**ACKNOWLEDGEMENT**

I like to extend my gratitude to Dr. Sonali Pednekar, our principal and all staff of Mulund College of Commerce for providing us moral support, conducive work environment and the much needed inspiration to complete this project on time.

I also take this opportunity to thank our Course Coordinator Dr. Reena Nagda and all the faculty members of Department of Computer Science for giving us the most needed guidance and continuous encouragement throughout the duration of the programme.

I wish to extend my deepest gratitude and special thanks to my project guide Ms. Bhumika Nakum, for giving their generous support, necessary inputs and companionship during my project work.

I would like to convey my special thanks to the Management and all the staff of the college for providing the required infrastructure and resource to enable the completion and enrichment of my project.

I am extremely grateful to the University of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally, I thank all my fellow friends who have directly or indirectly helped me in completing my project.

Vaibhav M. Kamtekar

**ACKNOWLEDGEMENT**

I have a great pleasure in representing this project report entitled “ Desert Hawk ” and I grab this opportunity to convey my immense regards towards all the distinguished people who have their valuable contribution in the hour of need.

I like to extend my gratitude to our beloved Principal Dr. Sonali Pednekar for her timely and prestigious guidance.

I take this opportunity to thank Dr. Reena Nagda, Coordinator of the Department and all the faculty members of the Department of Computer Science of Mulund College of Commerce, for giving me an opportunity to complete this project and the most needed guidance throughout the duration of the programme.

I am extremely grateful to my project guide Dr. Vaishnavi Assarfor her valuable guidance and necessary support during each phase of the project. She was the source of continuous encouragement as each milestone was crossed.

A special thanks to the University of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally, I also owe to my fellow friends who have been a constant source of help to solve the problems that cropped up during the project development process.

Name\_of\_the\_student

1. **Desert Hawk**
2. **Introduction**
   1. **Objective of the project:**

* This is a 2D single player game designed for Android and PCs.
* This gaming project is easy to operate and understood.
* Talking about the gameplay, User has to dodge the hawk from enemies and destroy in order to gain score points.
* Recently, the video game market appears to be of an unprecedented stage, which means the springing up of more platforms lead to more competition.
* The video game market is not just serviced for PC, PS3 and Xbox. The mobile platforms basis on iOS, Android and Windows Phone rise sharply. As a result, “cross-platform” come into people’s eyes.
  1. **Description of the current system:**
* The language used for the development of this simple game is " C# language ".
* This project mainly deals with the development of a 3D game application with the Unity 4 game engine for Windows OS.
* The game engine used for the project was Unity 4. Unity 4 was developed by the Unity Technologies.
* The theme throughout the game project is the word mechanic to relate to the actions which are taking place in games from the internal operations of animation and programming to the interactions between the environment and the player.
  1. **Limitations of current system:**
* Integrated Gamming PC’s are required.
* Puts so much load on the devices.
* Not enough smooth gameplay.

**2.4 Description of proposed system:**

* The game can be played without any special training.
* There are different stages available for the player.
* The bullets are fired automatically after the game
* PC controls are (Left, Right Arrow Keys for movement].
* The game will be more accurate and reliable than the current one.
* The language used for this simple game is C# .
* User has to dodge the hawk from enemies and destroy in order to gain score points.

**2.5 Advantages of proposed system**

* + Reduced time consumption.
  + No extra equipments are necessary.
  + No loss of records.
  + Can find what you need quickly.

1. **Requirement Specification**

**3.1 Software Requirement**

* + Operating System – Microsoft Windows 2010
  + Application Software – Front end: C#

Back end: SQL

* + Platform – Unity.

**3.2 Hardware Requirement**

* + Processor - 1.6 GHz
  + RAM - 4GB or more
  + Disk Space – 3 GB or less

**3.3 Data Requirement:**

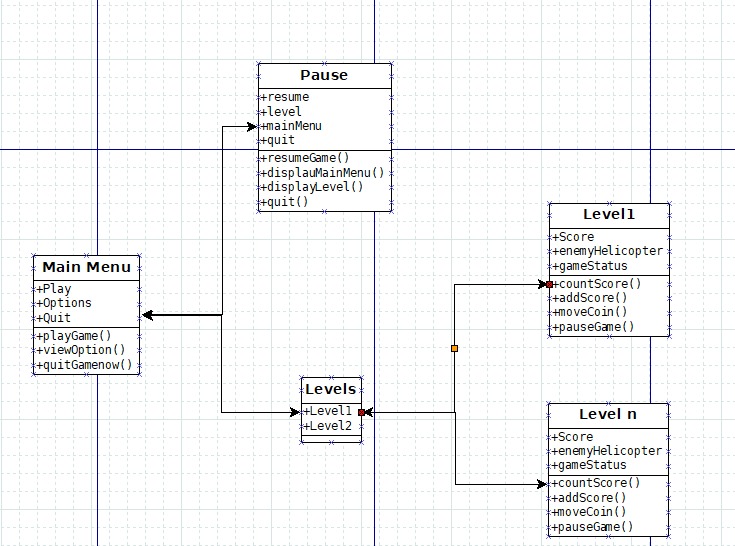
* + Unique Id.
  1. **Fact finding questions:**
  + Why do you want a game with Helicopter shooting?
* Is it expected to have any future scope?
  + What hardware and software will the system product be set up on?
  + How hard a game level can be?

1. **System Design Details**

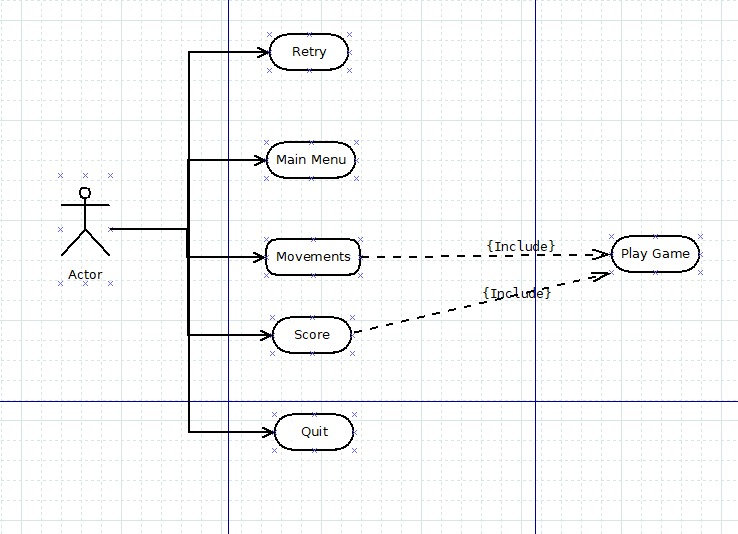
**4.1 Event Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Event | Source | Activity | Response |
| 1 | Touch | User | Moves the helicopter right/left | Moves the helicopter right/left |
| 2 | Quit | User | Stops all activity | Exists the game |
| 3 | Restart | User | Restarts all the activities in game | Restarts the game from the beginning |
| 4 | Main Menu | User | Redirects to a specific page | Redirects to the Main menu |

