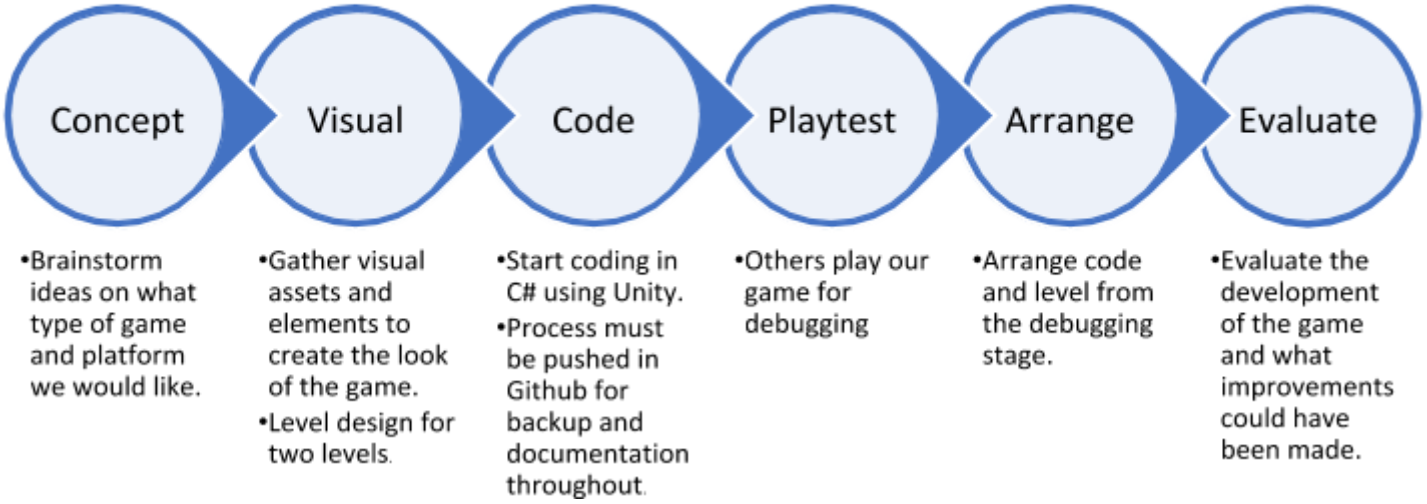
Phase 2; Analysis

Rakel Borg & Matthew Agius

For this phase we intend to do a platformer game. Our game concept doesn't really have a defined story after it however, we stuck to a dark, futuristic, post-apocalyptic theme. This was important as to choose visual assets regarding this concept.

Whilst designing and creating our game we followed this procedure below;



This model is useful as we followed through it in our past units regarding gaming. This type of production model is also seen in the book "Game Design Workshop" by Tracy Fullerton however, they focus more on playtesting not just once. Debugging is important to cover all the possible aspects in the game to see if the player continues playing or would be stuck in the game because of the game itself.

