Game Documentation

Duck Hunt

Team Members:

Alejandro Perez
Leal Laya
Jesse Aronson
Annabel Herrera
Abdullah Asiri

Table of Contents

1. Introduction

Background

Technology has advanced in unprecedented ways, with the exponential evolution of computers containing high computational power as well as industries within the field such as software engineering growing immensely. These advances have fueled the gaming revolution into high quality HD gaming, resembling, and simulating real life while giving players the opportunity to experience the game through multiple points of view. These features allow players to fully immerse themselves into a fantasy world that feels like reality. The 3D duck hunt shooter game enables players to get a feel for what duck hunting is really like. The ambiance and terrain plunge the player right into the appropriate environment. The first-person point of view allows the player to experience the difficulty to aiming at a moving animal. Advances in technology, specifically this field, have allowed people to experience thrilling activities that otherwise might be difficult to do in real life.

Game Description

'Duck Hunt' is a 3D shooter game for personal computers where the player's main objective is to hunt and shoot moving ducks on the screen using a weapon from a first-person point of view. The players goal is to shoot several targets within the time allotted to unlock the next level. The levels difficulty increases as the player progresses through the game, the ducks speed increases as well.

Purpose

The game of 'Duck Hunt' is created mainly for entertainment purposes. The game allows players to engage in the simulation of a duck hunt without the need to pursue this activity in real life. It may serve as an educational tool for those interested in getting a feel for what this activity entails. The scenery of the game closely simulates that of a real-life duck hunting experience. The ducks move continuously while the player takes the role of hunter from a first-person point of view.

Product Scope

Project scope includes the following:

- Designing/Development of a personal computer game.
- Game should have a duck hunting like gameplay.
- Three levels for user to complete.
- Three different types of terrains.

- Three different types weapons.
- One type of enemy.
- One FPS character controller.
- Level timer.

Aims & Objectives

The goal of this product are as follows:

- Create a game that engages the user into a real-life duck hunting environment.

Target Demographic

The intended audience for our game is for any gender 13 years and up. Viewer discretion is advised due to the violent nature of the game.

2. Overall Description

2.1 Product Perspective

The duck hunt game is designed for seamless play on a PC at any designated time. No internet connection is needed to play because it is a stand-alone application.

2.2 Game Mechanics

2.2.1 Camera

2.2.1.1 There will be a main camera attached to player, giving the First-Person perspective.

2.2.2 Player

2.2.2.1 The player is controlled by the user and it is a single player experience.

2.2.3 Enemy Movement

2.2.3.1 The enemy is controlled by an enemy movement script.

2.2.3.2 The enemy moves diagonally, lowers and gains height.

2.2.4 Player Movement

2.2.4.1 To move, the user must use pointing device the player only moves the upper body, there is no walking, sprinting or jumping.

2.2.5 Weapon

2.2.5.1 The weapon is moved by the player movement.

2.2.5.2 The weapon shoots one bullet at a time and user must wait a second to be able to shoot again.

2.2.5.3 The weapon has a set number of bullets and the user will have to wait two seconds to reload.

2.2.6 Bullet Collision

2.2.6.1 The bullet is the same for all weapons.

2.2.6.2 The bullet has a large collider to make it easier for the user to hit moving targets.

2.2.7 Timer

2.2.7.1 The timer is set to 60 seconds.

2.2.8 Level Progression

2.2.8.1 The more the user progress in game levels become more difficult.

2.2.8.2 User cannot restart levels.

2.2.8.3 The enemies get faster and smaller on level progression.

2.3 Operating Environment

We used Unity Engine (version 2020.1.6f1 Personal), which is the world's leading real-time 2D and 3D development platform for games, animation, film, automotive, transportation, architecture, engineering, manufacturing & construction environment to work on our project.

The game can be played as a compiled application on a Personal Computer (Windows or MAC operating system).

2.3.2 Client Side Technology

- C-Sharp - C# is one of the most popular programming languages and can be used for a variety of things, including mobile applications, game development, and enterprise software.

2.5 User Manual Documentation

The game is very intuitive, easy and user friendly. **‘How to Play’** can be found under the **‘Options’** menu. Also, a text file is included inside the game files called User Manual, which explains thoroughly the game.

3. Functional Requirements

These are descriptions of services that the game must offer. It describes the game system and its components. Also, it demonstrates how the game will behave and should react to a particular system or situation. Thoroughly, it explains the functionality of the game, facilitating the development process.

3.1 Main Menu

3.1.1 The user shall be able to interact and distinguish main menu.

3.1.1.1 The system must display pointing device cursor.

3.1.1.2 The user shall be able to see a background image representing game.

3.1.1.3 The user shall be able to hover buttons using pointing device.

3.1.1.4 The user shall be able to click a button.

3.1.2 The user shall be able to play the game.

3.1.2.1 The system must redirect user to level 1.

3.1.3 The user shall be able to learn how to play the game.

3.1.3.1 The user shall be able to see detailed instructions on how to shoot weapon, aim at enemies, number of levels, time limit per level.