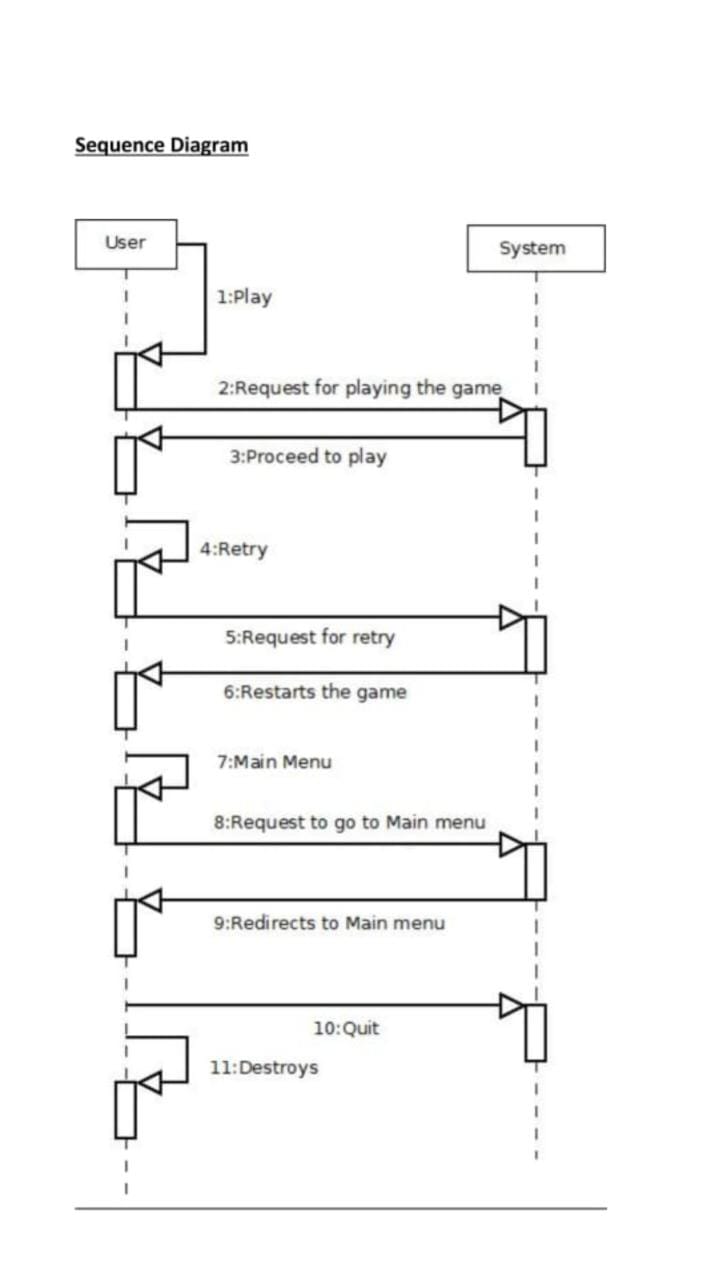
**4.2 Class Diagram:**



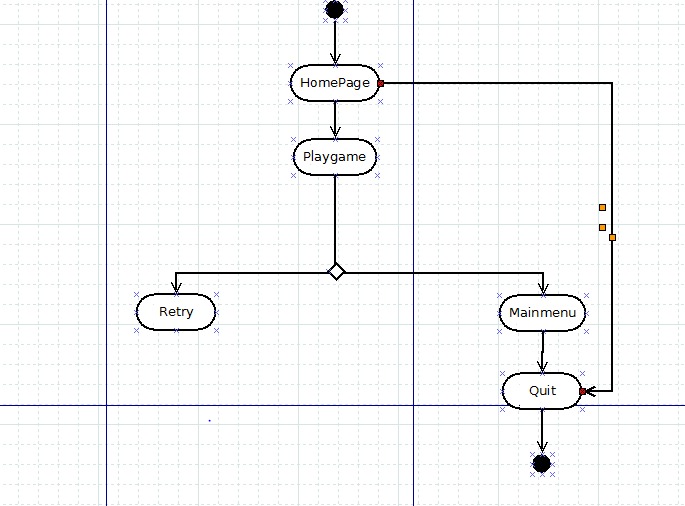
**4.3 Use Case Diagram:**



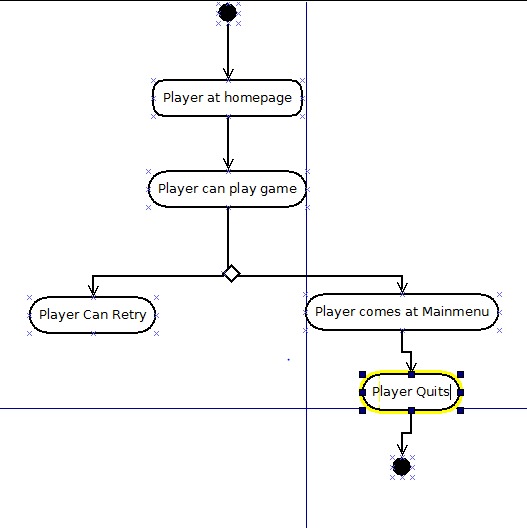
**4.4 Sequence Diagram:**

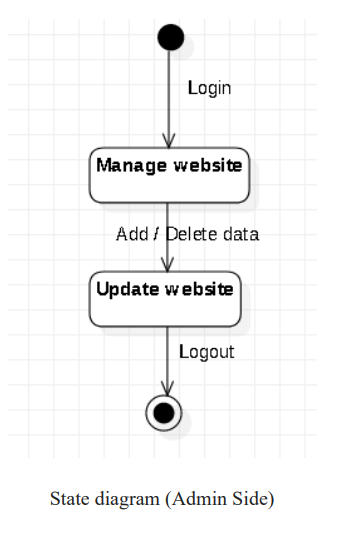


**4.5 Activity Diagram:**

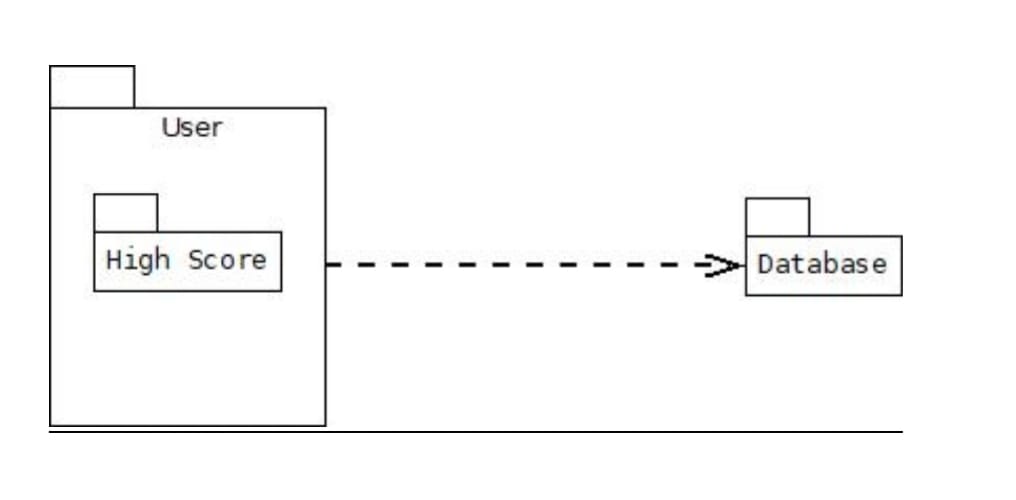


**4.6 State Diagram:**

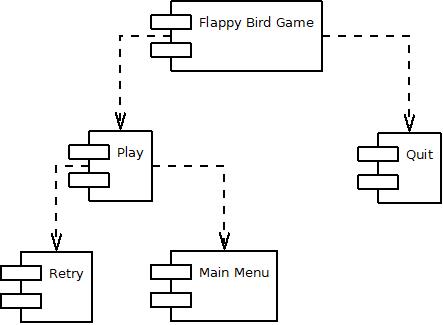




**4.7 Package Diagram:**

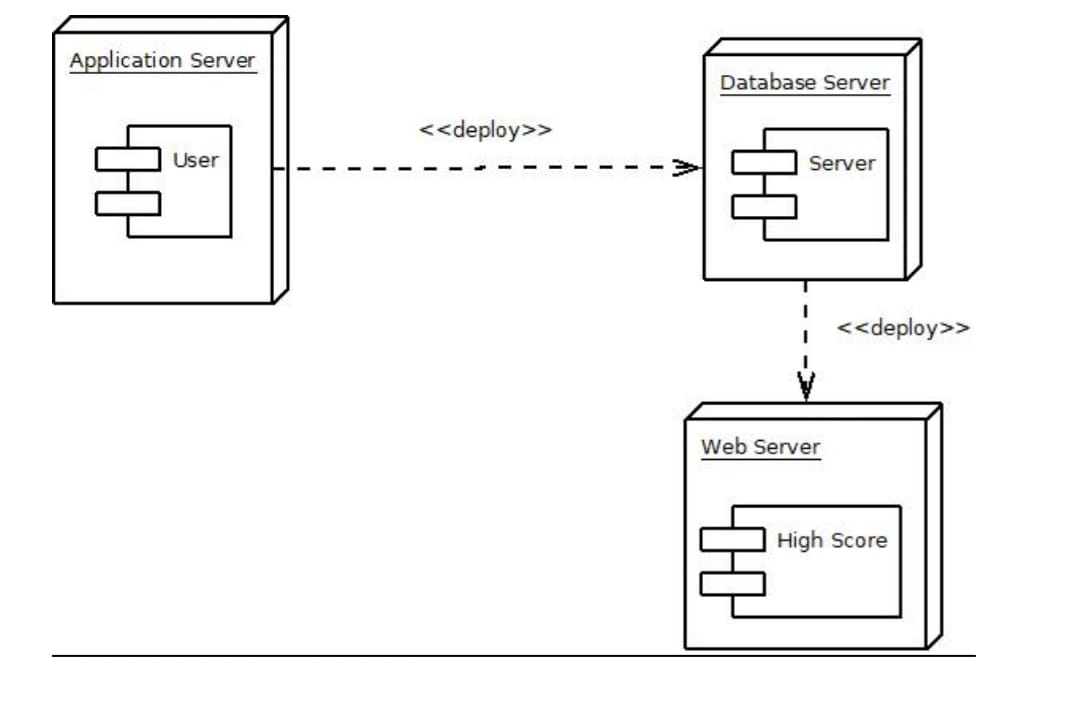


**4.8 Component Diagram:**

****

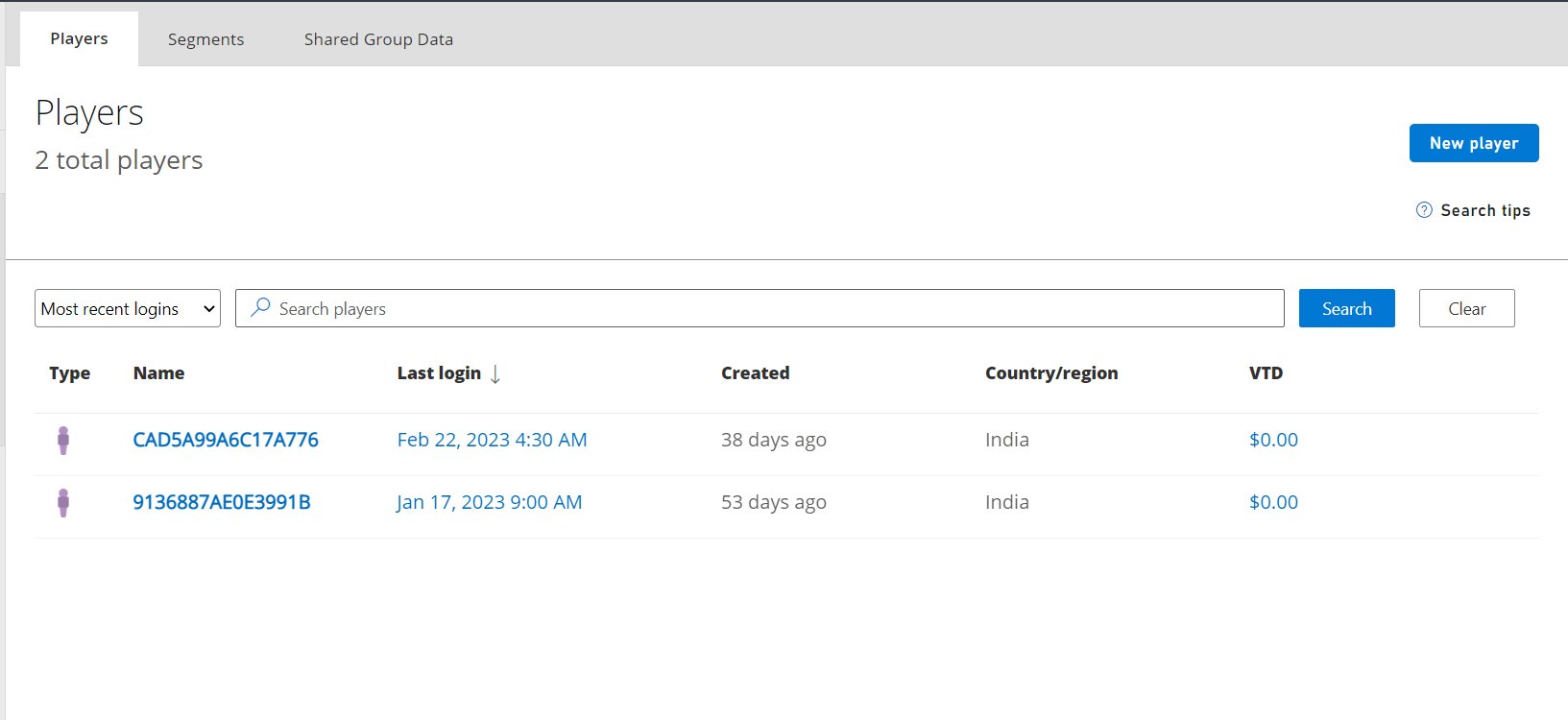
DesertHawkGame

**4.9 Deployment Diagram:**

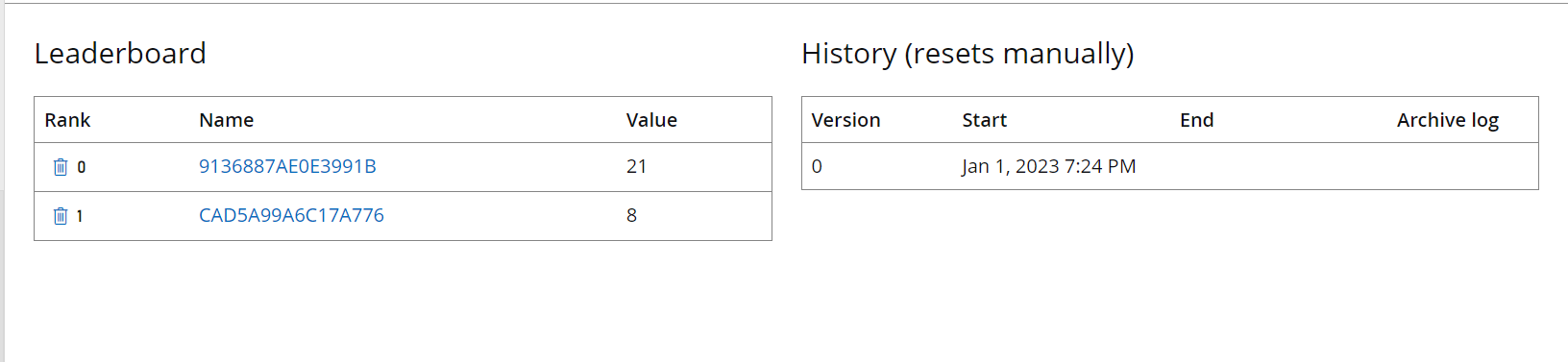


**4.10 Database Diagram:**

4.10.1 Players Information:



4.10.1 Players Highscore:



5. IMPLEMENTATION:

**1)Background.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Background : MonoBehaviour {

public float speed;

private Vector2 offset = Vector2.zero;

private Material mat;

void Awake(){

}

// Use this for initialization

void Start () {

InitializeBackground ();

}

// Update is called once per frame

void Update () {

if (GameplayController.instance.startMoving) {

ScrollMovement ();

}

}

void InitializeBackground(){

mat = GetComponent<Renderer> ().material;

offset = mat.GetTextureOffset ("\_MainTex");

float height = Camera.main.orthographicSize \* 2f;

float width = height \* Screen.width / Screen.height;

transform.localScale = new Vector3 (width, height + 1, transform.position.z);

}

void ScrollMovement(){

offset.y += speed \* Time.deltaTime;

mat.SetTextureOffset ("\_MainTex", offset);

}

}

2)Cloud.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Cloud : MonoBehaviour {

public float speed;

private float maxTop, maxBottom;

private Vector3 target;

void Awake(){

InitializeVariables ();

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if (transform.position != target) {

transform.position = Vector3.MoveTowards (transform.position, target, speed \* Time.deltaTime);

} else {

Destroy (gameObject);

GameplayController.instance.InitializeSpawners ();

