

# 2D MULTIPLAYER GAME DEVELOPMENT USING UNITY

Kumaresan G<sup>1</sup>, Sharath A<sup>2</sup>, Sheshanth B P<sup>3</sup>, Somesh A<sup>4</sup>

<sup>1</sup> Assistant Professor, Department of Computer Science and Engineering, SRM Valliammai Engineering College, Tamil Nadu, India

<sup>2</sup> UG Student, Department of Computer Science and Engineering, SRM Valliammai Engineering College, Tamil Nadu, India

<sup>3</sup> UG Student, Department of Computer Science and Engineering, SRM Valliammai Engineering College, Tamil Nadu, India

<sup>4</sup> UG Student, Department of Computer Science and Engineering, SRM Valliammai Engineering College, Tamil Nadu, India

## ABSTRACT

Video games are common these days. The games have been developed for various purposes including entertainment, informational purposes, marketing, and advertising purposes. Game developers are continually developing multi-platform games that can be played on computers, mobile devices like iPads and smartphones and other video game devices. Today, mobile games consume a huge market share within the games industry. Considering that nearly two billion mobile devices are running a Unity-made game, it is probable that quite a large majority of these new companies will be using Unity to develop mobile games. Mobile games are becoming more and more popular and well-received by different types of users around the world. A lot of frameworks that allow developers to create the games in a faster and easier way. The Unity 3D engine is employed by an outsized majority of developers to make games. It owns a forty-five percent market share and is considered one of the biggest development tools today. This project involves the approach of creating a 2D mobile game using Unity. Using 2D development tools present in Unity, a multiplayer game is to be created. The presence of colliders, triggers, and use of sprites makes Unity the best choice in game design. The main idea of the game is to make the players work as a team and explore the map with different players playing in different smartphones. This involves creating game levels using tilemaps and sprite sheets and how to implement physics like jumping, running, shooting, etc., The game is built in both Android and IOS platforms. This game can be played in different mobile phones with different ranges of hardware and operating systems.

**Keywords:** - Unity, Frameworks, colliders, and platforms

---

## 1. INTRODUCTION

Mobile games share an important part in today's gaming world. Different mobile games have been in the trend for the past two years. Their success depends on the different ideas and the constant updates they provide. Today many developers choose the mobile gaming industry as their entry-level game designing profession. Our game is a multiplayer 2d game played with friends via the Internet. First, a single host creates a Room to which other players can join. They are provided with basic movements and shooting mechanism to interact with the environment. The Unity Engine is used in the game design and the PUN engine is used for the multiplayer connectivity. Since 2d multiplayer is a less explored genre in mobile gaming and with only a handful of games, New games with interesting concepts are always expected to shine bigger and better in this genre.

## 2. RELATED RESEARCH

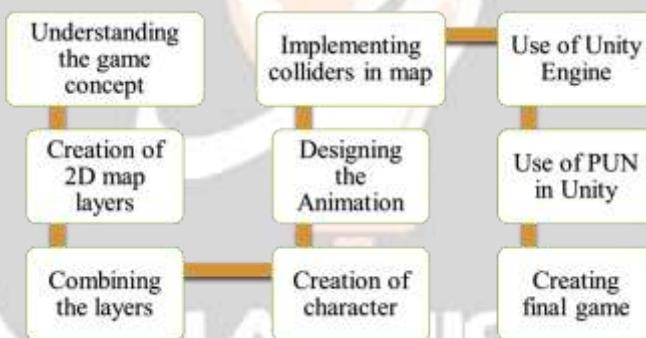
In android and IOS platforms, multiplayer games are a trend now. Games like Mini Militia – Doodle Army 2 is celebrated and enjoyed by different players around the world and PUBG Mobile is still the undisputed multiplayer in the Mobile game industry. The aspect of touch controls may seem a bit off for new players. But with minimal controls and precise buttons, touch controls provide one of the best gaming experiences in mobile phones.

The game Mini Militia involves different players playing in different mobile devices and includes a wide variety of maps to play. The players can customize their avatar to their needs. It involves an analog touch control system for movement as well as shooting.