3.1.3.2 The user shall be able to click a button and go back to main menu.

3.1.4 The user must be able to see credits given to asset authors.

3.1.4.1 The user shall be able to see appropriate source credit of assets in text.

3.1.5 The user shall be able to quit the game.

3.1.5.1 The system must quit the game application.

3.2 Pausing The Game

3.2.1 The user shall be able to use keyboard button to pause game.

3.2.1.1 The user shall be able to distinguish when pause mode is on.

3.2.1.2 The user shall be able to see a pause menu.

3.2.1.3 The user shall be able to distinguish pause menu buttons.

3.2.1.4 The user shall be able to see level timer stopped.

3.2.1.5 The system must stop time scale of game.

3.2.1.6 The system shall be able to stop player movement.

3.2.1.7 The system shall stop enemy movement.

3.2.2 The user shall be able to use pointing device to interact with pause menu.

3.2.2.1 The system shall be able to make pointing device cursor available for user.

3.2.2.2 The user shall be able to move cursor using pointing device.

3.2.2.3 The user shall be able to click a button using pointing device.

3.2.3 The user shall be able to resume game when button is pressed.

3.2.3.1 The system must resume time scale of game.

3.2.3.2 The user shall be able to see level timer continue.

3.2.3.3 The user shall be able to see enemies start moving again.

3.2.3.4 The user shall be able to see environment live again.

3.2.3.5 The user shall be able to move player using pointing device.

3.2.4 The user shall be able to go back to main menu when button is pressed.

3.2.4.1 The system must redirect user to main menu.

3.2.4.2 The system must reset any graphical user interface components used.

3.2.4.3 The system must start game time scale.

3.2.5 The user shall be able to quit game when button is pressed.

3.2.5.1 The system shall be able to quit game application.

3.3 Completing A Level

3.3.1 The system must congratulate user on winning current level.

3.3.1.1 The user shall be able to see a background image on scene.

3.3.1.2 The user shall be able to see congratulating text displaying, "Level Complete".

3.3.1.3 The user shall be able to see a menu.

3.3.2 The user shall be able to interact with menu buttons.

3.3.2.1 The system must display the pointing device cursor.

3.3.2.2 The user shall be able to move cursor using pointing device.

3.3.3 The user shall be able to go to next level using button.

3.3.3.1 The system must redirect user to next level.

3.3.4 The user shall be able to go back to main menu with buttons using pointing device.

3.3.4.1 The system must redirect user to main menu.

3.3.5 The user shall be able to quit game with buttons using pointing device.

3.3.5.1 The system shall be able to quit the game application.

3.4 Losing The Game

3.4.1 The system must display a game over text when user loses the game.

3.4.1.3 The user shall be able to see a menu.

3.4.2 The user shall be able to interact with menu.

3.4.2.1 The system must display pointing device cursor.

3.4.2.2 The user shall be able to distinguish buttons by hovering over them.

3.4.2.3 The user shall be able to click game over graphical user interface buttons.

3.4.3 The user shall be able to play the game again.

3.4.3.1 The system must redirect user to level one.

3.4.3.2 The system must reset level timer and GUI's.

3.4.4 The user shall be able to go to back to main menu.

3.4.4.1 The system must redirect user to main menu.