}

}

void InitializeVariables(){

Vector3 topBound = Camera.main.ViewportToWorldPoint (new Vector3 (0, 1, 0));

Vector3 bottomBound = Camera.main.ViewportToWorldPoint (new Vector3 (0, 0, 0));

Vector3 temp = transform.position;

target = new Vector3(0, maxBottom - 20, 0);

maxTop = topBound.y;

maxBottom = bottomBound.y;

temp.y = maxTop + 11;

transform.position = temp;

}

}

3)BossBulet.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BossBullet : MonoBehaviour {

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

void OnTriggerEnter2D(Collider2D collider){

if(collider.CompareTag("Player")){

collider.transform.GetComponent<PlayerController> ().PlayerDied ();

}

}

}

4)BossController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BossController : MonoBehaviour {

public float speed;

public Transform targetPoint;

void Awake(){

targetPoint = GameObject.Find ("Target Point").transform;

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if (transform.position != targetPoint.position) {

transform.position = Vector3.MoveTowards (transform.position, targetPoint.position, speed \* Time.deltaTime);

} else {

transform.GetComponent<BossHealth> ().invulnerable = false;

transform.GetComponent<BossShoot> ().isReadyToShoot = true;

}

}

}

5)BossHealth.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class BossHealth : MonoBehaviour {

public int maxHealth;

public Slider health;

public bool invulnerable;

public GameObject explode, coin;

void Awake(){

InitializeBossVariables ();

}

// Use this for initialization

void Start () {

}

void InitializeBossVariables(){

health.maxValue = maxHealth;

health.value = maxHealth;

invulnerable = true;

}

public void Health(int damage){

if (!invulnerable) {

if (!health.gameObject.activeInHierarchy) {

health.gameObject.SetActive (true);

}

if (health.value > 0) {

health.value -= damage;

}

if (health.value == 0) {

BossDestroyed ();

}

}

}

void BossDestroyed(){

Destroy (gameObject);