Many games in today's mobile gaming industry are developed using Unity. Unity is the best game design and development engine where developers with different ideas put their ideas into use. With a user-friendly interface to work with, Unity is the best option for 2D game developers to create and showcase their games.

## 3. PROPOSED METHOD

Our proposed method is to develop a 2D multiplayer game using Unity 3D software. Unity 3D is the best choice among the other tools for creating a game. The concept of the game is a multiplayer shooting game. The map for the game is designed using Adobe Photoshop and imported into the Unity 3D. A character is designed for the game and various image sequences are created to form animation for basic movements like running, jumping, idle, etc., Unity has a built-in animator with which the character is animated. The backend code is written in C# using Visual studio. In this game, a portal concept is introduced using which player can teleport from one place of the map to another. Finally, the game is converted into the multiplayer game using Photo unity Network (PUN). PUN is a plug-in developed by exit games for excellent multiplayer capabilities



**Fig -1:** Basic Structure



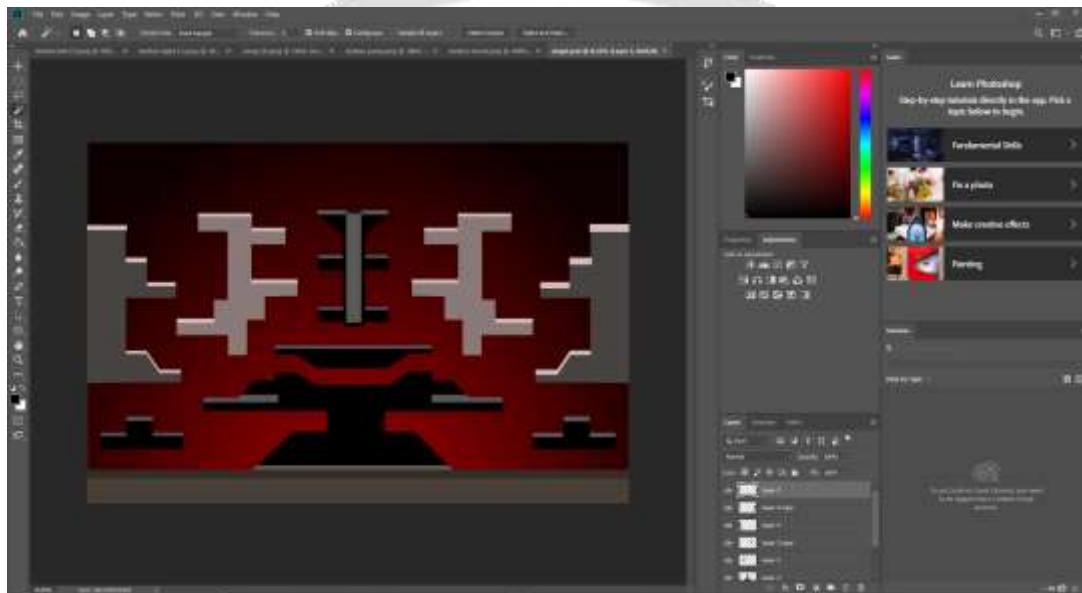
**Fig -2:** Software Structure

### 3.1 Creating Map and Character using Adobe Photoshop

Photoshop is an important tool employed by artists who make games. Mobile games have the foremost potential of all three of the sport types now. It takes many various sorts of artists to form a game, and that they all have one thing in common: They use Photoshop regularly.

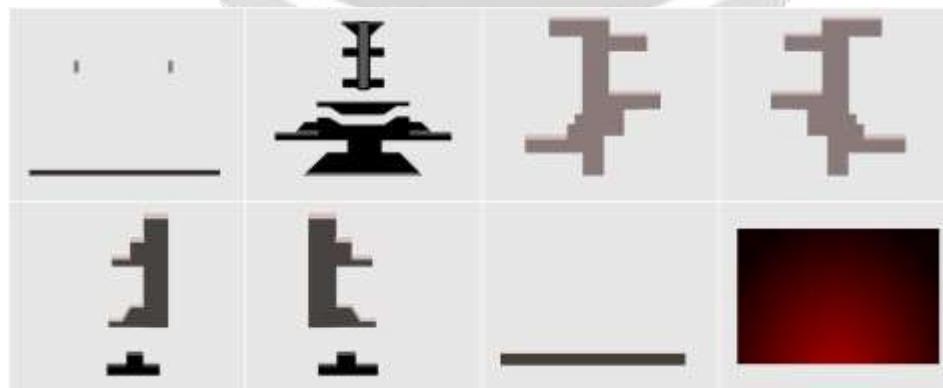
Photoshop enables designers to create different pictures and environmental elements. These elements can be designed and developed by the game designers based on the game they are making.