3.4.5 The user shall be able to quit the game application with buttons using pointing device.

3.4.5.1 The system must quit the game application.

3.5 Winning The Game

3.5.1 The system must redirect user to credits screen upon completion of all levels.

3.5.1.1 The user shall be able to see text displaying "Thank You For Playing".

3.5.1.2 The user shall be able to see text displaying project developers' names.

3.5.1.3 The user shall be able to see a menu.

3.5.2 The user shall be able to interact with buttons on menu.

3.5.2.1 The system must display pointing device cursor.

3.5.2.2 The user shall be able to move cursor using pointing device.

3.5.2.3 The user shall be able to click a button.

3.5.3 The user shall be able to play again the game.

3.5.3.1 The system must redirect user to level one.

3.5.3.2 The system must lock the pointing device cursor on transition of scenes.

3.5.3.3 The system must hide the pointing device cursor on transition of scenes.

3.5.3.4 The system must reset any timer and GUI's.

3.5.4 The user shall be able to go back to menu with buttons using pointing device.

3.5.4.1 The system shall be able to redirect user to main menu.

3.5.5 The user to be able to quit game with buttons using pointing device.

3.5.5.1 The system must quit game application.

3.6 Levels

3.6.1 The user shall be able to play different levels in the game.

3.6.1.1 The system must load current level.

3.6.1.2 The system must load next level only when user completes the previous level.

3.6.2 The system must load the corresponding environment for the current level.

3.6.2.1 The user shall be able to see a new environment for each new level.

3.6.3 The user shall be able to see new weapon on player for each level.

3.6.3.1 The system must load the corresponding weapon for the corresponding level.

3.6.3.2 The system must make the bullet on the weapon faster than the previous level.

3.6.3.3 The user shall be able to see a shooting and reloading animation on each level.

3.6.3.4 The user shall be able to see ammo left on weapon.

3.6.4 The system must increase enemy's difficulty on next level.

3.6.4.1 The system must increase the enemy's velocity.

3.6.4.2 The system must make enemies smaller in scale size.

3.6.5 The system must load timer on each level.

3.6.5.1 The system must make timer start when level starts.

3.6.5.2 The system must make each level the same time limit.

3.7 Graphical User Interface

3.7.1 The user shall be able to see timer displayed as a text.

3.7.1.1 The user shall be able to distinguish where the timer is located at.

3.7.1.2 The user shall be able to see timer going down in seconds.

3.7.2 The user shall be able to see remaining enemies.

3.7.2.1 The system must display enemies alive as an image.

3.7.4 The user shall be able to see how much ammo their weapon has left.

3.7.4.1 The system must count ammo.

3.7.4.2 The system must reset the ammo when user reloads weapon.

3.7.4.3 The system must display ammo left as a text.

3.7.4.4 The user shall be able to distinguish where the ammo text is located.

3.7.5 The user shall be able to scope using a sniper rifle.

3.7.5.1 The system must display scope when pointing device button is pressed.

3.7.5.2 The system must set new point of view when weapon is scoped.

3.7.5.3 The system must reset the point of view when user stops scoping weapon.

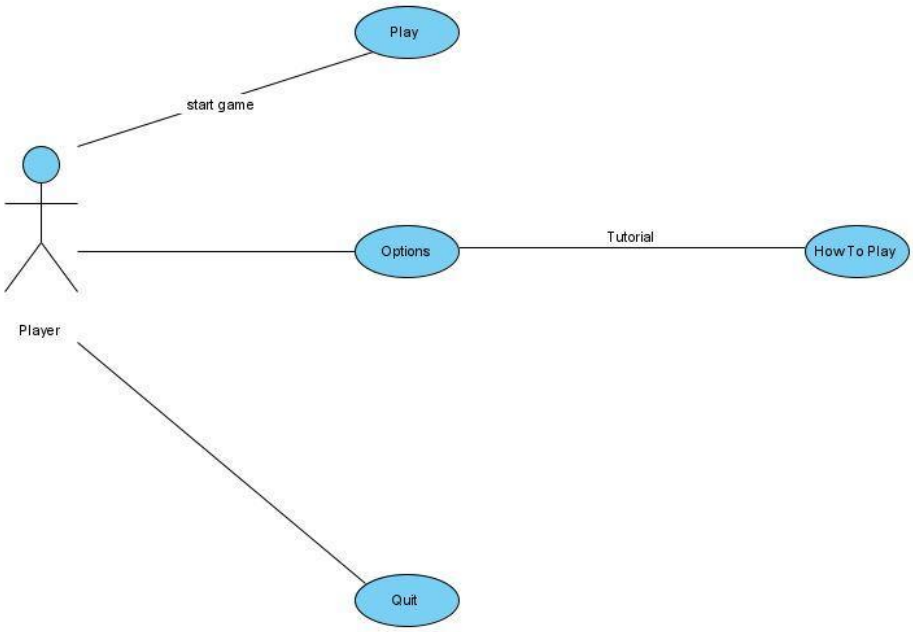
3.7.6 The user shall be able have a crosshair to aim with shotgun weapons.

