

# 2D Semester Project Game



Session: 2019 Section \_B\_\_

**Submitted by:**

Your Name                      Vaniza Riaz

Your Registration No    2019-CS-105

**Submitted To:**

**Mr. Laeeq Khan Niazi**

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

## **Game Environment:**

Catapult Quest 2D game. This is an android based game.

## **Characters:**

It consists of the following characters:

- Catapult
- Ball
- Monsters
- Hurdles which consist of crates

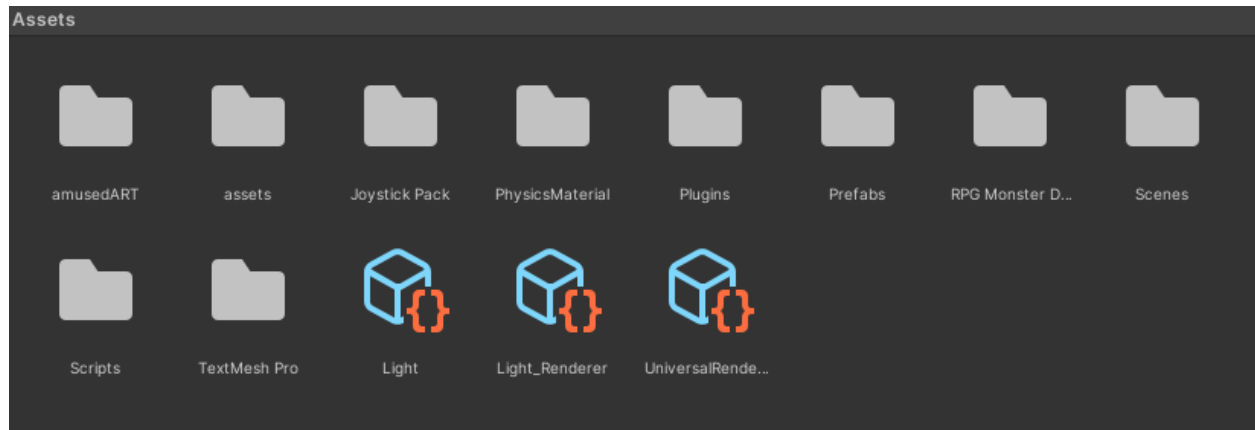
## **Levels:**

It consists of four levels

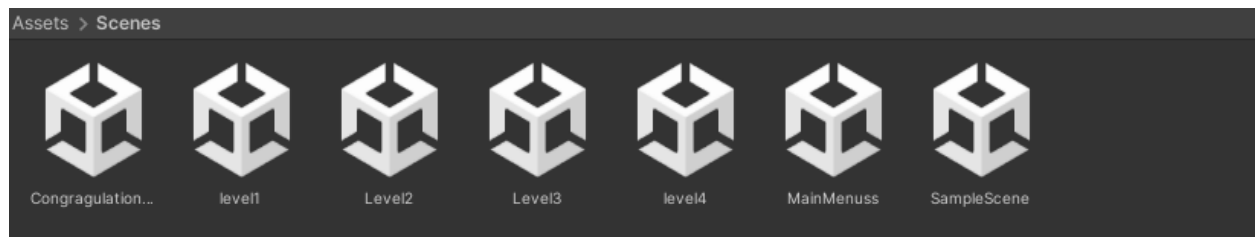
## **GameStory:**

Catapult Quest is a game in which the player drags a Ball with the help of a catapult to different buildings to collect as many monsters as you can. The next level will be loaded when the Player collects all monsters in that level. And the end of playing all levels, the player will get a congratulation screen.