The use of photoshop in game design also involves the creation of character and player elements. The designers use photoshop to create layers of environments which are then implemented in Unity. Photoshop is mainly used to design game elements for 2D games. The 3d games involve different software for the creation of game elements like environments objects and characters. The 3D design software includes 3DsMax, Maya, etc., Whereas the photoshop is mainly for the 2D game design and development. The PNG format files of the game elements are then exported and implemented in the Unity Game engine.



**Fig -3:** Map in Photoshop

This map is then split into different layers and saved as png. The png files of the map are then added to the unity for the game design and development.

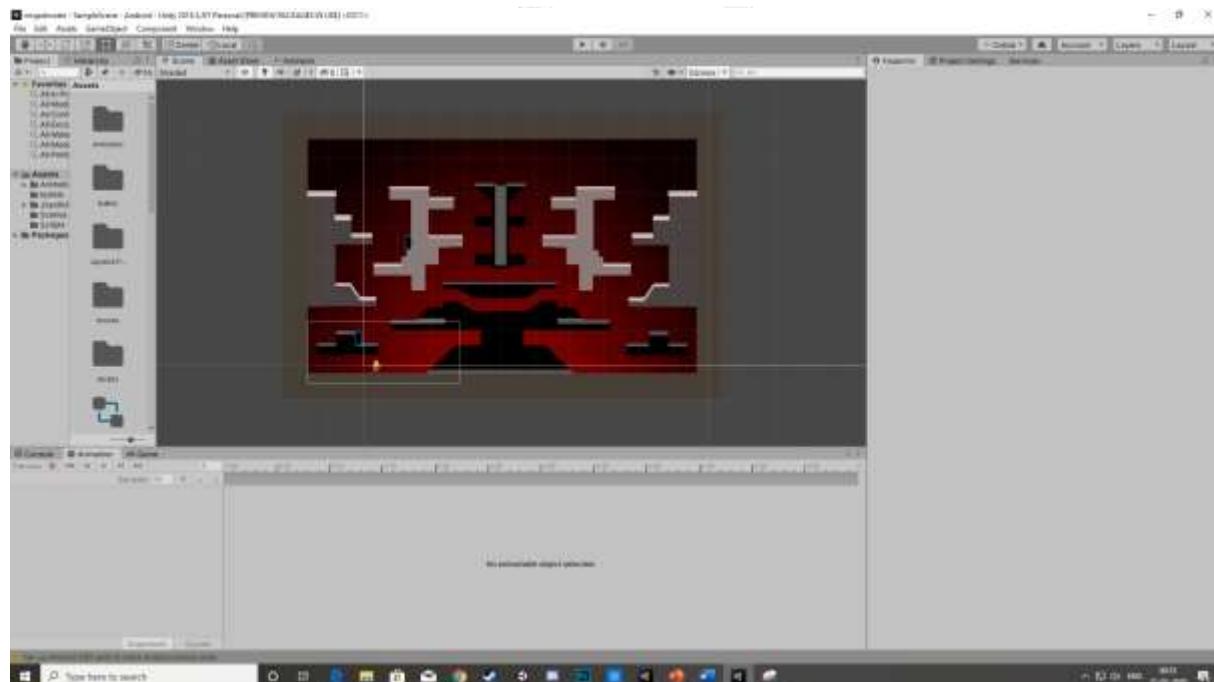


**Fig -4:** Layers of Map

### 3.2 Use of Unity Engine

Today developers use game engines to construct games for consoles, mobile devices, and private computers. The key functionality typically provided by a game engine includes a rendering engine for 2D or 3D graphics, a physics engine for collision detection, sound, scripting, animation, AI, networking, streaming, memory management, threading, localization support, scene graph, and should include video support for cinematics.