3.7.6.1 The user shall be able to see crosshair in the center of their screen.

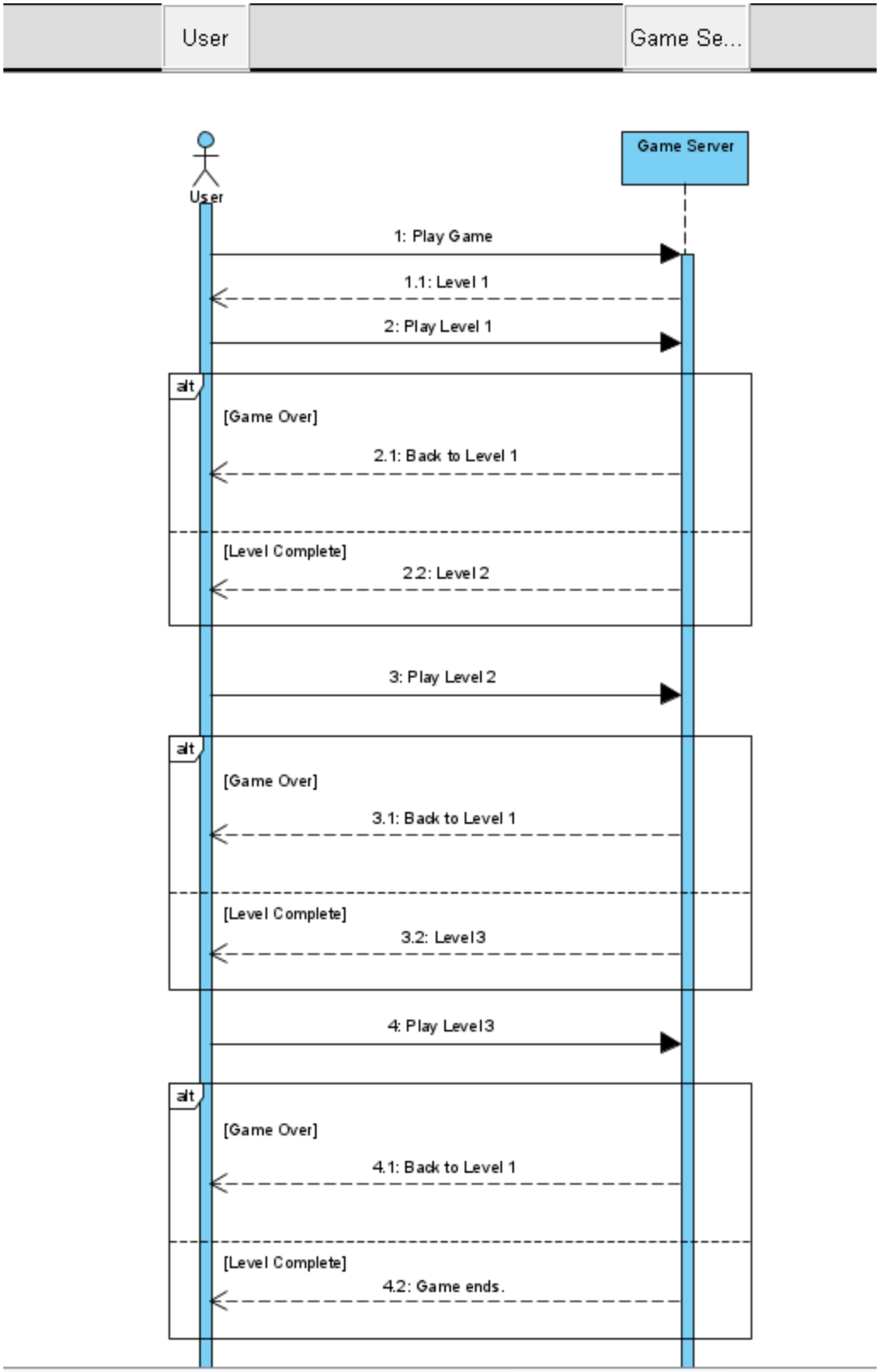
3.7.6.2 The user shall be able to use pointing device to move crosshair.

