Practical Gaming 2024

# Name of Student Stefan Lajtar

# T Number T00225936

# Name of Project Santa Christmas Runner

# Gameplay

My game is called “Santa Christmas Runner”, it is a platform game, endless running. Only one level exists. Map consists of four sections, “Start Section” is static and its always same. Other three sections are being spawned randomly after certain amount of time using an array. After some time, sections are being removed. Each section contains objects that are making up that section, houses, churches, trees, lanterns, fire logs and snow that makes up surrounding scenery. Game contains a coins which are already placed on the map (they are not spawned randomly). The amount of collected coins are shown in the top left corner. Santa has a step counter which represents the amount of steps made during a game, it is shown during a game in a top right corner. Santa is controlled by using WASD or arrow keys. Game starts by pressing play at the main menu. Then we are transferred to the game. It fades in with counting down (Ready, set, go). During that period, we are not able to control Santa. After counter is done player is getting control over the Santa. Aim is to avoid the obstacles which are represented in terms of rocks, snow piles, fire logs, fences and fallen trees. After we fall the game is finished and the number of coins collected, and steps done are displayed on the middle of the screen. After few seconds the screen fades to black and we are transferred to the main menu screen where we can choose to play again or to exit.

# Coding

Under each of the following headings, please describe the concept, why is it or isn’t it useful/needed, where do you implement in your project, you may provide screenshots or cut and past code segments etc..

* Frame Rate Independence

Class: Santa Movement

A screenshot of a computer code

Description automatically generated

I am using “Time.deltaTime” to ensure that game movements speed is consistent regardless of the frame rate. It means that object will move same distance over a second regardless of the frame rate eg 30fps 60fps. Without it on the 60fps object would be moving two times faster than on 30fps.s

* Interfaces

Code in the SantaMovement class

A screenshot of a computer program

Description automatically generated

IObserver, is a behavioral interface in the context of the Observer design pattern. In this design pattern, the interface defines a mechanism for subscriber objects (observers) to receive notifications when there is a change in the state of another object (subject).

PlayerScore class:

A screen shot of a computer program

Description automatically generated

In the provided code, OnTriggerEnter is used to detect when the player's collider intersects with a collider tagged as "Coin". When this happens, it logs a message and then calls NotifyObserver, which would in turn notify all observers that are registered with \_playerSubject. In this case, since PlayerScore is an observer, its OnNotify method would be called, updating the player's score display.

* Inheritance

Inheritance is when a new class is created based on an existing class. One of the features is that it allows us to reuse the code without having to rewrite it thus reducing redundancy and errors.

A screen shot of a computer program

Description automatically generated

The class SantaMovement is inheriting from Subject class. The Subject class is a generic class that has a type parameter <IObserver>, indicating that it's designed to work with objects that implement the IObserver interface. This inheritance relationship means that SantaMovement inherits all the members and methods of the Subject class, including any public or protected fields, properties, and methods. This allows SantaMovement to access and use the functionality provided by the Subject class.