Implementers often economize on the method of game development by reusing/adapting, in large part, an equivalent game engine to supply different games or to assist in porting games to multiple platforms. The Unity game engine is the most popular game engine in designing video games for mobile devices. The unity engine also involves different build supports for different Operating systems like Android, IOS, etc.,



**Fig -5:** Unity Engine

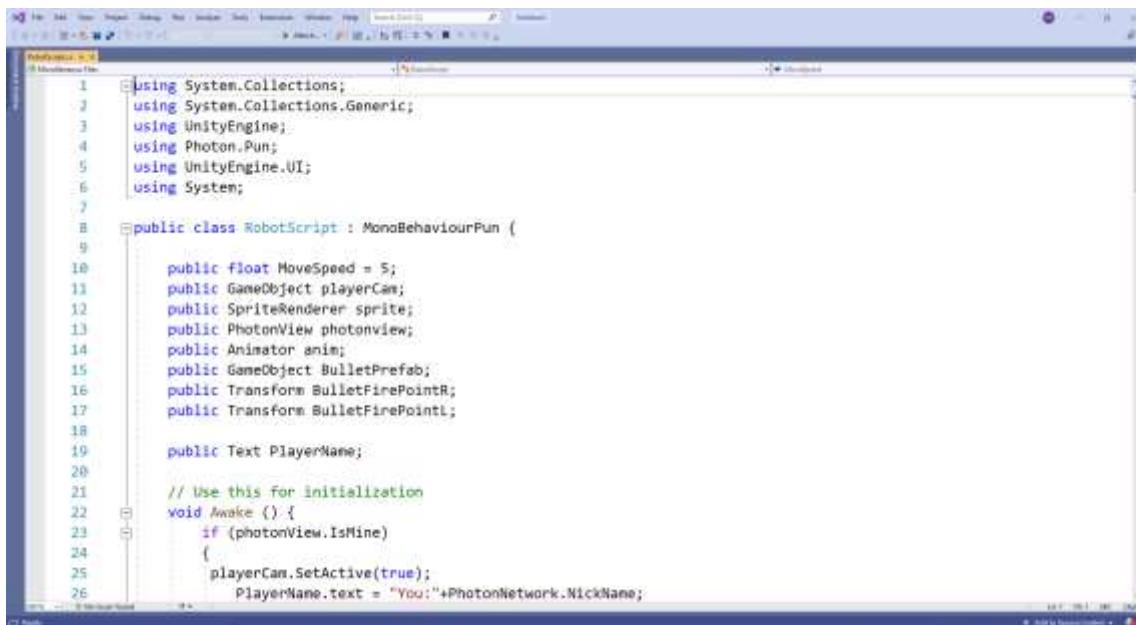
### 3.3 Coding in Visual studio

Microsoft Visual Studio is a development environment by Microsoft. It is used to develop computer programs, websites, web applications, web services, and mobile apps. 36 different programming languages are supported by Visual Studio which includes C#, C++, etc., and allows the code editor and debugger to support nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, JavaScript, TypeScript, XML, HTML, and CSS. used in the Unity Game engine for coding purposes. Here in Unity Game Engine, the main programming language used is C#. It involves the use of inbuilt packages and conditionals statements which results in various scripts in game development which includes scripts like character controllers, camera tracking, player movement, etc., These are the Three main software involved in the development of the 2D game. In our game, the basic C# scripts include scripts for player movement, bullet movement, network connectivity, etc., These scripts are created and edited in Visual Studio.

### 3.4 Using PUN

PUN is used in Unity to develop Multiplayer Online games in an effective way. PUN is useful and capable of exporting the multiplayer game to different platforms supported by Unity which includes platforms like Android, IOS, Consoles, etc., It is also a cross-platform multiplayer service that enables multiplayer games to be played from different devices with different platforms.