4. Diagrams

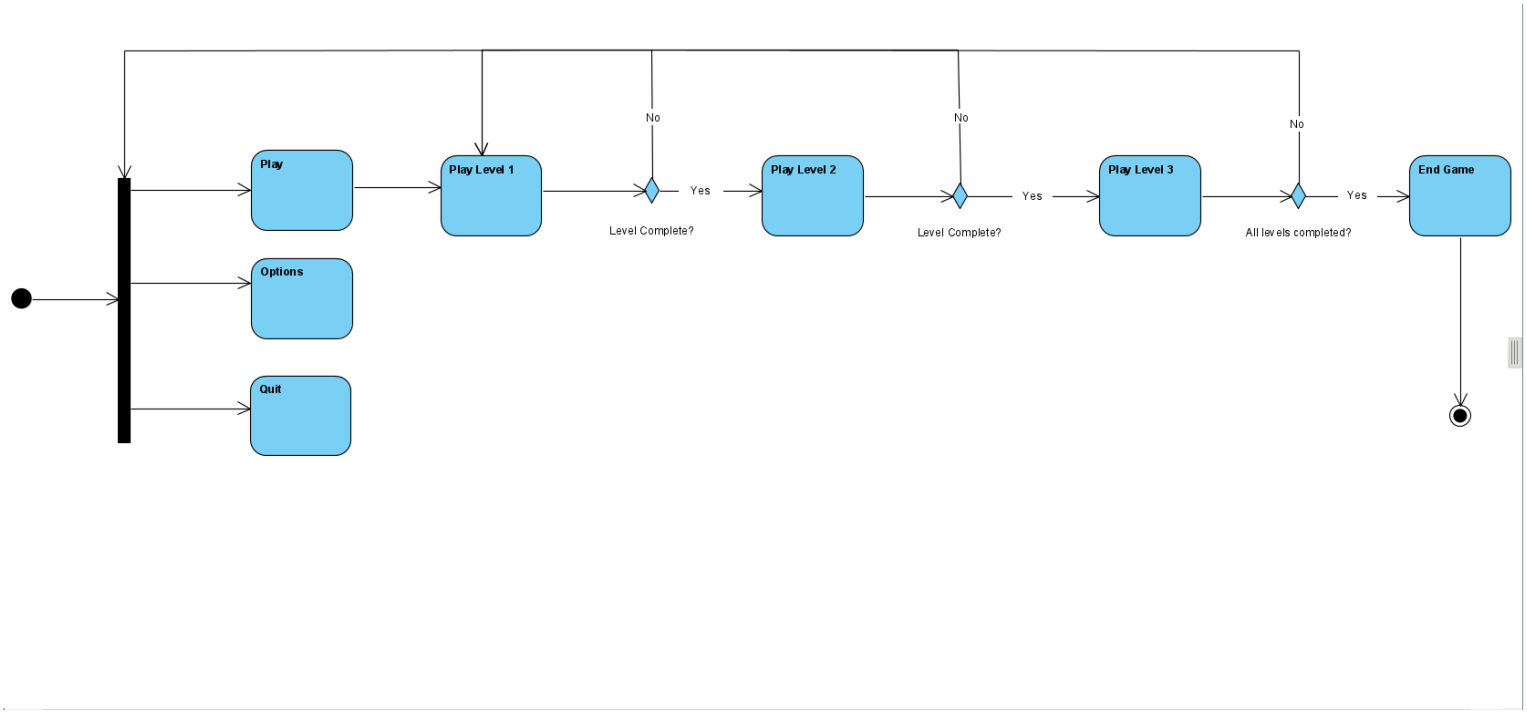
4.1 Use Case Diagram



4.2 Game Sequence Diagram



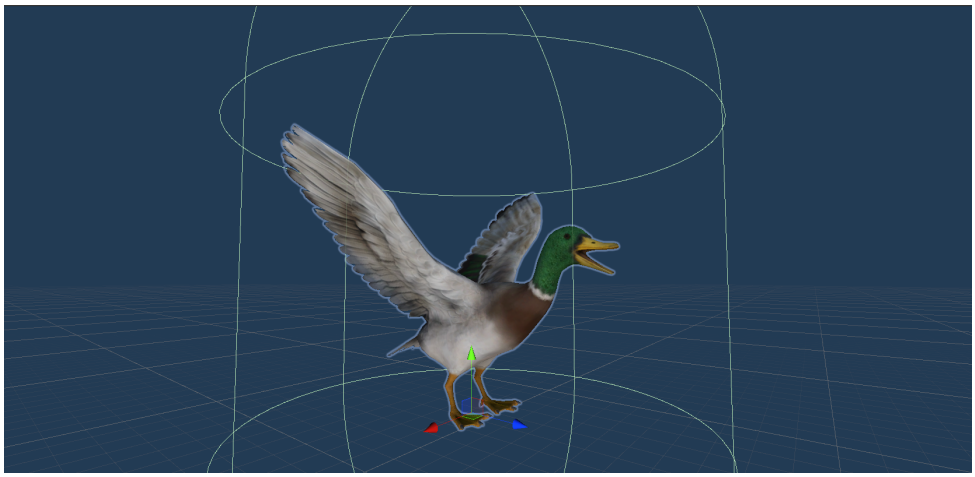
4.3 Game Flow Activity Diagram



5. Game Elements

5.1 Player

5.2 Enemy



5.3 Main Menu


DUCK HUNT
PLAY
OPTIONS
QUIT


DUCK HUNT
HOW TO PLAY
CREDITS
BACK

5.4 Levels

5.4.1 Level 1

5.4.2 Level 2

5.4.3 Level 3

5.5 Level Complete

5.6 Game Over

5.7 Pause Mode

Appendix A C-Sharp Implementation

The game was developed in C#.

The player movements, enemy movement, enemy spawner, shooting the weapon, using scope on weapon, scene transitions, GUI buttons functionality, GUI timer, GUI enemies left, and GUI ammo, were all coded in C-Sharp language using Visual Studio 2019 (free version) IDE.