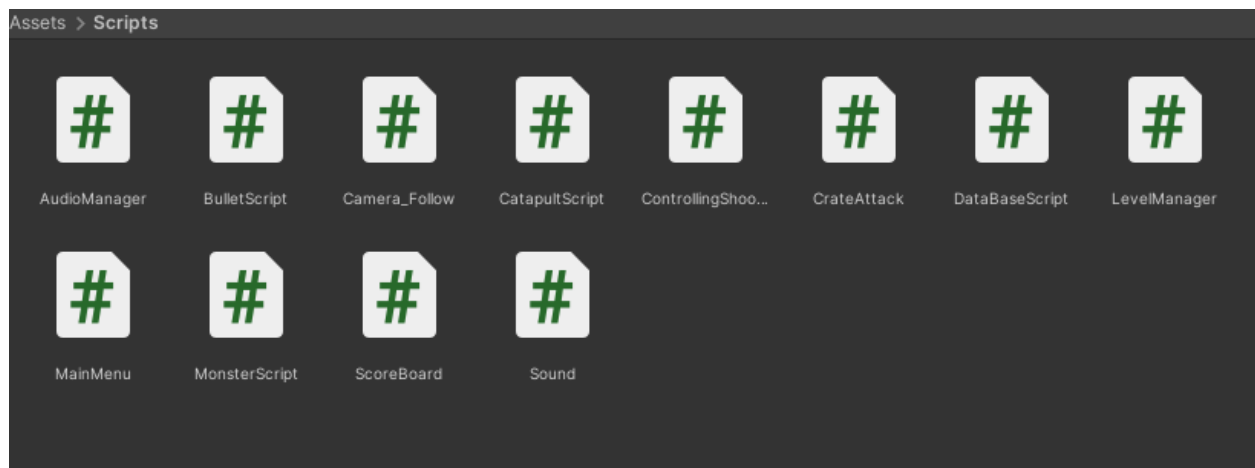
## Assets:



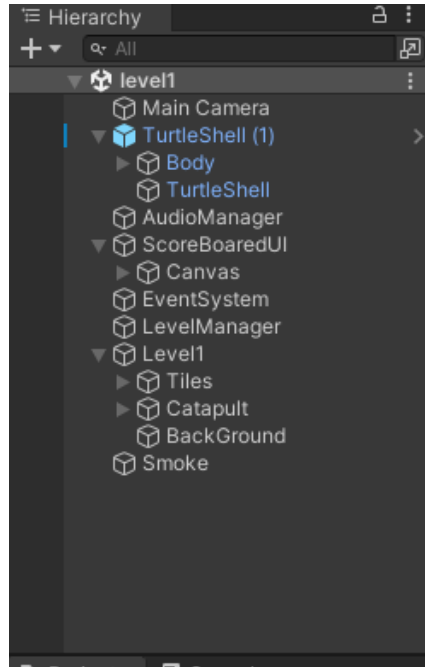
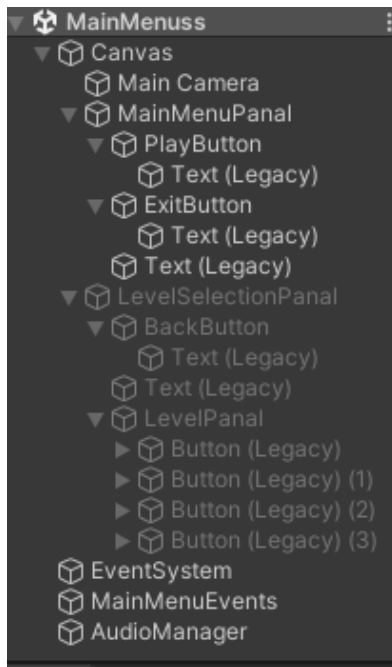
## Scenes:



## Scripts:



## Hierarchy:

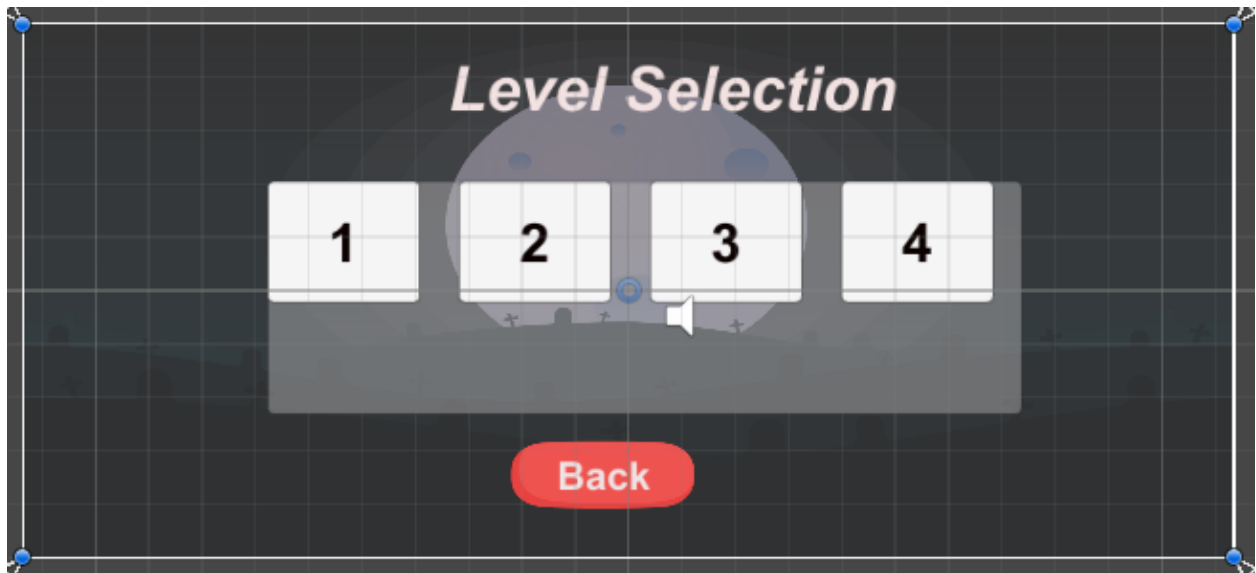


## Scenes:

Main Menu:



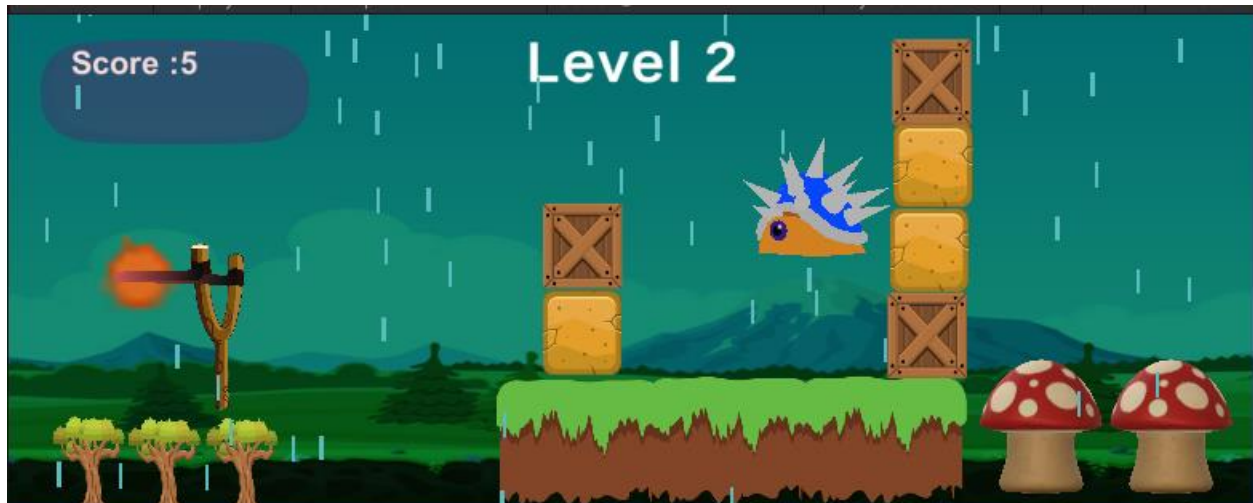
LevelSelector:



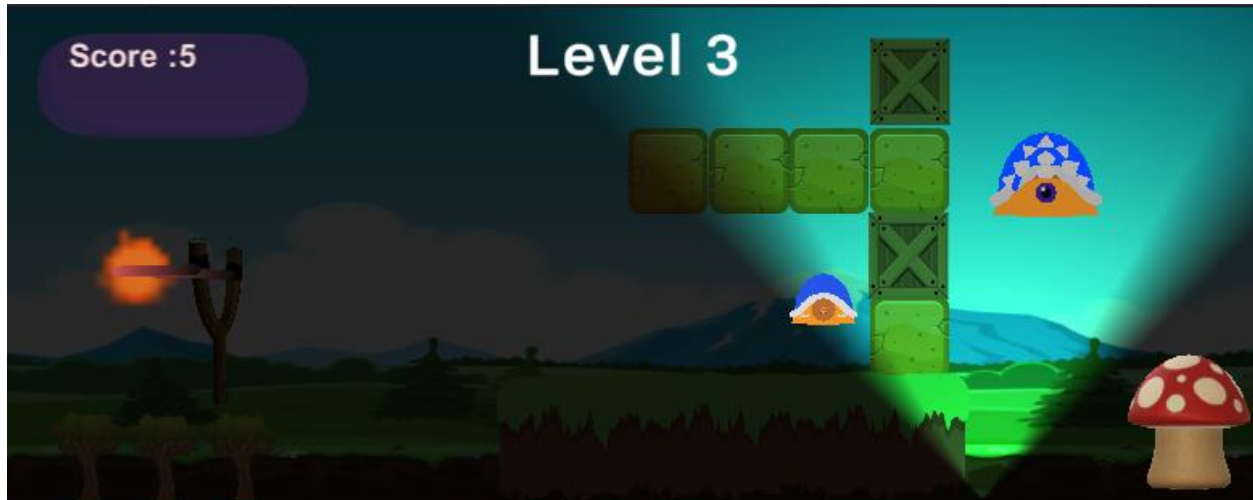
Level 1: With the smoke effect



Level 2: With the rain effect



Level 3: With the light effect



Level 4: With more complexity to destroy monsters to win the game



Congratulation screen:



# Scripts:

## MainMenu:

```
1  using UnityEngine.SceneManagement;
2  using UnityEngine;
3
4  @ Unity Script (1 asset reference) | 0 references
5  public class MainMenu : MonoBehaviour
6  {
7      0 references
8      public void QuitGame()
9      {
10         Application.Quit();
11     }
12
13     0 references
14     public void LoadLevel(int levelIndex)
15     {
16         SceneManager.LoadScene(levelIndex);
17     }
18 }
```

## CatapultScript:

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  [System.Serializable]
5  @ Unity Script (5 asset references) | 0 references
6  public class CatapultScript : MonoBehaviour
7  {
8      public LineRenderer[] lineRenderers;
9      public Transform[] stripPositions;
10     public Transform center;
11     public Transform idlePosition;
12     public Vector3 currentPosition;
13     public float maxLength;
14     public float bottomBoundary;
15     bool isMouseDown;
16     public GameObject birdPrefab;
17     public float birdPositionOffset;
18     Rigidbody2D Vanizarigid;
19     Collider2D birdCollider;
20     public float force;
21     public AudioManager audioManager;
22     @ Unity Message | 0 references
23     public void Awake()
24     {
25         audioManager = FindObjectOfType<AudioManager>();
26     }
27
28     @ Unity Message | 0 references
29     void Start()
30     {
31         lineRenderers[0].positionCount = 2;
32     }
33 }
```