In this game, the Photon is introduced to the Unity. All the changes the user or the player makes in his/her game environment is reflected in the games of the other players connected in the network. This enables the multiplayer capabilities of the game. Using the PUN provides our game with high Scalability options, were a large number of players can create and play matches together. Its pricing depends on the CCU (Concurrently Connected Users) of the game. PUN is highly customizable and can be flexible enough for different kinds of Multiplayer games.



```

RobotScript.cs
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Photon.Pun;
5  using UnityEngine.UI;
6  using System;
7
8  public class RobotScript : MonoBehaviourPun {
9
10    public float MoveSpeed = 5;
11    public GameObject playerCan;
12    public SpriteRenderer sprite;
13    public PhotonView photonview;
14    public Animator anim;
15    public GameObject BulletPrefab;
16    public Transform BulletFirePointR;
17    public Transform BulletFirePointL;
18
19    public Text PlayerName;
20
21    // Use this for initialization
22    void Awake () {
23        if (photonView.IsMine)
24        {
25            playerCan.SetActive(true);
26            PlayerName.text = "You:" + PhotonNetwork.NickName;
27        }
28    }
29
30    void Update () {
31        if (Input.GetKey(KeyCode.W))
32        {
33            transform.Translate(0, 0, MoveSpeed * Time.deltaTime);
34        }
35        if (Input.GetKey(KeyCode.S))
36        {
37            transform.Translate(0, 0, -MoveSpeed * Time.deltaTime);
38        }
39        if (Input.GetKey(KeyCode.A))
40        {
41            transform.Rotate(0, 0, -10);
42        }
43        if (Input.GetKey(KeyCode.D))
44        {
45            transform.Rotate(0, 0, 10);
46        }
47    }
48
49    void OnPhotonSerializeView (byte id, PhotonStream stream, PhotonMessageInfo info) {
50        if (id == 0)
51        {
52            stream.SendNext(photonview.IsMine);
53        }
54    }
55
56    void Shoot () {
57        if (Input.GetKeyDown(KeyCode.Space))
58        {
59            GameObject bullet = Instantiate(BulletPrefab, BulletFirePointR.position, BulletFirePointR.rotation);
60            bullet.GetComponent().AddForce(BulletFirePointR.forward * 100);
61        }
62    }
63
64    void OnTriggerEnter(Collider other)
65    {
66        if (other.gameObject.tag == "Player")
67        {
68            if (other.gameObject != playerCan)
69            {
70                other.gameObject.GetComponent<PlayerScript>().TakeDamage(10);
71            }
72        }
73    }
74
75    void TakeDamage (int damage)
76    {
77        if (photonView.IsMine)
78        {
79            health -= damage;
80            if (health <= 0)
81            {
82                photonView.Kill();
83            }
84        }
85    }
86
87    void OnPhotonSerializeView (byte id, PhotonStream stream, PhotonMessageInfo info) {
88        if (id == 1)
89        {
90            stream.SendNext(health);
91        }
92    }
93
94    void OnPhotonSerializeView (byte id, PhotonStream stream, PhotonMessageInfo info) {
95        if (id == 2)
96        {
97            stream.SendNext(photonview.IsMine);
98        }
99    }
100}

```

**Fig -6:** Inclusion of PUN code in Visual Studio

## 4. CONCLUSIONS

The 2D Multiplayer game is developed which can be played with different players and friends. The main concept is to create a room and to enable other players to join the room. Once the players join the room, the matchmaking can be started. This game is for both Android and IOS. Since we are using PUN for Multiplayer connectivity, this game is cross-platform too. This enables players from IOS and Android to join the same match.

In the future, the Mobile game industry can reach a greater height with games like these to support the Multiplayer gaming genre with Consistent Player count all around the world.

## 5. REFERENCES

- [1]. LiJiyuan and HuWenfeng, “Development of Puzzle Game for IOS Platform Based On Unity3D”, 2015
- [2]. “Learning 2D Game Development with Unity” - Matthew Johnson and James A. Henley, 2014
- [3]. Sergii Shepelenko, University of Tartu, “Creating a 2D Platform Game”
- [4]. D. Polančec, I. Mekterović, “Developing MOBA games using the Unity game engine”, IEEE, 2017

[5]. “2D Tilemap Collision” - <http://jonathanwhiting.com/tutorial/collision/>

[6]. <https://www.youtube.com/user/Unity3D> – by Unity

[7]. “2D Tilemap Collision” - <http://jonathanwhiting.com/tutorial/collision/>

[8]. <https://gamedevacademy.org/unity-photon-tutorial/> - Gamedevacademy