GameObject.FindGameObjectWithTag ("Spawner").transform.GetComponent<EnemySpawner> ().active = true;

GameObject.FindGameObjectWithTag ("Spawner").transform.GetComponent<EnemySpawner> ().isBossReady = false;

if(GameController.instance != null && MusicController.instance != null){

if(GameController.instance.isMusicOn){

MusicController.instance.audioSource.PlayOneShot (MusicController.instance.bossExplode);

}

}

Instantiate (explode, transform.position, Quaternion.identity);

for (int i = 0; i < 5; i++) {

Instantiate (coin, transform.position, Quaternion.identity);

}

}

}

6)BossShoot.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BossShoot : MonoBehaviour {

public GameObject bullet;

public bool isReadyToShoot, fireBullet;

private float fireRate, firstDelay, secondDelay;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if(isReadyToShoot){

fireRate = Random.Range(4, 6);

firstDelay += Time.deltaTime;

if (firstDelay >= fireRate) {

if (!fireBullet) {

fireBullet = true;

if (GameObject.FindGameObjectWithTag ("Player") != null) {

GameObject newBullet = Instantiate (bullet, new Vector3 (transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;

Transform target = GameObject.FindGameObjectWithTag ("Player").transform;

newBullet.GetComponent<Rigidbody2D> ().velocity = (target.position - transform.position).normalized \* 5f;

} else {

GameObject newBullet = Instantiate (bullet, new Vector3 (transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = new Vector2 (Random.Range(-1f, 1f), -2f);

}

}

if (fireBullet) {

secondDelay += Time.deltaTime;

if (secondDelay >= 0.5f) {

secondDelay = 0;

firstDelay = 0;

if (GameObject.FindGameObjectWithTag ("Player") != null) {

GameObject newBullet = Instantiate (bullet, new Vector3 (transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;

Transform target = GameObject.FindGameObjectWithTag ("Player").transform;

newBullet.GetComponent<Rigidbody2D> ().velocity = (target.position - transform.position).normalized \* 5f;