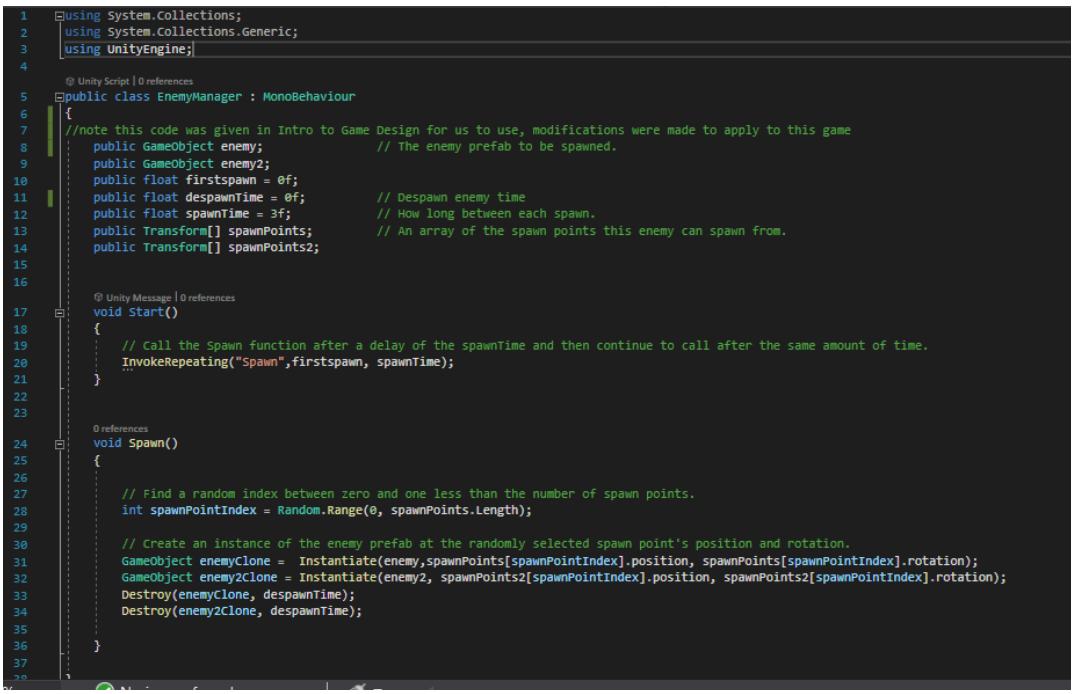
Duck Movement Script

The duck movement script moves the enemy between two bounds. Once the enemy reaches the bound it will turn to the other bound. On the code comments are attached explaining what the code those.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

Unity Script | 0 references
public class MovingBird : MonoBehaviour
{
    private Animator animator;
    private bool dead = false;

    // left and right marks
    public Transform left;
    public Transform right;
    // speed
    public float speed = 1.0f;

    // current direction (false means to the left, true means to the right)
    bool dir = false;
    Unity Message | 0 references
    private void Start()
    {
        animator = GetComponent<Animator>();

    }
    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (dead == true)
        {
            return;
        }
        if (dir)
        {
            transform.LookAt(right.position);
            // go closer to the right one
            transform.position = Vector3.MoveTowards(transform.position,
                                                     right.position,
                                                     Time.deltaTime * speed);
            // reached it?
            if (transform.position == right.position)
            {
                dir = !dir; // go to opposite direction next time
            }
        }
        else
        {
            transform.LookAt(left.position);
            // go closer to the left one
            transform.position = Vector3.MoveTowards(transform.position,
                                                     left.position,
                                                     Time.deltaTime * speed);

            // reached it?
            if (transform.position == left.position)
            {
                dir = !dir; // go to opposite direction next time
            }

        }
    }
    0 references
    public void Die()
    {
        //print("dying");
        Debug.Log("bro why");
        dead = true;
        animator.SetBool("dead", true);


    }
}
```

Enemy Spawner Script

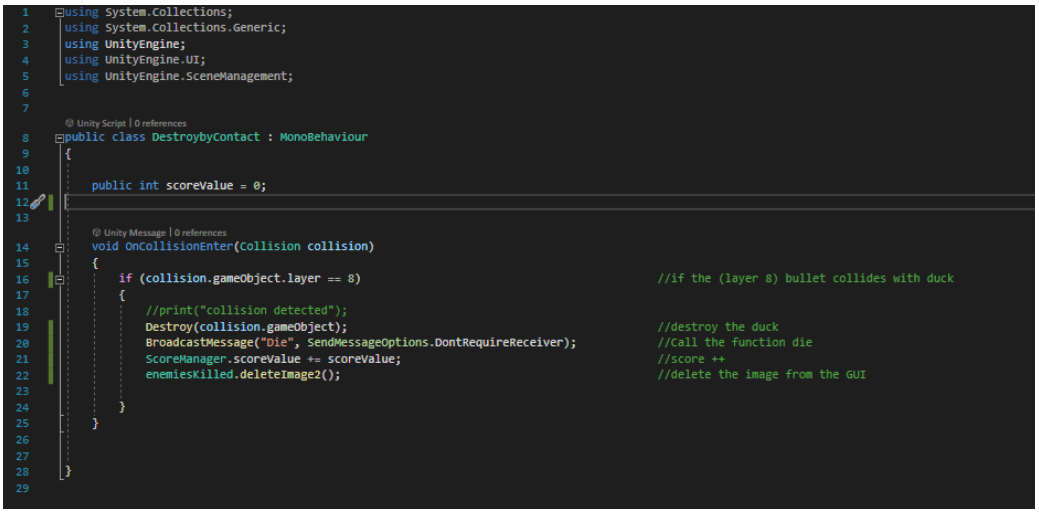
This script spawns the enemy at the spawning locations we desire. On the code comments are attached explaining what the code those.



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

Unity Script | 0 references
public class EnemyManager : MonoBehaviour
{
//note this code was given in Intro to Game Design for us to use, modifications were made to apply to this game
    public GameObject enemy;              // The enemy prefab to be spawned.
    public GameObject enemy2;
    public float firstspawn = 0f;
    public float despawnTime = 0f;        // Despawn enemy time
    public float spawnTime = 3f;          // How long between each spawn.
    public Transform[] spawnPoints;       // An array of the spawn points this enemy can spawn from.
    public Transform[] spawnPoints2;


    Unity Message | 0 references
    void Start()
    {
        // Call the Spawn function after a delay of the spawnTime and then continue to call after the same amount of time.
        InvokeRepeating("Spawn",firstspawn, spawnTime);
    }


    0 references
    void Spawn()
    {

        // Find a random index between zero and one less than the number of spawn points.
        int spawnPointIndex = Random.Range(0, spawnPoints.Length);

        // Create an instance of the enemy prefab at the randomly selected spawn point's position and rotation.
        GameObject enemyClone =  Instantiate(enemy,spawnPoints[spawnPointIndex].position, spawnPoints[spawnPointIndex].rotation);
        GameObject enemy2Clone = Instantiate(enemy2, spawnPoints2[spawnPointIndex].position, spawnPoints2[spawnPointIndex].rotation);
        Destroy(enemyClone, despawnTime);
        Destroy(enemy2Clone, despawnTime);

    }
```

