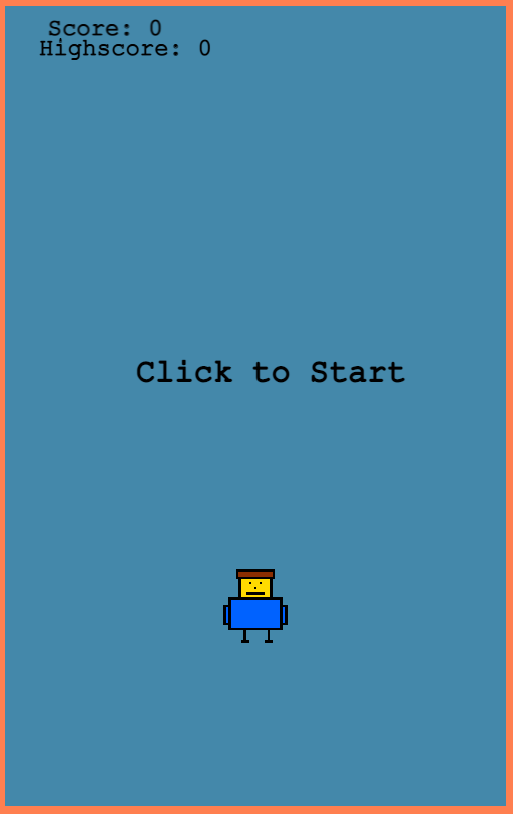
The game I’ve created is an endless jumper game, I chose this style of game as it wont require constant updates for it still to be enjoyable for example a Mario style platformer game would require new levels being added for it to still produce a challenge however as the game I have created is theoretically endless this is not the case.

A game I enjoyed playing growing up was doodle jump where the idea is to jump from platform to platform avoiding enemy’s and not falling. This was such an addictive game as you always knew it was possible to gain a higher score. Another game with a similar principle is temple run where the aim is to keep running for the longest amount of time possible while the game slowly speeds up. This was the approach I took with my game with the speed of the enemy’s increasing the further you travel.

Before the game starts, in the onCreate method I set up the camera. To do this it is set to follow the player, a deadzone is added so it only follows the player up and down and not left and right. An offset is the added so the player is towards the bottom of the camera.

Inside onCreate all the text is added and set to invisible if it isn’t meant to be shown until after the player has died. The player, enemy and platforms are all added to the scene and all have colliders added. The platform group is also set to immovable so they don’t start falling when then player lands on one. The animations and sounds are also created to be used later.

The game begins with a start screen showing the top score achieved and the player sprite.

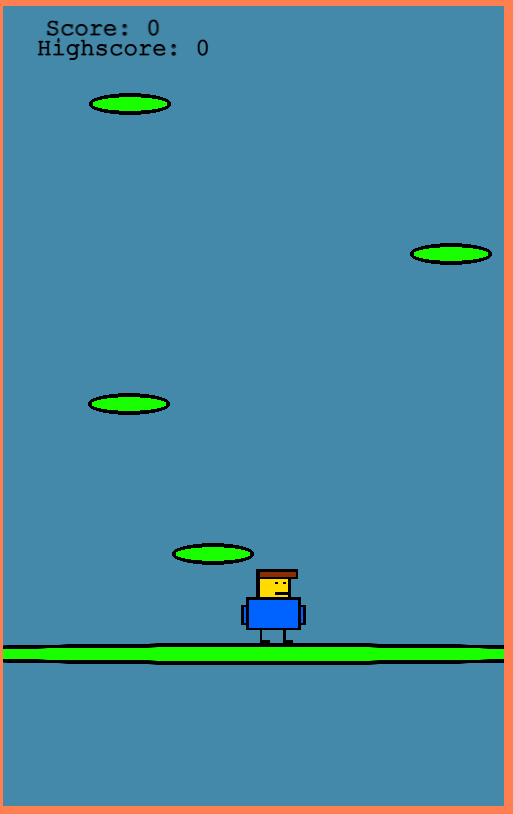
Once the screen has been clicked to begin the game the startGame function is called this removes the start screen listener and text, creates the first platform underneath player for them to start on, calls the createPlayer function and the createPlatforms function.

The create player function sets the gravity of the player to 900, I found this was a good value that allowed the player to jump high enough and give them a bit of time to drift to a platform but was also strong enough to pull the player down quick enough to allow fast gameplay.

I turned off collisions for the player on all sides other than the bottom to allow them to travel through platforms and land on top of them.