} else {

GameObject newBullet = Instantiate (bullet, new Vector3 (transform.position.x, 0.3f, transform.position.z), Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = new Vector2 (Random.Range(-0.1f, 0.1f), -2f);

}

fireBullet = false;

}

}

}

}

}

}

7)GameBoundry.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GameBoundary : MonoBehaviour {

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

void OnTriggerEnter2D(Collider2D collider){

if(collider.CompareTag("Player Bullet") || collider.CompareTag("Enemy") || collider.CompareTag("Coin") || collider.CompareTag("Boss Bullet")){

Destroy (collider.gameObject);

}

}

}

8)EnemyController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyController : MonoBehaviour {

public float speed;

private Rigidbody2D myRigidBody;

void Awake(){

myRigidBody = GetComponent<Rigidbody2D> ();

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

EnemyMovement ();

}

void EnemyMovement(){

myRigidBody.velocity = new Vector2 (myRigidBody.velocity.x, -speed);

}

void OnTriggerEnter2D(Collider2D collider){

if(collider.CompareTag("Player")){

collider.transform.GetComponent<PlayerController> ().PlayerDied ();

}

}

}

9)EnemyHealth.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class EnemyHealth : MonoBehaviour {

public Slider health;

public int maxHealth;

public GameObject explosion, coin;

private bool spawn;

void Awake(){

health.maxValue = maxHealth;

health.value = maxHealth;

}

// Use this for initialization

void Start () {

}

public void Health(int damage){

if(!health.gameObject.activeInHierarchy){

health.gameObject.SetActive (true);

}

if(health.value > 0){

health.value -= damage;

}

if(health.value == 0){

Destroy (gameObject);

if(!spawn){

spawn = true;

Instantiate (explosion, transform.position, Quaternion.identity);

Instantiate (coin, transform.position, Quaternion.identity);

}

}

}

}

10)EnemySpawner.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemySpawner : MonoBehaviour {

public GameObject[] enemies;

public GameObject[] boss;

public int count, maxCount;

public bool active, isBossReady;

public float time;

private float delay;

private float maxTop;

void Awake(){

}

// Use this for initialization

void Start () {

InitializeVariables ();

LimitBounds ();

}

// Update is called once per frame

void Update () {

if(GameplayController.instance.gameInProgress){

LimitBounds ();

SpawnEnemies ();

CheckedEnemies ();

}

}

void SpawnEnemies(){

if (active) {

if (count != 0) {

delay = 3;

Vector3[] position = new[] {

new Vector3 (-2, maxTop, 0),

new Vector3 (-1, maxTop, 0),

new Vector3 (0, maxTop, 0),

new Vector3 (1, maxTop, 0),

new Vector3 (2, maxTop, 0)

};

time += Time.deltaTime;

if (time > delay) {

delay = Random.Range (3, 8);

int randomEnemy = Random.Range (0, position.Length);

time = 0;

for (int i = 0; i < position.Length; i++) {

if (i != randomEnemy) {

Instantiate (enemies [0], position [i], Quaternion.Euler (0, 0, 180));

} else {

Instantiate (enemies [1], position [i], Quaternion.Euler (0, 0, 180));

}

}

count--;

}

} else {

active = false;

}

} else {

count = maxCount;

}

}

void LimitBounds(){

Vector3 topBoundary = Camera.main.ViewportToWorldPoint (new Vector3 (0, 1, 0));

maxTop = topBoundary.y;

}

void CheckedEnemies(){

if(!active){

if (GameObject.FindGameObjectWithTag("Enemy") == null) {

if(!isBossReady){

isBossReady = true;

GameplayController.instance.ShowWarning();

}

}

}

}

public void SpawnBoss(){

Instantiate (boss[0], new Vector3 (0, maxTop + 3, 0), Quaternion.Euler(0, 0, 180));

}

public void InitializeVariables(){

count = maxCount;

time = 0f;

active = true;

isBossReady = false;

}

}

11)DestroyTime.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class DestroyTime : MonoBehaviour {

void Awake(){

Destroy (gameObject, 2f);

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

}

12)WarningText.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class WarningText : MonoBehaviour {

private AudioSource audioSource;

void Awake(){

audioSource = GetComponent<AudioSource> ();

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

public void PlayWarning(){

if (GameController.instance != null && MusicController.instance != null) {

if(GameController.instance.isMusicOn){

if (audioSource.clip != null) {

audioSource.loop = true;

audioSource.Play ();

}

}

}

}

public void StopWarning(){

if(audioSource.isPlaying){

audioSource.Stop ();

}

}

}

13)GameController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

public class GameController : MonoBehaviour {

public static GameController instance;

public bool isGameStartedFirstTime;

public bool isMusicOn;

public int coins;

public int weaponLevel;

private GameData data;

void Awake(){

CreateInstance ();

InitializeGameVariables ();

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

void CreateInstance(){

if (instance != null) {

Destroy (gameObject);

} else {

instance = this;

DontDestroyOnLoad (gameObject);

}

}

void InitializeGameVariables(){

Load ();

if (data != null) {

isGameStartedFirstTime = data.GetIsGameStartedFirstTime ();

} else {

isGameStartedFirstTime = true;

}

if (isGameStartedFirstTime) {

isGameStartedFirstTime = false;

isMusicOn = true;

coins = 0;

weaponLevel = 1;

data = new GameData ();

data.SetIsGameStartedFirstTime (isGameStartedFirstTime);

data.SetIsMusicOn (isMusicOn);

data.SetCoins (coins);

data.SetWeaponLevel (weaponLevel);

Save ();

Load ();

} else {

isGameStartedFirstTime = data.GetIsGameStartedFirstTime ();

isMusicOn = data.GetIsMusicOn ();

coins = data.GetCoins ();

weaponLevel = data.GetWeaponLevel ();

}

}

public void Save(){

FileStream file = null;

try{

BinaryFormatter bf = new BinaryFormatter();

file = File.Create(Application.persistentDataPath + "/data.dat");

if(file != null){

data.SetIsGameStartedFirstTime(isGameStartedFirstTime);

data.SetIsMusicOn(isMusicOn);

data.SetCoins (coins);

data.SetWeaponLevel (weaponLevel);

bf.Serialize(file, data);

}

}catch(Exception e){

Debug.LogException (e, this);

}finally{

if(file != null){

file.Close ();

}

}

}

public void Load(){

FileStream file = null;

try{

BinaryFormatter bf = new BinaryFormatter();

file = File.Open(Application.persistentDataPath + "/data.dat", FileMode.Open);

data = bf.Deserialize(file) as GameData;

}catch(Exception e){

Debug.LogException (e, this);

}finally{

if(file != null){

file.Close ();

}

}

}

}

[Serializable]

class GameData{

private bool isGameStartedFirstTime;

private bool isMusicOn;

private int coins;

private int weaponLevel;

public void SetIsGameStartedFirstTime(bool isGameStartedFirstTime){

this.isGameStartedFirstTime = isGameStartedFirstTime;

}

public bool GetIsGameStartedFirstTime(){

return this.isGameStartedFirstTime;

}

public void SetIsMusicOn(bool isMusicOn){

this.isMusicOn = isMusicOn;

}

public bool GetIsMusicOn(){

return this.isMusicOn;