Destroy By Contact Script

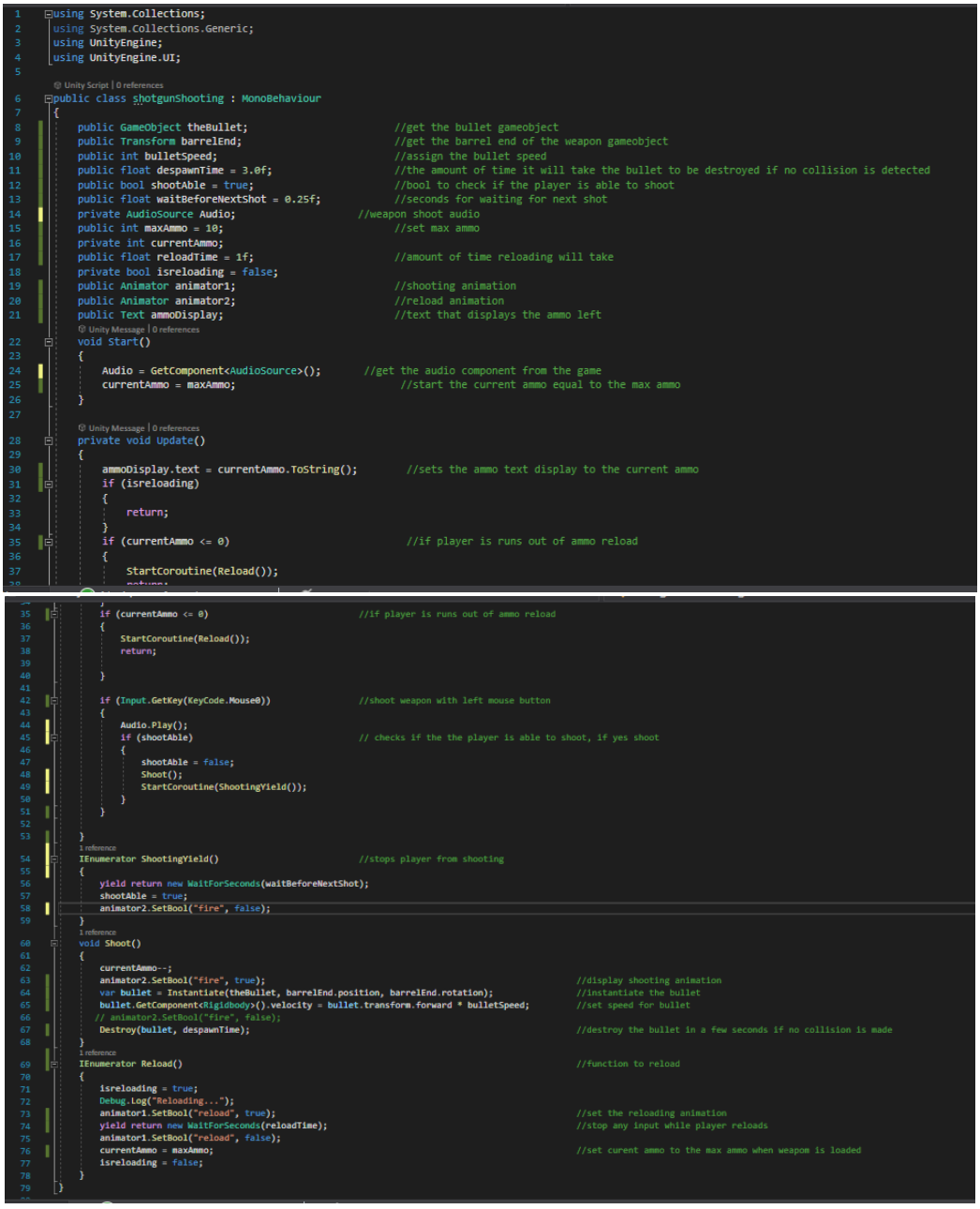
This script destroys the enemy and the bullet if the bullet collides with the enemy game object. On the code comments are attached explaining what the code those.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;


Unity Script | 0 references
public class DestroybyContact : MonoBehaviour
{

    public int scoreValue = 0;


