International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 07 | July -2020 ISSN: 2582-3930

Game Designing using C Sharp

Binoy Gogoi[1], Pranav Kumar[2]

[1]Department of Information Technology, Kaziranga University, Jorhat, Assam-785006, India,
[2]Department of Computer Science and Engineering, Kaziranga University, Jorhat, Assam-785006, India,

---

Abstract -

Countless hyper casual games are being developed and published over the world every year. Likewise, we will be developing an Endless Hyper Casual game of my own for pc's having Windows OS. Hyper Casual games are not only instantly playable but also infinitely re-playable, making them highly addictive and engaging. In the title the word 'Endless' refers that the extent to which the scores can be made is not limited or is unlimited. In this Endless Hyper Casual game, the player controls a character. The player has to successfully get-pass the obstacles while the score simultaneously gains up. The game will end if the player is unable to successfully overcome the given obstacles. We will also add a few power-up's or special abilities in the game to make the game more challenging and interesting to play. The game will include various assets and will be of 2-dimensional (2D) design. We will be developing the game in a cross-platform game engine called 'Unity', developed by Unity Technologies, with the help of the programming language "C sharp".

1.INTRODUCTION

This project covers the basis procedure of creating a simple 2D game with the help of Unity game engine and C# programming language. The participant in this project is a game enthusiast and is interested in playing games. The need for making this project is to show every person interested in games, how to create their own 2D game simply with the help of the free software's available in the internet.

The contents of this project can be divided into three sections. Firstly, Unity game engine is used for this game project. The use of game engine is discussed providing some knowledge about what is a game engine, its purpose and the supported platforms. The features that it provides, the physics engine, audio engine and the user friendly interface. Unity included 2D development tools in the update4.3 (Goldstone, 2013). After that comes the source code programming language that is C Sharp or C#. It provide the description on why C# is used for game development. The different types of C# functions that can be used with the help of C# codes. Also a few of the advantages, disadvantages and applications of C#.

Lastly, it describes about the game mechanics i.e. a complete brief guide on how the game is developed in Unity. It show the assets and game objects that are being used. The attributes of the game components inside the game objects and the C# scripts required in this game along with their respective C# codes. The end result of this project is a 2D Game that can be enjoyed playing endlessly.

2. GAME MECHANICS

Game Play

Firstly, our game is a hyper casual 2D game. It is a game simple to understand as well as to play. The playing character of the game a square shaped smiley object. Here the challenges are that the player has to react and respond at accurate time. The platform will spawn from the bottom of the screen .The player controlling the character has to move sideways and fall on a platform to avoid getting crushed by the spike at the top of the screen.

Assets in the Game: Here the following are the Assets in our unity game paper.

a) Animations: Combine 2D and 3D Dope sheet and curve animation for sprites. Animate any property in your game. Automatically create animated sprites from sprite sheets.

- **Idle:** It has the animation of the standard platform in this game.
- **Break Animations:** It has the animations of the cracks to be displayed in the breakable platform.

- **Breakable controller :** Breakable controller is an Animator Controller.

b) **Materials:**
- **BG Materials:** BG Materials is a material created in this unity paper to attach the background of this game.

c) **Prefabs:** Prefab in Unity system allows you to create, configure, and store a GameObject completely with all its property values, components, and child GameObject's as a reusable Asset. This Asset acts as a guide from which you can create new Prefab instances in the Scene. When you want to reuse a GameObject configured in a particular way in multiple places in your Scene, you should convert it to a Prefab. This is better than to copy & paste the GameObject, because the Prefab system helps you to keep all the copies in sync automatically.Here, the prefabs to be present in our Game paper are the total number of platforms used and the Player**.**

d) **Scenes:** Scenes contain the environments and menus of your game. Think of each unique Scene file in the game as a unique level. In each Scene, you place your obstacles, environments and decorations, basically to design and build your game in pieces

- **Sample-Scene:** Here in the Scenes folder, Sample Scene will be our gamming environment in which our game will be made**.**
- **Scripts:** Scripts or C# scripts in unity is basically the codes written in in C# programming language. These gameplay scripts are easy to modify and are updated by non-programmers to tweak the gameplay.
- **BGScroll:**BGScroll is a script in C sharp edited in 'visual studio code'. This script will grant us the feature to scroll the background image upwards.
- **PlatformScript:** Here, PlatformScript is a script in C sharp edited in 'visual studio code'. This script will be attached to all the platforms objects in the hierarchy.
- **PlatformSpawner:** Here, PlatformScript is a script in C sharp edited in 'visual studio code'. This script will allow us to spawn new platforms in the game continuously from random positions as well as spawn random types of platforms.
- **Player Bounds:** Here, PlatformScript is a script in C sharp edited in 'visual studio code'. This script will create the bounds in the scene of our game so that the player would not drift away from the displaying screen of the game.

- **Player Movement:** Here, PlatformScript is a script in C sharp edited in 'visual studio code'. This script will provide us the feature to input the velocity of the player's movement in the game environment.

e) **Sound:**A sound effect is an artificially created or enhanced sound, or sound process used to highlight artistic or other content of Video games. Sound is of outmost importance in a game to make a better user gaming experience and also for the purpose of entertainment, of course. It will also help the game user to feel the game experience in a intrigued gaming atmosphere.

f) **Sprites:**Sprites are 2D Graphic objects. Sprites are basically just standard textures but there are special techniques for combining and managing sprite textures for the convenience and efficiency during development. Unity renders a placeholder Sprite Creator ,a built-in Sprite Renderer, Sprite Editor, and a Sprite Packer.