}

public void SetCoins(int coins){

this.coins = coins;

}

public int GetCoins(){

return this.coins;

}

public void SetWeaponLevel(int weaponLevel){

this.weaponLevel = weaponLevel;

}

public int GetWeaponLevel(){

return this.weaponLevel;

}

}

14)GameplayController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class GameplayController : MonoBehaviour {

public static GameplayController instance;

public playfabManager playfabmanager;

public GameObject player, statusTab, deployButton, upgradeButton, coinTab, upgradMenuPanel, arrowIcon, gameoverPanel, pausePanel, cloud, pauseMenu, pauseButton, hsPanel;

public bool gameInProgress, startMoving;

public Text warningText, coinText, currentCoinText, currentWeaponLevel, nextWeaponLevel, maxLevelText, activeWeaponLevelText, priceText, scoreText, gameoverCoinText;

public int coins, price;

public Animator upgrade;

void Awake(){

CreateInstance ();

}

// Use this for initialization

void Start () {

if(MusicController.instance != null && GameController.instance != null){

if (GameController.instance.isMusicOn) {

MusicController.instance.PlayGameplaySound ();

} else {

MusicController.instance.StopAllSound ();

}

}

pauseMenu.SetActive(false);

hsPanel.SetActive(false);

InitializeGameplayVariables ();

}

// Update is called once per frame

void Update () {

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

PausePanel ();

}

}

public void HighScore()

{

hsPanel.SetActive(true);

gameoverPanel.SetActive(false);

}

public void PauseGame()

{

pauseMenu.SetActive(true);

Time.timeScale = 0f;

}

void InitializeGameplayVariables(){

if(GameController.instance != null){

InitializeCoins ();

InitializeWeaponLevel ();

InitializeUpgradePrice ();

}

InitializePlayer ();

statusTab.SetActive (true);

deployButton.SetActive (true);

upgradeButton.SetActive (true);

coinTab.SetActive (false);

gameoverPanel.SetActive (false);

startMoving = false;

}

void InitializeCoins(){

coins = 0;

coinText.text = coins.ToString ();

currentCoinText.text = GameController.instance.coins.ToString("N0");

}

void InitializeWeaponLevel(){

if (GameController.instance.weaponLevel != 5) {

currentWeaponLevel.text = "LVL " + GameController.instance.weaponLevel.ToString ();

int newLevel = GameController.instance.weaponLevel + 1;

nextWeaponLevel.text = "LVL " + newLevel.ToString ();

maxLevelText.gameObject.SetActive (false);

activeWeaponLevelText.text = GameController.instance.weaponLevel.ToString();

} else {

arrowIcon.SetActive (false);

currentWeaponLevel.gameObject.SetActive (false);

nextWeaponLevel.gameObject.SetActive (false);

maxLevelText.gameObject.SetActive (true);

activeWeaponLevelText.text = "MAX";

}

}

void InitializeUpgradePrice(){

if (GameController.instance.weaponLevel != 5) {

price = GetPrice (GameController.instance.weaponLevel);

priceText.text = price.ToString ();

} else {

priceText.transform.GetChild (0).gameObject.SetActive (false);

priceText.text = "MAX";

}

}

private int GetPrice(int level){

int nextPrice = 0;

switch(level){

case 1:

nextPrice = 500;

break;

case 2:

nextPrice = 1200;

break;

case 3:

nextPrice = 2500;

break;

case 4:

nextPrice = 5000;

break;

}

return nextPrice;

}

void InitializePlayer(){

Instantiate (player, new Vector3 (0, -3.5f, 0), Quaternion.identity);

}

void CreateInstance(){

if(instance == null){

instance = this;

}

}

void ClearAllEnemies(){

GameObject[] enemies = GameObject.FindGameObjectsWithTag ("Enemy");

GameObject[] bossbullets = GameObject.FindGameObjectsWithTag ("Boss Bullet");

GameObject[] coins = GameObject.FindGameObjectsWithTag ("Coin");

GameObject boss = GameObject.FindGameObjectWithTag ("Boss");

if(boss != null){

Destroy (boss);

}

if(enemies != null){

foreach (GameObject enemy in enemies) {

Destroy (enemy);

}

}

if(bossbullets != null){

foreach (GameObject bossBullet in bossbullets) {

Destroy (bossBullet);

}

}

if(coins != null){

foreach (GameObject coin in coins) {

Destroy (coin);

}

}

}

public void ShowWarning(){

StartCoroutine (SpawnBossTime());

}

IEnumerator SpawnBossTime(){

warningText.gameObject.SetActive (true);

warningText.transform.GetComponent<WarningText> ().PlayWarning ();

yield return new WaitForSeconds (5f);

warningText.gameObject.SetActive (false);

warningText.transform.GetComponent<WarningText> ().StopWarning ();

GameObject.FindGameObjectWithTag ("Spawner").transform.GetComponent<EnemySpawner> ().SpawnBoss ();

}

public void DeployButton(){

statusTab.SetActive (false);

deployButton.SetActive (false);

upgradeButton.SetActive (false);

coinTab.SetActive (true);

Instantiate (cloud, new Vector3(0, 0, 0), Quaternion.identity);

startMoving = true;

}

public void UpgradeButton(){

upgradMenuPanel.SetActive (true);

}

public void CloseButton(){

upgradMenuPanel.SetActive (false);

}

public void PurchaseUpgrade(){

if(GameController.instance != null){

if(GameController.instance.weaponLevel != 5){

if (GameController.instance.coins >= price) {

GameController.instance.coins -= price;

GameController.instance.weaponLevel += 1;

GameController.instance.Save ();

InitializeCoins ();

InitializeUpgradePrice ();

InitializeWeaponLevel ();

} else {

upgrade.Play ("Upgrade Button Anim");

}

}

}

}

public void UpdateCoins(){

coins += 1;

coinText.text = coins.ToString ();

}

public void GameOver(){

gameoverPanel.SetActive (true);

gameoverCoinText.text = coins.ToString ("N0");

scoreText.text = coins.ToString ("N0");

//For LeaderBoard

string gameoverCoin = gameoverCoinText.text;

int gc = int.Parse(gameoverCoin);

playfabmanager.SendLeaderboard(gc);

//playfabmanager.GetLeaderboard();

//

if (GameController.instance.coins != null){

GameController.instance.coins += coins;

GameController.instance.Save ();

}

}

public void RetryButton(){

InitializeGameplayVariables ();

ClearAllEnemies ();

hsPanel.SetActive(false);

gameInProgress = false;

GameObject.FindGameObjectWithTag ("Spawner").transform.GetComponent<EnemySpawner> ().active = false;

}

public void InitializeSpawners(){

GameObject.FindGameObjectWithTag ("Spawner").transform.GetComponent<EnemySpawner> ().InitializeVariables ();

gameInProgress = true;

}

public void ResumeButton(){

pausePanel.SetActive (false);

Time.timeScale = 1;

gameInProgress = true;

}

public void QuitButton(){

SceneManager.LoadScene ("Main Menu");

Time.timeScale = 1;

}

void PausePanel(){

if (gameInProgress && !pausePanel.activeInHierarchy) {

pausePanel.SetActive (true);