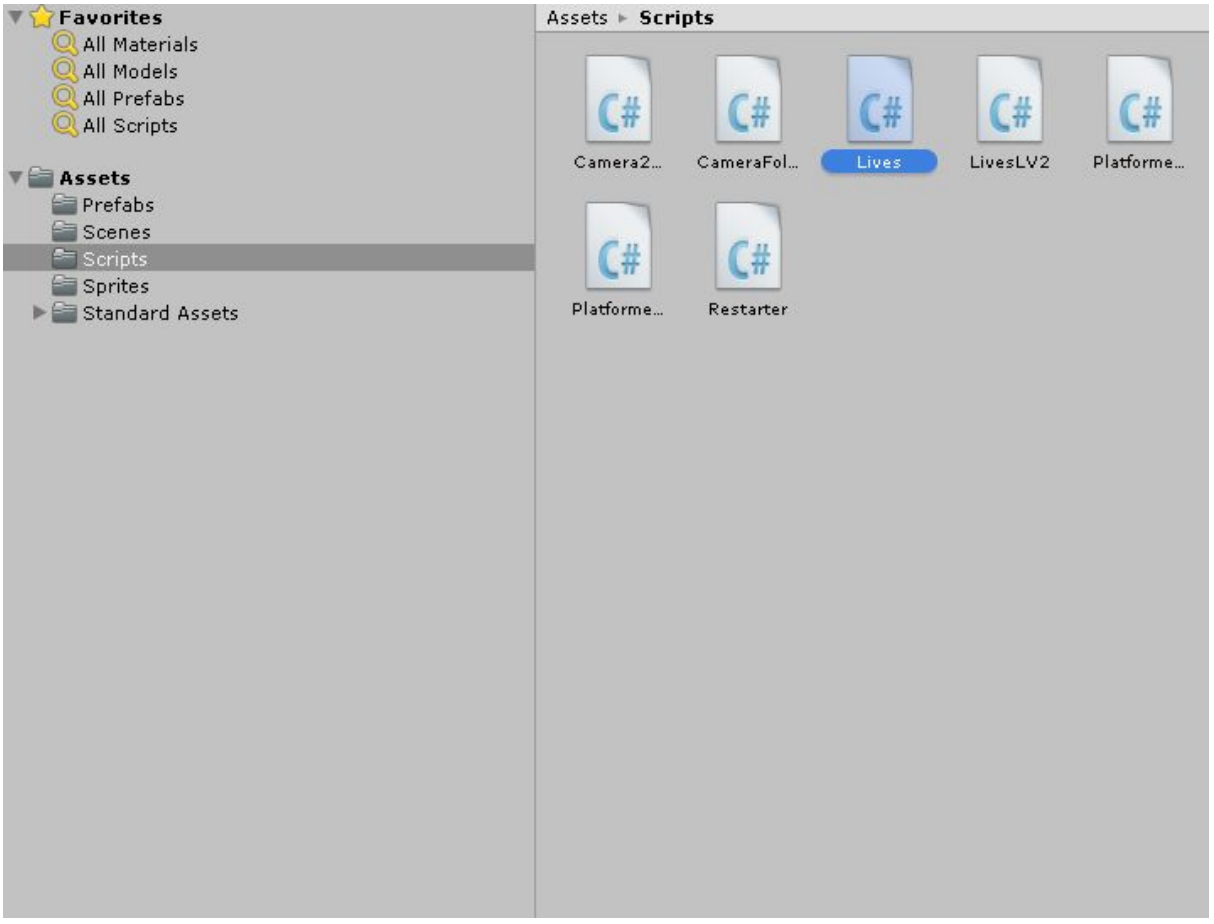
In order to do this one must play to break the rules and in an unconventional method. Exploring of the game is an asset whilst testing out all different kinds of commands in order to see the result. A well-known game with such bugs and glitches is Assasin's Creed Unity where a lot of characters which roam around the player are either stuck in walls, spawned at wrong moments or are hanging in mid-air. This is important in order to make the player more immersed in the game and not ruin the level of engagement.

Visuals and level design also indulge the player into the game. If the game catches the eye of the player, he would then try to play it to see if it interest's him. If the level design is gradually increasing in difficulty, then it would engage the level of flow in the player as one may take it as a challenge. Sound also helps to enhance once senses to get involved more in it.

In conjunction to this, performance of the player is an important factor when it comes to see the relation between presence and task performance. If a player has a good immersive tendency and motivation, then that individual is more likely to be engaged in what one is doing/playing.

For our project, our GitHub folder will contain other subfolders. Similar to our phase 1 task these would include animations, images, prefabs, scenes and scripts. We will probably have a GameManager which as the name itself suggest manages the game. This would include code about

initialisation of the game and levels and how they connect with each other. Scripts with the name of the levels would follow as these would have code regarding to the respective level.