```

1 void Start()
2 {
3     lineRenderers[0].positionCount = 2;
4     lineRenderers[1].positionCount = 2;
5     lineRenderers[0].SetPosition(0, stripPositions[0].position);
6     lineRenderers[1].SetPosition(0, stripPositions[1].position);
7     CreateBird();
8 }
9
10 1 reference
11 void CreateBird()
12 {
13     Vanizarigid = Instantiate(birdPrefab).GetComponent<Rigidbody2D>();
14     birdCollider = Vanizarigid.GetComponent<Collider2D>();
15     birdCollider.enabled = false;
16     Vanizarigid.isKinematic = true;
17     ResetStrips();
18 }
19
20 @ Unity Message | 0 references
21 void Update()
22 {
23     if (isMouseDown)
24     {
25         Vector3 mousePosition = Input.mousePosition;
26         mousePosition.z = 10;
27
28         currentPosition = Camera.main.ScreenToWorldPoint(mousePosition);
29         currentPosition = center.position + Vector3.ClampMagnitude(currentPosition
30             - center.position, maxLength);
31
32         currentPosition = ClampBoundary(currentPosition);
33
34         SetStrips(currentPosition);
35
36         SetStrips(currentPosition);
37
38         if (birdCollider)
39         {
40             birdCollider.enabled = true;
41         }
42         else
43         {
44             ResetStrips();
45         }
46     }
47 }
48
49 @ Unity Message | 0 references
50 private void OnMouseDown()
51 {
52     isMouseDown = true;
53     AudioManager.playMySound("Drag");
54 }
55
56 @ Unity Message | 0 references
57 private void OnMouseUp()
58 {
59     isMouseDown = false;
60     Shoot();
61     currentPosition = idlePosition.position;
62     AudioManager.playMySound("Fly");
63 }

```

```

        audioManager.playMySound("Fly");
    }

    1 reference
    void Shoot()
    {
        Vanizarigid.isKinematic = false;
        Vector3 birdForce = (currentPosition - center.position) * force * -1;
        Vanizarigid.velocity = birdForce;

        // bird.GetComponent<Ball>().Release();

        Vanizarigid = null;
        birdCollider = null;
        Invoke("CreateBird", 2);
    }

    2 references
    void ResetStrips()
    {
        currentPosition = idlePosition.position;
        SetStrips(currentPosition);
    }

    2 references
    void SetStrips(Vector3 position)
    {
        lineRenderers[0].SetPosition(1, position);
        lineRenderers[1].SetPosition(1, position);

        if (Vanizarigid)
        {
            Vector3 dir = position - center.position;
            Vanizarigid.transform.position = position + dir.normalized * birdPositionOffset;
            Vanizarigid.transform.right = -dir.normalized;
        }
    }

```

```

        Vanizarigid.transform.right = -dir.normalized;
    }

    1 reference
    Vector3 ClampBoundary(Vector3 vector)
    {
        vector.y = Mathf.Clamp(vector.y, bottomBoundary, 1000);
        return vector;
    }

```

## ControllingShoot:

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

@ Unity Script (13 asset references) | 0 references
public class ControllingShoots : MonoBehaviour
{
    public AudioManager audioManager;
    static int count;
    public int obj = -1;

    @ Unity Message | 0 references
    public void Awake()
    {
        audioManager = FindObjectOfType<AudioManager>();
        audioManager.playMySound("Start");
    }

    @ Unity Message | 0 references
    void Start()
    {
        obj = GameObject.FindGameObjectsWithTag("Monsters").Length;
        StartCoroutine(WaitAndDestroy());
    }

    // Update is called once per frame
    @ Unity Message | 0 references
    void Update()
    {
        if (count == obj)
        {
            count = 0;
        }
    }

    if (count == obj)
    {
        count = 0;
        ScoreBoard.GetInstance().score = 0;
        Debug.Log("created");
        LevelManager.Instance.LoadNextLevels();
    }
}

1 reference
IEnumerator WaitAndDestroy()
{
    yield return new WaitForSeconds(5);
    Destroy(gameObject);
}

@ Unity Message | 0 references
public void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag == "Ball")
    {
        Destroy(gameObject);
        DataBaseScript.GetInstance().InsertScore(5);
        DataBaseScript.GetInstance().UpdateScore(0, 5);
        audioManager.playMySound("Hit");
        ScoreBoard.GetInstance().AddScore(5);
        DataBaseScript.GetInstance().GetScore();
    }
}

// ...
DataBaseScript.GetInstance().GetScore();
}

if (collision.gameObject.tag == "crate")
{
    audioManager.playMySound("CrateHit");
}
```