Time.timeScale = 0;

gameInProgress = false;

} else if(!gameInProgress && !pausePanel.activeInHierarchy) {

SceneManager.LoadScene ("Main Menu");

}

}

}

15)MainmenuController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class MainMenuController : MonoBehaviour {

public GameObject quitPanel;

// Use this for initialization

void Start () {

if(GameController.instance && MusicController.instance){

if (GameController.instance.isMusicOn) {

MusicController.instance.PlayBackgroundSound ();

} else {

MusicController.instance.StopAllSound ();

}

}

}

void Update(){

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

if (!quitPanel.activeInHierarchy) {

quitPanel.SetActive (true);

} else {

quitPanel.SetActive (false);

}

}

}

public void StartButton(){

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

public void YesButton(){

Application.Quit ();

}

public void NoButton(){

if (quitPanel.activeInHierarchy) {

quitPanel.SetActive (false);

}

}

}

16)MusicController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MusicController : MonoBehaviour {

public static MusicController instance;

[HideInInspector]

public AudioSource audioSource;

public AudioClip background, gameplay, explode, coin, bossExplode;

void Awake(){

audioSource = GetComponent<AudioSource> ();

CreateInstance ();

}

// Use this for initialization

void Start () {

}

void CreateInstance(){

if (instance != null) {

Destroy (gameObject);

} else {

instance = this;

DontDestroyOnLoad (gameObject);

}

}

public void PlayBackgroundSound(){

if(background){

audioSource.clip = background;

audioSource.volume = 0.5f;

audioSource.loop = true;

audioSource.Play ();

}

}

public void PlayGameplaySound(){

if(gameplay){

audioSource.clip = gameplay;

audioSource.volume = 0.5f;

audioSource.loop = true;

audioSource.Play ();

}

}

public void StopAllSound(){

if(audioSource.isPlaying){

audioSource.Stop ();

}

}

public void PlayerDeath(){

if(explode){

audioSource.PlayOneShot (explode);

}

}

}

17)CoinMotion.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CoinMotion : MonoBehaviour {

private Rigidbody2D myRigidBody;

private float force = 5f;

void Awake(){

myRigidBody = GetComponent<Rigidbody2D> ();

myRigidBody.AddForce (new Vector2(Random.Range(-0.5f, 0.5f), force), ForceMode2D.Impulse);

}

void OnTriggerEnter2D(Collider2D collider){

if(collider.CompareTag("Player")){

Destroy (gameObject);

if(GameController.instance != null && MusicController.instance != null){

if(GameController.instance.isMusicOn){

MusicController.instance.audioSource.PlayOneShot (MusicController.instance.coin);

}

}

GameplayController.instance.UpdateCoins ();

}

}

}

18)PlayerBullet.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerBullet : MonoBehaviour {

public int damage;

public GameObject hit;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

void OnTriggerEnter2D(Collider2D collider){

if(collider.CompareTag("Enemy")){

Instantiate (hit, transform.position, Quaternion.identity);

collider.transform.GetComponent<EnemyHealth> ().Health (damage);

Destroy (gameObject);

}

if(collider.CompareTag("Boss")){

Instantiate (hit, transform.position, Quaternion.identity);

collider.transform.GetComponent<BossHealth> ().Health (damage);

Destroy (gameObject);

}

}

}

19)PlayerController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerController : MonoBehaviour {

public float speed;

public GameObject explosion;

private Vector3 position, realPos;

private float maxLeft, maxRight;

private AudioSource audioSource;

void Awake(){

InitializePlayerVariables ();

}

// Use this for initialization

void Start () {

if(GameController.instance != null && MusicController.instance != null){

if (GameController.instance.isMusicOn) {

audioSource.Play ();

} else {

audioSource.Stop ();

}

}

}

// Update is called once per frame

void Update () {

if (GameplayController.instance.gameInProgress) {

LimitPosition ();

if (Application.platform == RuntimePlatform.WindowsEditor) {

PlayerMovement ();

}else if(Application.platform == RuntimePlatform.Android){

TouchMovement ();

}

}

}

void PlayerMovement(){

if(Input.GetKey(KeyCode.LeftArrow)){

position.x -= speed \* Time.deltaTime;

}else if(Input.GetKey(KeyCode.RightArrow)){

position.x += speed \* Time.deltaTime;

}

position.x = Mathf.Clamp (position.x, maxLeft + 0.5f, maxRight - 0.5f);

transform.position = position;

}

void TouchMovement(){

if(Input.touchCount > 0){

Touch touch = Input.GetTouch (0);

if(touch.phase == TouchPhase.Stationary || touch.phase == TouchPhase.Began){

Vector3 fingerPos = touch.position;

position.x = fingerPos.x;

realPos = Camera.main.ScreenToWorldPoint (new Vector3(position.x, position.y ,position.z));

transform.position = Vector3.Lerp (transform.position, new Vector3(Mathf.Clamp(realPos.x, maxLeft + 0.5f, maxRight - 0.5f), transform.position.y, transform.position.z), Time.deltaTime \* speed);

}

if (touch.phase == TouchPhase.Moved) {

Vector3 fingerPos = touch.position;

position.x = fingerPos.x;

realPos = Camera.main.ScreenToWorldPoint (new Vector3(position.x, position.y ,position.z));

transform.position = Vector3.Lerp (transform.position, new Vector3(Mathf.Clamp(realPos.x, maxLeft + 0.5f, maxRight - 0.5f), transform.position.y, transform.position.z), speed);

}

}

}

void LimitPosition(){

Vector3 leftBound = Camera.main.ViewportToWorldPoint (new Vector3 (0, 0, Camera.main.transform.position.z));

Vector3 rightBound = Camera.main.ViewportToWorldPoint (new Vector3 (1, 0, Camera.main.transform.position.z));

maxLeft = leftBound.x;

maxRight = rightBound.x;

}

void InitializePlayerVariables(){

position = transform.position;

audioSource = GetComponent<AudioSource> ();

}

public void PlayerDied(){

if(GameController.instance != null && MusicController.instance != null){

if(GameController.instance.isMusicOn){

MusicController.instance.PlayerDeath ();

}

}

Instantiate (explosion, transform.position, Quaternion.identity);

Destroy (gameObject);

GameplayController.instance.GameOver ();

}

}

20)PlayerShoot.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerShoot : MonoBehaviour {

public GameObject bullet;

public float speed;

public AudioClip shoot;

private float time = 0f;

private float delay = 0.1f;

void Awake(){

}

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if (GameplayController.instance.gameInProgress) {

ShootBullet ();

}

}

void ShootBullet(){

if (GameController.instance != null) {

switch(GameController.instance.weaponLevel){

case 1:

time += Time.deltaTime;

if (time >= delay) {

time = 0;

Vector3[] position = new[] {

new Vector3 (transform.position.x - 0.15f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.15f, transform.position.y + 0.4f, 0)

};

for (int i = 0; i < position.Length; i++) {

GameObject newBullet = Instantiate (bullet, position [i], Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = Vector2.up \* speed;

}

if (GameController.instance != null) {

if (GameController.instance.isMusicOn) {

transform.GetComponent<AudioSource> ().PlayOneShot (shoot);