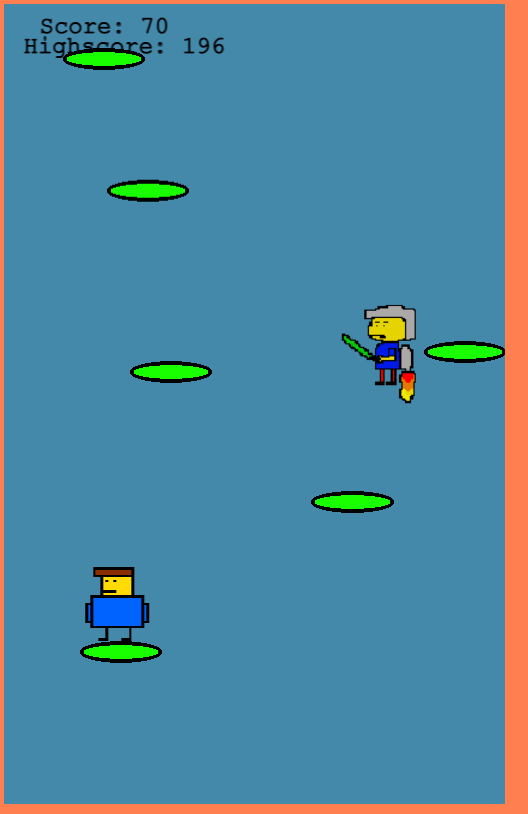
The create platforms function then uses a for loop to create the 7 platforms needed for the game. The x position is a random value from left to right of the game minus a few pixels on either side so half the platform doesn’t go off the edge. The y position is the platform number x 150 so each platform is 150 pixels apart

Once the game has started the isPlaying bool is set to true and everything inside the update function starts.

The first thing it calls is the updatePlayer function. This takes input from the player and allows them to move around, it checks if the player is touching down and if so will allow them to jump, this is done by setting the Y velocity to the opposite direction to the gravity to allow the player to move up the screen and will play the jump noise effect. It also checks for left and right keys and sets the x velocity accordingly. It also checks for a down arrow and sets the velocity in the same direction of the gravity. I added this so more skilled players will be able to move up the game faster by pressing it when they are above a platform rather than waiting for the gravity to bring them back down so they can jump again and so that if you are about to hit an enemy you can push it and it will give you a chance to avoid them.

This function also checks if the player is at the edges of the screen and if so it will wrap them to other side so they don’t get lost of screen or to jump to a platform on the opposite edge.

The next function is the addPlatforms. This works by iterating over the group of platforms, working out if a platforms y position is more than the distance the camera has scrolled + 30 pixels and if so it will change its y position to be distance the camera has scrolled – a random number between 50 and 100, so it adds variation to the positions the platforms and its x position to be random from the left to right of the screen. I used this method of moving platforms up the screen rather than just putting it more for performance as pooling means only the first 7 platforms are the only ones created, rather than creating a new one every few seconds.

Next the score is updated this is done by getting the distance the camera has scrolled and dividing it by 10 so it’s not too large and then multiplying by – 1 to give a positive score.

Next the bottom platform variable is updated to be the platform furthest down the screen, this is then used to calculate if the player has fallen by seeing if its y position is below that of the bottom platform.

Next the enemy is moved further up the game, once the player has passed it and it is off the screen. I could of instantiated an enemy from a class every time I wanted to spawn one in but I chose to just use one and move them up the screen, once the players passed them, because there will only ever be one on the screen at a time so creating and destroying over and over would be pointless. Also in this if statement is a calculation to increase the speed of the enemy, so every time the player encounters one it will be faster than the last one.

A check is then done to see if the player is close to the enemy and if they are a sound will play to let them know it is ahead. The enemy is also moved in the update function by calculating how far they are to the edge and changing their velocity to the opposite direction so they move from left to right.

A check collision function is then called to determined if the player and enemy are touching by working out both of their x and y positions and checking if the difference is less than 100 and if it is the gameOver function will be called.

The game over function sets the isPlaying bool to false, plays the game over sound, turns off all collisions and camera follow for the player so they fall off the screen, for effect. it then compares the players score to the high score, if it is larger the score is stored as the new high score and a message displays letting the player know they’ve beaten the high score. The game over text is shown with a click listener to restart the game.

This calls the restartGame function which resets the score and speed of the enemy and calls this.scene.restart().

To improve the game in the future I would change the enemy’s sprite when they move so it looks like they’re a different person each time.

I would also have a powerups class that give the player boosts such as a trampoline to shoot you higher up the game.