## MonstersScript:

```
using UnityEngine.SceneManagement;
// Unity Script (3 asset references) | 0 references
public class MonsterScript : MonoBehaviour
{
    static int count;
    public int obj = -1;
    public SpriteRenderer spriterenderer;
    private int nextSceneToLoad;
    // Unity Message | 0 references
    public void Awake()
    {
        spriterenderer = GetComponent<SpriteRenderer>();
    }
    // Unity Message | 0 references
    private void Start()
    {
        obj = GameObject.FindGameObjectsWithTag("Monsters").Length;
        nextSceneToLoad = SceneManager.GetActiveScene().buildIndex + 1;
    }
    // Unity Message | 0 references
    public void Update()
    {
        if (count == obj)
        {
            count = 0;
            ScoreBoard.GetInstance().score = 0;
            SceneManager.LoadScene(nextSceneToLoad);
        }
    }
}
```

## LevelManagerScript:

```
using UnityEngine;
// Unity Script (4 asset references) | 2 references
public class LevelManager : MonoBehaviour
{
    public GameObject[] Levels;
    private int CurrentLevel;
    public static LevelManager instance;
    // Unity Message | 0 references
    public void Awake()
    {
        if (instance == null)
        {
            instance = this;
        }
        else
        {
            //Destroy(GameObject);
        }
        LoadNextLevels();
    }

    1 reference
    public void HideAllLevels()
    {
        foreach (GameObject level in Levels)
        {
            level.SetActive(false);
        }
    }

    2 references
    public void LoadNextLevels()
    {
        if (CurrentLevel < Levels.Length)
        {
            HideAllLevels();
            Levels[CurrentLevel].SetActive(true);
            CurrentLevel++;
        }
    }
}
```

## CrateAttackScript:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [Unity Script (34 asset references) | 0 references]
6 public class CrateAttack : MonoBehaviour
7 {
8     [Unity Message | 0 references]
9     public void OnCollisionEnter2D(Collision2D collision)
10    {
11        if (collision.gameObject.tag == "Ball")
12        {
13            Destroy(gameObject);
14        }
15    }
16 }
```

## ScoreBoardScript:

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

7 references
public class ScoreBoard
{
    private int kills;
    public int score;
    private int attempts;
    //the private static single instance to access the class
    private static ScoreBoard instance = new ScoreBoard();
    private Text ScoreText;
    //the private constructor
    1 reference
    public ScoreBoard()
    {
        ScoreText = GameObject.FindWithTag("Score").GetComponent<Text>();
    }

    // the public class to access instance
    3 references
    public static ScoreBoard getInstance()
    {
        return instance;
    }

    1 reference
    No issues found
```

```

1 reference
1 public void AddScore(int score)
{
    this.score += score;
    UpdateUI();
    Debug.Log("score" + this.score);
}

```

```

0 references
1 public int getScore()
{
    return this.score;
}

```

```

0 references
1 public void addkill(int kills)
{
    this.kills++;
    Debug.Log("kills" + this.kills);
}

```

```

0 references
1 public int getkill()
{
    return this.kills;
}

```

```

0 references
1 public void addattempt(int attempts)
{

```

```

0 references
1 public void addattempt(int attempts)
{
    this.attempts++;
    Debug.Log("attempts" + this.attempts);
}

```

```

0 references
1 public int getattempts()
{
    return this.attempts;
}

```

```

1 reference
1 public void UpdateUI()
{
    if (ScoreText)
    {
        ScoreText.text = "Score :" + this.score;
    }
    else
    {
        ScoreText = GameObject.FindWithTag("Score").GetComponent<Text>();
    }
}
}

```

# DataBaseScript:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;

@ Unity Script (2 asset references) | 7 references
public class DataBaseScript:MonoBehaviour
{
    string dbfile = "URI=file:SqliteTest.db";
    public static DataBaseScript instance;
    1 reference
    private DataBaseScript()
    {
    }
    @ Unity Message | 0 references
    void Start()
    {
        CreateScoreTable();
    }
    3 references
    public static DataBaseScript getInstance()
    {
        if (instance == null)
        {
            instance = new DataBaseScript();
        }
        return instance;
    }


    private void CreateScoreTable()
    {
        using (var connection = new SqliteConnection(dbfile))
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                command.CommandText =
                    "create table if not exists PlayerScore(" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                    "score varchar(30))";
                command.ExecuteNonQuery();
                Debug.Log("New table created");
            }
            connection.Close();
        }
    }