* Case pattern

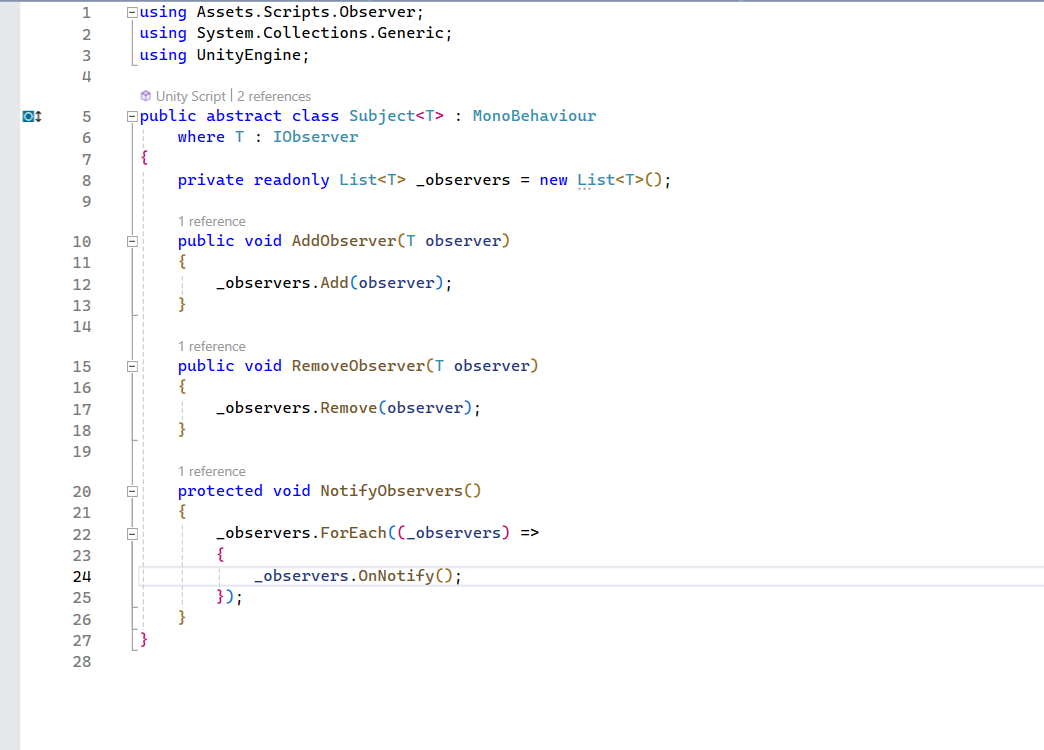
A screen shot of a computer program

Description automatically generated

DropdownManager class uses a switch-case pattern to handle different selections from a dropdown menu. This pattern is common for handling multiple distinct cases or options, each requiring different actions. DropdownManager is assigned to a dropdown menu at the MainMenu scene where user can choose difficulty. Switch case pattern is more efficient than a series of nested if-elses for these types of choices. It also makes it clear that the code is choosing among a number of discrete, known options.

* Observer Pattern

This class forms the central part of the Observer pattern's subject side, allowing observers to register for and be notified of events.

Class: Subject

Class: IObserver

The IObserver interface declares a method OnNotify that any implementing class must define. This method is what the Subject<T> will call when it needs to notify its observers of a change or an event.

A screenshot of a computer

Description automatically generated

Class: PlayerScore

The PlayerScore class, which implements the IObserver interface. The PlayerScore class, which implements the IObserver interface. This class has the OnNotify method that updates the player's score. It registers itself with the subject to receive updates.

The PlayerScore class holds a serialized reference to the Subject<IObserver>, allowing it to register and unregister as an observer during the OnEnable and OnDisable events, respectively.A screenshot of a computer program

Description automatically generated

Class: SantaMovement

OnTriggerEnter method detects a collision with an object tagged "Coin" and then calls NotifyObserver. The object with this collision detection code acts as a concrete subject, executing the logic that determines when the observers should be notified.

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

The good thing about the observer pattern is consistency, It ensures that all observers are made aware of changes in the subject, keeping them synchronized.

* Polymorphism

Polymorphism is a concept in object-oriented programming (OOP) that refers to the ability of different classes to be treated as instances of the same class through a common interface or base class. There are two methods: method overloading and method overriding. Polymorphism is useful because it allows flexibility which means that one piece of code can be used on objects of different classes as if they were objects of a base class. In this way we can write more reusable code, it also simplifies code by allowing single method to handle different types of objects. Unfortunately, I don’t have any example of polymorphism in my code.

* Communication between scripts/game objects

Class Santa Movement

A screenshot of a computer code

Description automatically generated

The script communicates with game object through Input. It listens for player input (through Input.GetKey) and translates that into movement by manipulating the game object's transform. The script checks the game object's position against predefined boundaries (MapBoundary.leftSide and MapBoundary.rightSide) to limit its movement within the game world.

* Instantiation and Prefabs

I have prefabs for fence, cliff, firewood etc. They all can be found in the sections, they represent obstacles that player has to avoid.

A screenshot of a video game

Description automatically generated

* Magic Numbers

Magic numbers are numeric values with unexplained meaning or context that just appear in code.