}

}

}

break;

case 2:

time += Time.deltaTime;

if (time >= delay) {

time = 0;

Vector3[] position = new[] {

new Vector3 (transform.position.x - 0.15f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.15f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x, transform.position.y + 0.4f, 0)

};

for (int i = 0; i < position.Length; i++) {

GameObject newBullet = Instantiate (bullet, position [i], Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = Vector2.up \* speed;

}

if (GameController.instance != null) {

if (GameController.instance.isMusicOn) {

transform.GetComponent<AudioSource> ().PlayOneShot (shoot);

}

}

}

break;

case 3:

time += Time.deltaTime;

if (time >= delay) {

time = 0;

Vector3[] position = new[] {

new Vector3 (transform.position.x - 0.05f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x - 0.15f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.05f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.15f, transform.position.y + 0.4f, 0)

};

for (int i = 0; i < position.Length; i++) {

GameObject newBullet = Instantiate (bullet, position [i], Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = Vector2.up \* speed;

}

if (GameController.instance != null) {

if (GameController.instance.isMusicOn) {

transform.GetComponent<AudioSource> ().PlayOneShot (shoot);

}

}

}

break;

case 4:

time += Time.deltaTime;

if (time >= delay) {

time = 0;

Vector3[] position = new[] {

new Vector3 (transform.position.x - 0.10f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x - 0.20f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.10f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.20f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x, transform.position.y + 0.4f, 0)

};

for (int i = 0; i < position.Length; i++) {

GameObject newBullet = Instantiate (bullet, position [i], Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = Vector2.up \* speed;

}

if (GameController.instance != null) {

if (GameController.instance.isMusicOn) {

transform.GetComponent<AudioSource> ().PlayOneShot (shoot);

}

}

}

break;

case 5:

time += Time.deltaTime;

if (time >= delay) {

time = 0;

Vector3[] position = new[] {

new Vector3 (transform.position.x - 0.05f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x - 0.15f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x - 0.25f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.05f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.15f, transform.position.y + 0.4f, 0),

new Vector3 (transform.position.x + 0.25f, transform.position.y + 0.4f, 0)

};

for (int i = 0; i < position.Length; i++) {

GameObject newBullet = Instantiate (bullet, position [i], Quaternion.identity) as GameObject;

newBullet.GetComponent<Rigidbody2D> ().velocity = Vector2.up \* speed;

}

if (GameController.instance != null) {

if (GameController.instance.isMusicOn) {

transform.GetComponent<AudioSource> ().PlayOneShot (shoot);

}

}

}

break;

}

}

}

}

21)playFabManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using PlayFab;

using PlayFab.ClientModels;

using UnityEngine.UI;

using Newtonsoft.Json;

public class playfabManager : MonoBehaviour

{

public GameObject rowPrefab;

public Transform rowsParent;

// Start is called before the first frame update

void Start()

{

Login();

}

void Login()

{

var request = new LoginWithCustomIDRequest

{

CustomId = SystemInfo.deviceUniqueIdentifier,

CreateAccount = true

};

PlayFabClientAPI.LoginWithCustomID(request, OnSuccess, OnError);

}

void OnSuccess(LoginResult result)

{

Debug.Log("Successful login/account create!");

}

void OnError(PlayFabError error)

{

Debug.Log("Error while logging in/creating account!");

Debug.Log(error.GenerateErrorReport());

}

public void SendLeaderboard(int score)

{

var request = new UpdatePlayerStatisticsRequest

{

Statistics = new List<StatisticUpdate>

{

new StatisticUpdate

{

StatisticName = "PlatformScore",

Value = score

}

}

};

PlayFabClientAPI.UpdatePlayerStatistics(request, OnLeaderboardUpdate, OnError);

}

void OnLeaderboardUpdate(UpdatePlayerStatisticsResult result)

{

Debug.Log("Successfull leaderboard sent");

}

//onleaderBoard Method

public void GetLeaderboard()

{

var request = new GetLeaderboardRequest

{

StatisticName = "PlatformScore",

StartPosition = 0,

MaxResultsCount = 10

};

PlayFabClientAPI.GetLeaderboard(request, OnLeaderboardGet, OnError);

}

void OnLeaderboardGet(GetLeaderboardResult result)

{

foreach (var item in result.Leaderboard)

{

GameObject newGo = Instantiate(rowPrefab, rowsParent);

Text[] texts = newGo.GetComponentsInChildren<Text>();

texts[0].text = item.StatValue.ToString();

Debug.Log(item.StatValue);

}

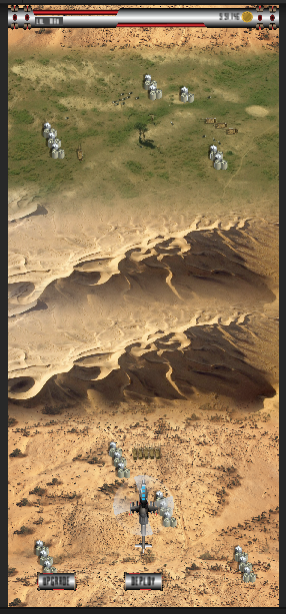
}

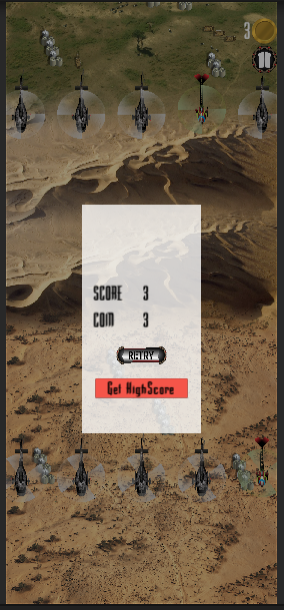
}

**6. Result:**

Screenshots:





**7.Conclusion and future scope**

**7.1.Future enhancement:**

* Adding a better UI.
* Adding new features.
* Changing the background and adding levels.
* User can change the helicopter.
* Increasing Difficulty level.

**7.2.Conclusion**

The black book has been created with the objective of providing a detail information about the game its features on what system it works and how it was developed. Also it provides the information regarding the specification required to run the game on android or desktop. It also provides us with information regarding how the bird is controlled and what future enhancements can be made.

**8.Reference**

* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)
* [www.wikipedia.com](http://www.wikipedia.com)

**9.Annexure**

|  |  |  |
| --- | --- | --- |
| Serial Number | Name of figure | Page Number |
| 4.2 | Class Diagram | 4 |
| 4.3 | Use Case Diagram | 5 |
| 4.4 | Sequence Diagram | 6 |
| 4.5 | Activity Diagram for user | 7 |
| 4.6 | State Diagram for user | 8 |
| 4.7 | Package Diagram | 9 |
| 4.8 | Component Diagram | 9 |
| 4.9 | Deployment Diagram | 10 |

**10.Table List**

|  |  |  |
| --- | --- | --- |
| Table number | Name of Table | Page Number |
| 4.1 | Event table | 4 |
| 4.10 | Database design | 11 |
| 4.10.3 | Database design for user record | 11 |
| 4.10.4 | Database design for user’s high score | 12 |