Kimberley Collins

MSD 6.1B

Abstract

Game Documentation

Assignment 2 – Runner Game

Object Oriented Programming

Contents

[Section 1 – Game Design 2](#_Toc93762426)

[Section 2 – Functionality Requirements 3](#_Toc93762427)

[Section 3 – Code Requirements 3](#_Toc93762428)

[Section 4 – The Final Game 3](#_Toc93762430)

## Section 1 – Game Design

1. Prepare a Game Design Document, which outlines the following: (see the downloaded documents in the downloads folder)
   1. Game Title and Description
   * Game title for the game is Runner. The game concept of this game is to create a player that needs to collect coins as much as possible in a few seconds and while facing the obstacles without touching them so the player lives and health won’t decrease and the player will end up losing. Throughout the game, the player is going to face obstacles, the player needs to run away from the obstacles which they will consists of different strengths. If the player hit a coin, the score will increase. At the end of the game, the game will show the player how much score and high score does the player have. To be able to move the main character who is a man, the user need to click on the four-arrow key of the keyboard which are the top, bottom, left and right keys and the user need to click on the space bar to make the man jumper higher to collect as many coins as possible in a few seconds because a timer will be applied to the game.
   1. Choose and describe a Game Type (Single Screen, Platformer, Scroller, Side-Scroller, Adventure)
   * The genre of this game is going to be a running and jumping game which is going to be developed for computer devices. It is going to be a 2D platformer game which is intended for kids who are six years old and more.
   1. Game Loop
      1. What will the player be doing in your game?
      2. How will they do it?
      3. How does the player progress through the game?
      4. How is the narrative delivered?
   2. Scope
      1. How long is the game?
      2. How many levels are there?
      3. What is the average play time?
      4. What are the objectives?
      5. How many playable characters?

* The game is about 25 seconds long which consists of a welcome scene, two levels, which the second level is a little bit difficult than the first one. The
  1. Art Style / Assets
     1. Include a description of your art style and supplement with art concepts or inspirational concepts. Describe your style for the environment, characters, UI, etc. You can also link to a different area/scene it lives in.

## 

## Section 2 – Functionality Requirements

Design one Animation using state machine system.

Diagram

Description automatically generated

Graphical user interface

Description automatically generated

## Section 3 – Code Requirements

## 3.4 Research and describe Exception Handling

In Object-Oriented Programming, exception handling is a dominant mechanism for centralized processing of issues and unusual situations which let the exceptions to be thrown and caught. This type of mechanism is controlled by the Common Language Runtime (CLR) which changes the procedure-oriented method or error handling in which each method or function returns a code that suggests a successful execution or an issue. Exception handling will answer to the exceptions when the program starts running. The exception will occur when the unpredictable event appears that it needs certain processing. For instance, a developer will produce an abnormal input, when the method is trying to attempt an answer to divide by zero or else a file system error is experienced throughout the write or read a file.

<https://www.tutorialspoint.com/csharp/csharp_exception_handling.htm>

<https://www.computerhope.com/jargon/e/exception-handling.htm>

## Section 4 – The Final Game

4.1 Design a class diagram to describe your OOP application.

4.3 Write a short report detailing:

a. The code design pattern and OOP concepts used in your game.

Code Design Pattern

I used code design patterns because these are solutions to minimize the common problems and extra coding in many scripts, which makes the project more reusable, clean, maintainable and easy to use. Two different types of patterns that I used were structural and real-world. Structural code when using types of names as explained in the UML class diagrams. While real-world provides situations where theses patterns are going to be used. Code design patterns that I used were GameData script, which is an abstract class, GameManager script was used to take care of the game states and handle the game data, singletons and inheritance.

Game Data script which is an abstract class was used so the other scripts can be derived from this class, so the object, especially the player when is moving, the player will remain between the camera main points and so it will not exceed them. Even when the player is in Level 2 and it is moving, the enemies which are moving along the player will stay between the camera main points. To avoid duplicates of code, the game data is going to save the score, the highscore and health of the player, which from this class, so later on these can be called from the GameManager script, so a method for each one of them can be created accordingly, which later on these methods can be called from another script like the Coin script, so when the user hits the coin, the score will be added and in the enemy script, so when the user hits an enemy the health will be reduced.

Singletons were used to make sure that a class has only one instance which provides access to that instance to every script of the project else if there are more than one instance it will destroy the other instance, so the project can run smoothly without any issues, so the instance can be used according to the developer needs. This was used in the GameManager script in the private void Awake() method, to make sure that only one instance is being used, if not then destroy the other instances. Also, the single pattern was used to make the player visible from level 1 scene to the level 2 scene from the private void Awake() method, so only one player will be visible, therefore this reduce the amount of code which needs to written in a script and attach it to another player.

Inheritance was also used to avoid duplicates of code. It is used in the Enemy script which inherits from the ConstantEnemy script. To make the enemies moving with the player, the code is only written in the ConstantEnemy and then from the Enemy script make the methods public override and in these methods in the Start and Update method call the Start and Update method of the ConstantEnemy with the base.Start() or base.Update().

OOP Concepts

OOP concepts that I used were access modifiers, inheritance, polymorphism, abstract class and the getters and setters. Certain access modifiers were implemented to be public, private and protected. Public, so that I can call and amend certain attributes from other scripts, for instance public static float XMin() from the GameData script to the Player script, so the player will not exceed the camera x and y positions. Private so that, I can use them within the same class only, so I will make sure not to amend them from any other scripts and protected, I used it for the methods when a script is being inherited from another script. Inheritance was used in the Enemy script which inherits from the ConstantEnemy script. To make the enemies moving with the player, the code is only written in the ConstantEnemy and then from the Enemy script make the methods public override and in these methods in the Start and Update method call the Start and Update method of the ConstantEnemy with the base.Start() or base.Update(). Polymorphism was used in the GameManager script to change the method PlayerScore which maintains the score of the player to return a string. So, the score can be changed into text, so the user can see how many points he has. Getters and setters were used, so certain attributes can be overwritten or can be read only. For example, get { return \_PlayerLives; } set { \_PlayerLives = value; }. To store the current live of the player when he won’t hit the enemy and when the user hit the enemy, his lives will reduce, and it will be shown to the user.

Abstract class was used in the GameData script so the other scripts can be derived from this class, so the object, especially the player when is moving, the player will remain between the camera main points and so it will not exceed them.

Graphical user interface

Description automatically generated with low confidence

b. Suggestions for improvement.

To improve my game, I could make the player move his legs while he is moving on the grass, and while he is jumping to collect the coins, I could make him jumping position instead of running. Another improvement could be that while the background is scrolling and the player is moving and jumping, I could create a script which it will take care of the grass, so as soon the last grass is showing, I could keep looping and showing the grass automatically throughout the game and could add more enemies with different health. If the player touches an enemy, a health bar will be shown with and the health of the health bar will be decreased according to the enemy’s health. I could add more levels, by making them more difficult. I could add a menu with different players, so the user can choose which player the user wants to play with. Also, I can make the game 2 versus 2, so two users can play their main characters, while the other two users can play as the enemies. As soon as the user clicks on the application a background music will be played until the user exits the game.