine” sets the “dashing” animation trigger, makes the players gravity scale 0, adds a velocity to the player and starts emitting a trail renderer. It will wait for “dashingTime” seconds
4. Once done waiting it will turn off the trail renderer, reset the players gravity scale and reset the “dashing” animation trigger. It will then wait for (dashingCooldown) seconds before setting “canDash” back to true.

## Fire() Function

The “Fire()” function is using the new input system in Unity, when the specified input is pressed (J on keyboard, Right Trigger on gamepad)

Text

Description automatically generated

In the Update function the rigid body (the player) is given a velocity which is a Vector 2 this value is the direction multiplied by the “speed” variable which is 8 by default. Because this is only changing which direction the player moves the y value is not changed.Text

Description automatically generated

## Player Jump

As stated in the player movement section this is using Unity’s new Input System. The Jump function takes in an Input from the controller or keyboard. When the jump button is pressed and the player has more than 0 jumps left (a variable stores how many jumps the player has left) a vertical velocity (“jumpForce”) is applied to the player and moves it up. The “jump” animation is played here. Once the player has jumped the “jumpsLeft” variable is now -1 of what it was.

If the player cancels the jump (lets go of the “jump” button) but their vertical velocity is greater than 0 i.e. they are still moving up. The velocity will be multiplied by .5 which lets the player choose how high they can jump. The player can jump the highest by holding the “jump” button or they can choose to jump lower by letting go early.

Text

Description automatically generated

## Coyote Time Jump

Coyote time is used to add a short window after the player leaves the ground to still be able to jump. This takes account for human error giving the player a .2 second window to jump after they leave the ground or a platform.

Two variables are needed. One to store how long the error window will be and one that can be changed.

Text

Description automatically generated

In the “Update()” function when the player is on the ground the coyoteTime is assigned to the variable “coyoteTimeCounter”. Once the player leaves the ground the “coyoteTimeCounter” will count down, when the timer gets to 0 or less the player will no longer be able to use their first jump.

Text

Description automatically generated

If the coyote timer is less than or equal to 0 then it will take away 1 jump from the “jumpsLeft” variable. This makes it so the player cant get extra jumps from the coyote timer.

Text

Description automatically generated

To make sure the player can’t jump indefinitely by pressing the jump button fast the timer has to be set to 0 once the jump button is released.

Text

Description automatically generated

## Double Jump

In “PlayerMvt.cs” the “Start()” function sets the number of jumps left equal to the “maxJumps” Variable

Text

Description automatically generated

In the “Update()” function ever time the player is on the ground the “jumpsLeft” variable will reset to the max number of jumps.

Text

Description automatically generated

In the “Jump()” function as long as there is more than 0 “jumpsLeft” the player will be able to jump. So, if “maxJumps” is at 2 the player will be able to double jump

Text

Description automatically generated

Once the jump is pressed the “jumpsLeft” will be updated with the new number of jumps left which is 1 less.

Text

Description automatically generated

In the “Update()” function “maxJumps” is set to “jumpsLeft” when the player is on the ground. This means the players “jumpsLeft” will reset to whatever value of “maxJumps” when they are on the ground.

Text

Description automatically generated

## Wall Jump

## Player Dash

## Tile map

Tile maps are used to create 2D worlds they allow the creator to select the tile they would like to place and draw the tiles into the scene.



## Sprite sheet

The player’s sprite sheet has been downloaded from the Unity assets store, this came with different animations that weren’t set up properly, I fixed this by just re selecting the animation frames that went together.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

## 

## Animation

In the “PlayerMvt.cs” “Update()” function when the player is on the ground the falling Boolean is set to false this will stop the fall animation from continually playing after the player falls from anywhere.

Text

Description automatically generated

Once the player’s velocity is less than 0 the fall animation is played and the jump trigger is reset to make it so the player can jump again. The nested if statement checking if “IsGrounded()” is true to stop the falling animation if the player is still holding one of the movement buttons

In “PlayerMvt.cs” in the “Jump()” function when the player pressed the “Space” button the trigger variable jump is toggled and the Boolean variable “falling” is set to false this will play the “Jump” animation. When the “Space” key is let go the “Jump” trigger is reset and the “falling” variable is set to true.

Text

Description automatically generated

The “Idle”, “Jump”, “JumptoFall” and “Run” animations all work.

## A picture containing graphical user interface Description automatically generated

When the user doesn’t press any buttons and isn’t falling the idle animation will play, if the user presses the “A” or “D” key the run animation will play. if the user presses the “jump” key

From “Idle” to “Jump” a trigger condition that checks when the parameter “jump” is toggled is needed. Once the user reaches the peak of their jump the variable “falling” is set to true

Graphical user interface, application

Description automatically generated

When the user stops pressing the “Jump” key or when the user reaches the peak of their jump the “JumptoFall” animation will play when the falling variable is set to true

Graphical user interface, application

Description automatically generated

Once the Character lands on the ground “falling” is set to false and will no longer play the “JumptoFall” animation.

Graphical user interface

Description automatically generated

If the user falls off a platform the “JumptoFall” animation will play

A screenshot of a computer

Description automatically generated with low confidence

If the user presses the “A” or “D” key to move the “Run” animation will play. Checking if speed is greater than 0.0001 is making the response a bit quicker so the character will react faster to the input of the user.

Graphical user interface, application

Description automatically generated

If the user stops pressing the “A” or “D” keys the “Run” animation will stop playing. Checking if speed is less than 0.0001 is making the player character stop quicker.

A picture containing accessory, case

Description automatically generated

This is Updated so the JumpToFall will go back to idle at the correct time, and the player can go from “Run” to “JumpToFall” this makes it so the player isn’t running in mid-air or falling while moving on the ground.

A screenshot of a computer

Description automatically generated with low confidence