    Unity Message | 0 references
    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.layer == 8)                                    //if the (layer 8) bullet collides with duck
        {
            //print("collision detected");
            Destroy(collision.gameObject);                                      //destroy the duck
            BroadcastMessage("Die", SendMessageOptions.DontRequireReceiver);    //Call the function die
            ScoreManager.scoreValue += scoreValue;                              //score ++
            enemiesKilled.deleteImage2();                                       //delete the image from the GUI

        }
    }


}
```

Shooting Script

This script is used to shoot the weapon, the player can only shoot one bullet at a time and has to wait one second to be able to shoot again. Once the player rans out of bullets the reload happens automatically and the player has to wait two seconds for the weapon to reload. On the code comments are attached explaining what the code those.



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

Unity Script | 0 references
public class shotgunShooting : MonoBehaviour
{
    public GameObject theBullet;                    //get the bullet gameobject
    public Transform barrelEnd;                     //get the barrel end of the weapon gameobject
    public int bulletSpeed;                         //assign the bullet speed
    public float despawnTime = 3.0f;                //the amount of time it will take the bullet to be destroyed if no collision is detected
    public bool shootAble = true;                   //bool to check if the player is able to shoot
    public float waitBeforeNextShot = 0.25f;        //seconds for waiting for next shot
    private AudioSource Audio;                    //weapon shoot audio
    public int maxAmmo = 10;                        //set max ammo
    private int currentAmmo;
    public float reloadTime = 1f;                   //amount of time reloading will take
    private bool isreloading = false;
    public Animator animator1;                      //shooting animation
    public Animator animator2;                      //reload animation
    public Text ammoDisplay;                        //text that displays the ammo left
    Unity Message | 0 references
    void Start()
    {
        Audio = GetComponent<AudioSource>();       //get the audio component from the game
        currentAmmo = maxAmmo;                      //start the current ammo equal to the max ammo
    }

    Unity Message | 0 references
    private void Update()
    {
        ammoDisplay.text = currentAmmo.ToString();       //sets the ammo text display to the current ammo
        if (isreloading)
        {
            return;
        }
        if (currentAmmo <= 0)                           //if player is runs out of ammo reload
        {
            StartCoroutine(Reload());
            return;

        }

        if (Input.GetKey(KeyCode.Mouse0))               //shoot weapon with left mouse button
        {
            Audio.Play();
            if (shootAble)                              // checks if the the player is able to shoot, if yes shoot
            {
                shootAble = false;
                Shoot();
                StartCoroutine(ShootingYield());
            }
        }

    }
    1 reference
    IEnumerator ShootingYield()                         //stops player from shooting
    {
        yield return new WaitForSeconds(waitBeforeNextShot);
        shootAble = true;
        animator2.SetBool("fire", false);
    }
    1 reference
    void Shoot()
    {
        currentAmmo--;
        animator2.SetBool("fire", true);                                    //display shooting animation
        var bullet = Instantiate(theBullet, barrelEnd.position, barrelEnd.rotation);        //instantiate the bullet
        bullet.GetComponent<Rigidbody>().velocity = bullet.transform.forward * bulletSpeed;     //set speed for bullet
       // animator2.SetBool("fire", false);
        Destroy(bullet, despawnTime);                                       //destroy the bullet in a few seconds if no collision is made
    }
    1 reference
    IEnumerator Reload()                                                    //function to reload
    {
        isreloading = true;
        Debug.Log("Reloading...");
        animator1.SetBool("reload", true);                                  //set the reloading animation
        yield return new WaitForSeconds(reloadTime);                        //stop any input while player reloads
        animator1.SetBool("reload", false);
        currentAmmo = maxAmmo;                                              //set curent ammo to the max ammo when weapom is loaded
        isreloading = false;
    }
}
```

Appendix B Unity Engine

We used the Unity Engine version 2020.1.6f1 Personal to build the game. Unity is an amazing game engine with a free version and paid version. Unity asset store lets you buy or find free assets made by many developers; assets are easy to download and easy to import and implement in your project. Unity building tools makes it easy to build whatever you need for your project, for example terrains. Unity terrain builder lets you build beautiful realistic terrain making your game look fantastic. Moreover, built-in functions let you create amazing Main Menus, attach scripts to object, and many more.

The figures below are snapshots of prefab components to show how the prototype was built. In the first image we have our weapon prefab that we attached the shooting script and we set each value inside the Unity Engine, Unity makes it easier, so you do not have to go back and change it in the code. On the second image we have our enemy, in which we attached our prefab to a sphere, then the mesh renderer was removed, this helped us get a collider for our enemy. Then Movement and Destroy by Contact scripts were attached, additionally, we implemented the game object into our game. To conclude, the final image demonstrates how we used Unity built-in on click function. The on click function allows you to attach a script and call a function in the script, with this we created a function to play the game which redirects the user to level 1. We used this built on click function in all our GUI buttons in the game. Overall, Unity is a very us ed us short ent time.