When doing the game this is found in our Restarter script. Camera contained all necessary code for the GUI to move with the character. Platformer contained the moves of the character. Below are some screenshots with the most important code for out game mechanics.

```
if (Input.GetKeyDown(KeyCode.Escape))
{

    Debug.Log(pauseToggle);
    pauseToggle = !pauseToggle;
    Enviorment.SetActive(false);
    rend.enabled = false;
    Pausegame.SetActive(true);
    Cursor.visible = true;
    Time.timeScale = 0;


}

if (pauseToggle == true)

{

    Time.timeScale = 1;
    rend.enabled = true;
    Enviorment.SetActive(true);
    Pausegame.SetActive(false);
    Cursor.visible = false;
}
```

This code controls the escape key in relation to the GUI.

```
if (life == 3)
{
    Debug.Log("life is now 3");
    Destroy(GameObject.Find("Life4"));

}


if (life == 2)
{
    Debug.Log("life is now 2");
    Destroy(GameObject.Find("Life3"));

}


if (life == 1)
{
    Debug.Log("life is now 1");
    Destroy(GameObject.Find("Life2"));

}

if (life == 0)
{
    Destroy(GameObject.Find("Life1"));
```

This code controls how life mechanics work resulting that if the player has 1-3 life it would be displayed if life is 0 it would result in ending the game, enabling the character to do any more moves.

```
if (other.tag == "end")
{
    SceneManager.LoadScene("MenuScene");

}
```

Code for ending the game and showing the menu scene.

```
    if (other.tag == "bounderies")
{

    life--;
    Debug.Log(life);
    transform.position = new Vector3(-6, 2, 0);
    StartCoroutine(Flash(flashSpeed));
}
if (other.tag == "Hazard")
if (other.tag == "Boost")
{
    Debug.Log("boost");
    rb.AddForce(Vector3.up * 930f);
}
```

Code for when collided with objects, decrease life.

Code for booster where force is added.

```
{
  if (other.tag == "trap")
  {

      trap.SetActive(true);
      Debug.Log("playerEnter");

  }

  if (other.tag == "Trap2")
  {

      trap2.SetActive(true);
      Debug.Log("playerEnter");

  }

  if (other.tag == "Trap3")
```

Activating the ball spike traps when the player enters the final part for level 2.

When hitting something that decreases life, character starts flashing indicating that a life is lost and that it had hurt the robot.

This enables the robot to start flashing.

```
IEnumerator Flash(float v)
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "particleactivator")
    {
        Debug.Log("particlesON");
        particles3.SetActive(true);

    }




    }
void OnTriggerExit2D(Collider2D other)
{
    if (other.tag == "particleactivator")
    {
        Debug.Log("particlesOff");
        particles3.SetActive(false);

    }
}
```

Enables particles on trigger when the robot steps on the platforms in level 2.

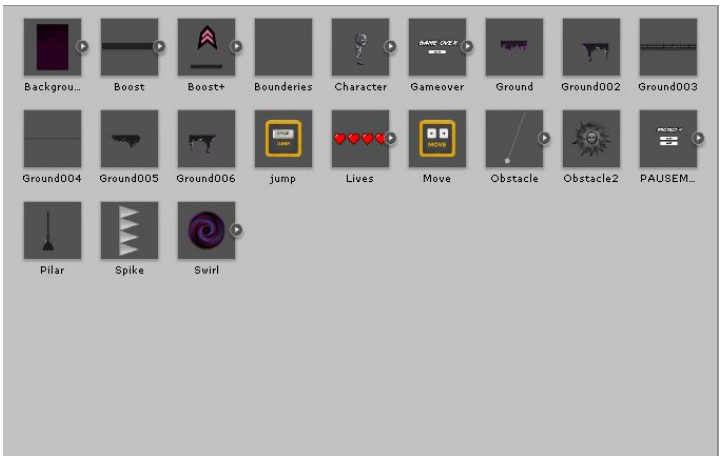
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;

public class menuscript : MonoBehaviour {

    public void Changescene(string sceneName)
    {
        Application.LoadLevel(sceneName);
    }


}
```

Loading of new scenes.



All the prefabs in the prefabs folder.

All these were implemented in order to have a functioning game with functioning mechanics.