Class: Map Boundary

A screenshot of a computer program

Description automatically generated

Magic numbers might be problematic like in this case reader might not know why are -3.4f and 3.4f chosen to be boundary. Other thing that might be a problem is maintainability, If the game's design changes and the boundaries need to be updated, these numbers would have to be found and manually updated in the code, which is error-prone.

* Model Animation

Santa has animations that are already coming in his package. The basic one which we use for running is called “HumanoidRun”. Other two animations that are used are: “HumanoidJumpUp” which is used for jumping and “HumanoidFall” which should represent fall of the character.

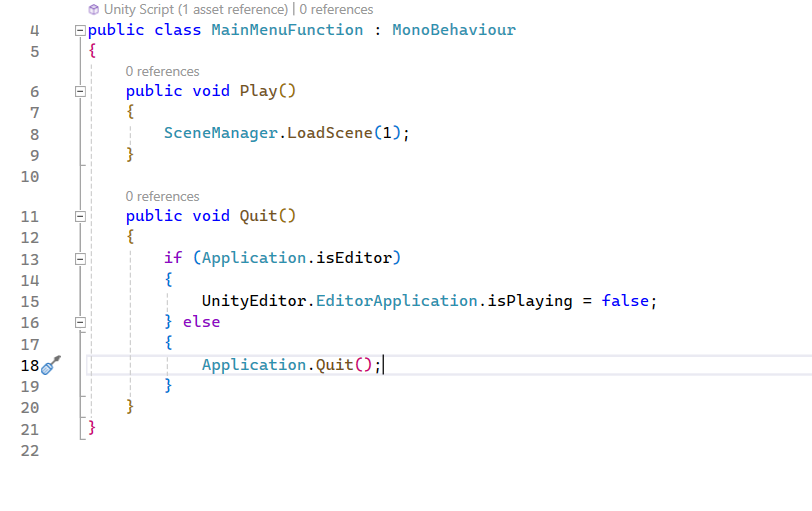
* Self made models and or animations

A screenshot of a computer

Description automatically generated

Other animations are made by me. FadeIn is an animation that starts of at the beginning, it lasts only one second so sometimes is not visible at all, but it serves as a “loading screen” in the game. Countdown and ready are part of animation responsible timer that is shown on the screen(Ready, Steady, Get Ready, Go). FadeOut serves as a transition after fall, screen turns black and player is transferred to the main menu. CameShake was supposed to be effect after Santa hits an object but I could not implemented it fully so its abandoned.

* Interactions between objects/scripts



This script is attached to main menu to handle UI interactions. First Method Play uses ScreenManager.LoadScene(1) to load a next scene (Main Scene) and its tied to a button PLAY.

The other method (Quit method) is connected to a QUIT button. If the game is running as a standalone application, it quits the application Application.Quit(). If the game is being run in the Unity Editor, it stops the play mode (UnityEditor.EditorApplication.isPlaying = false).

* Propper code placement

Class: Start Run

A screenshot of a computer program

Description automatically generated

The StartCount class contains a coroutine CountdownSequence that is started in the Start method. This coroutine appears to manage a countdown sequence by activating certain GameObjects (ready, steady, getSet, go) and playing audio effects (countdownFx, goFx) at one-second intervals before enabling movement (SantaMovement.canMove=true).

* Code repetition

I don’t have any code repetition to show, I was trying to avoid it as much as I could and I think I managed that successfully.

* Feature 1

One of the features would be timer/clock which contains Ready, Set, GetReady, Go. It lasts about 4 seconds, and they all contain sound. When timer is on, Santa movement is limited(non-existent) so player cannot move Santa until timer is finished. For the duration of timer “canMove is false” so Santa can not move, after its finished “CanMove” changes to true and now Santa can be controlled.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

* Feature 2

Other feature available is that Santa speed will gradually increase during the game. Every certain amount of the steps speed will increase for some amount.

A screenshot of a computer program

Description automatically generated

* Feature 3

One more feature that I would like to show is SectionDestroyer. As the game progresses more and more sections are spawned, more sections can lead to fps drop and game might start lagging. To try to avoid this, all sections that are behind will eventually after some time start disappearing.

A screenshot of a computer program

Description automatically generated