    1 reference
    public void InsertScore(int score)
    {
        using (var connection = new SqliteConnection(dbfile))
        {
            connection.Open();
            string query = "Insert into PlayerScore (score) values (@score)";
            using (var command = connection.CreateCommand())
            {
                command.CommandText = query;
                command.Parameters.AddWithValue("@score", score);
                command.ExecuteNonQuery();
                Debug.Log("Data inserted ");
            }
        }
    }
}
```

1 reference

```
public void GetScore()
{
    using (var connection = new SqlConnection(dbfile))
    {
        connection.Open();
        string query = "select score from PlayerScore where id = 1";
        using (var command = connection.CreateCommand())
        {
            command.CommandText = query;
            SqlDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                Debug.Log(reader["score"]);
            }
        }
    }
}

public void UpdateScore(int id, int score)
{
    using (var connection = new SqlConnection(dbfile))
    {
        connection.Open();
        string query = "update PlayerScore set score=@score where id = @id";
        using (var command = connection.CreateCommand())
        {
            command.CommandText = query;
            command.Parameters.AddWithValue("@id", id);
            command.Parameters.AddWithValue("@score", score);
            command.ExecuteNonQuery();
            Debug.Log("Data Updated");
        }
    }
}
```

 class System.String

10 references

```
public void DeleteScore(int id)
{
    using (var connection = new SqlConnection(dbfile))
    {
        connection.Open();
        string query = "delete from PlayerScore where id = @id";
        using (var command = connection.CreateCommand())
        {
            command.CommandText = query;
            command.Parameters.AddWithValue("@id", id);
            command.ExecuteNonQuery();
            Debug.Log("Data Deleted");
        }
    }
}
```

BulletScript:



```

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

[Unity Script | 0 references]
public class BulletScript : MonoBehaviour
{
    [Unity Message | 0 references]
    void Start()
    {
        StartCoroutine(WaitAndDestroy());
    }

    // Update is called once per frame
    [Unity Message | 0 references]
    void Update()
    {
    }

    1 reference
    IEnumerator WaitAndDestroy()
    {
        yield return new WaitForSeconds(5);
        Destroy(gameObject);
    }

    [Unity Message | 0 references]
    public void OnCollisionEnter2D(Collision2D collision)
    {
        Destroy(gameObject);
    }
}

```

AudioManager:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

@ Unity Script (5 asset references) | 4 references
public class AudioManager : MonoBehaviour
{
    public Sound[] sounds;
    @ Unity Message | 0 references
    public void Awake()
    {
        foreach (var sound in sounds)
        {
            sound.audioSource = gameObject.AddComponent<AudioSource>();
            sound.audioSource.clip = sound.clip;
            sound.audioSource.volume = sound.volume;
            sound.audioSource.pitch = sound.pitch;
        }
    }

    5 references
    public void playMySound(string name)
    {
        foreach (var sound in sounds)
        {
            if (sound.name == name)
            {
                sound.audioSource.Play();
            }
        }
    }
}

```

SoundScript:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

[System.Serializable]
1 reference
public class Sound
{
    public string name;
    [Range(1f, 3f)]
    public float pitch;
    [Range(0f, 1f)]
    public float volume;
    public AudioClip clip;
    public bool loop;

    [HideInInspector]
    public AudioSource audioSource;
}

```

SQLStudio:

Databases

Filter by name

SqliteTest (SQLite 3)

Tables (1)

PlayerScore

Views

Structure

Data

Constraints

Indexes

Triggers

DDL

SqliteTest

Table name: PlayerScore

☐ WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	id	INTEGER								NULL
2	score	varchar ...								NULL

Type

Name

Details

Structure

Data

Constraints

Grid view

Form view

↺

+

↻

✓

✕

⬅

	id	score
1	1	10
2	2	10
3	3	10
4	4	10
5	5	10
6	6	5
7	7	5
8	8	5
9	9	5
10	10	5
11	11	5
12	12	5
13	13	5
14	14	5
15	15	5
16	16	5
17	17	5
18	18	5
19	19	5
20	20	5