Components in the Game

- **Transform:** The Transform component specifies the Position, Rotation, and Scale of each and every object in the scene. Every GameObject has a Transform.
- **Camera:** Camera is the devices that capture and display the world to the player. You can make the presentation of your game truly unique by customizing and manipulating cameras. In a scene there can be unlimited number of cameras. They can be placed in any order, at any place on the screen, or particularly only certain parts of the screen.
- **AudioListener:** The Audio Listener is used as a microphone-like device. The input is received from any given Audio Source in the scene and plays sounds through the speakers of the computer. In most applications listeners are attached the to the Main Camera **If** an audio listener is within the boundaries of a Reverb Zone reverberation is applied to all audible sounds in the scene. Also, if Audio Effects can be applied to the listener then it will be implemented d to all audible sounds in the scene**.**
- **Quad (Mesh Filter):** The Mesh Filter takes a mesh from your assets and passes it to the Mesh Renderer for rendering on the screen
- **Mesh Renderer:** The MeshRenderer takes the geometry from the Mesh Filter and renders it at the position defined by the GameObject's Transform component.Materials that the Mesh Renderer is using, the Materials portion in the Mesh Renderer Inspector displays all of them. The meshes imported from 3D modeling s/w can use multiple Materials, and each sub-Mesh uses a single Material from the list.Suppose a Mesh contains more Materials than sub-Meshes, Unity renders the last sub-Mesh with

each of the remaining Materials, one on top of the next. This will let you to set up multi-pass rendering on that sub-Mesh. Nevertheless, this can affect the performance at run time. Fully non transparent Materials overwrite the previous layers, which causes a decrease in performance.

- **Mesh Collider**: The Mesh Asset builds its Collider based on that Mesh taken from the Mesh Collider. It is more accurate for collision detection. The convex marked Mesh Colliders can collide with other Mesh Colliders.
- **BGScroll (Script):** The BGScroll (Script) is given in the refereed marking numbers

Property	Function
Script	BGScroll
Scroll_Speed	0.3

Table no.1- BG scroll

- **Sprite Renderer**
  The Sprite Renderer component provides the Sprite and controls how it visually appears in a Scene. When you create a 2D Sprite, Unity automatically creates a GameObject with the Sprite Renderer component attached to it. You can add the component to an existing GameObject with the help of the Components menu.
- **Box Collider 2D:** The Box Collider is an invisible shape that is used to handle physical collisions for an object. A collider do not need to be precisely the same shape as the object's mesh – a rough approximation is enough for the gameplay
- **Rigidbody 2D:** It places an object under the control of the physics engine. Rigidbody 2D is familiar from the standard Rigidbody component; the differences are that in 2D, objects can only move in the XY plane. It can only rotate on an axis perpendicular to that plane.

Applications of :

Standard platform

- Drag and drop the platform sprite in the Hierarchy table/ scene to create a new game object.
- In the sprite renderer game component set the order in layer to 1.
- Resize the platform to a favorable size.
- Add two new game components: Rigidbody2D & Box collider2D.
- Set the body type to kinematic in Rigidbody2D.
- This will make the object to be affected by gravity. Whereas Dynamic is not affected by gravity.
- Now duplicate the game object platform and rename as a new game object as Breakable platform.

Breakable platform.

- It consists of the same components as of the standard platform.
- Add a new game component, Animator.
- Create an animator controller in Paper Assets and attach it to the animator game component.
- Now in the animation tab, create an animation as idle.
- The animation tab will show the animations that we have created.
- Create another animation as Break.
- Select the break animation sprites, Drag and drop them in the break animation.
- Adjust the speed at which the platform will jump.
- Uncheck the loop time in the break animation.
- Set the idle animation as default.
- Now duplicate the game object platform and rename as a new game object as Breakable platform

Spike Platform

- Drag and drop the spike platform sprite to create a new game object. Rename it as Spike platform
- Set order in layer to 2 in the sprite renderer game object.
- Adjust the size of the spike platform.
- Add Boxcollider2d and set it to trigger.
- Add Rigidbody2D and set body type to kinematic.

Moving platform left

- Create a new game object as Moving platform left

- Select all the 8 moving platform sprites, drag and drop it in the scene, it will automatically prompt a window to create an animation.
- Create the moving animation.
- Set order in layer to 1.
- Adjust the size of the platform.
- Add Boxcollider2D, they are going to be solid game object.
- Add Rigidbody2D and set body type to kinematic.

Moving platform right

- Duplicate the Moving platform left and rename to moving platform right.
- Go to sprite renderer, check the 'x' in flip to flip the animation.
- Now the platform will move to the right direction.

Result and Discussion

When you open the .exe file on your window a configuration window of Cube Fall will appear.In this configuration window there are two tabs "Graphics" and "Input" as well as two buttons in the bottom of the configuration panel "Play" and "Quit".

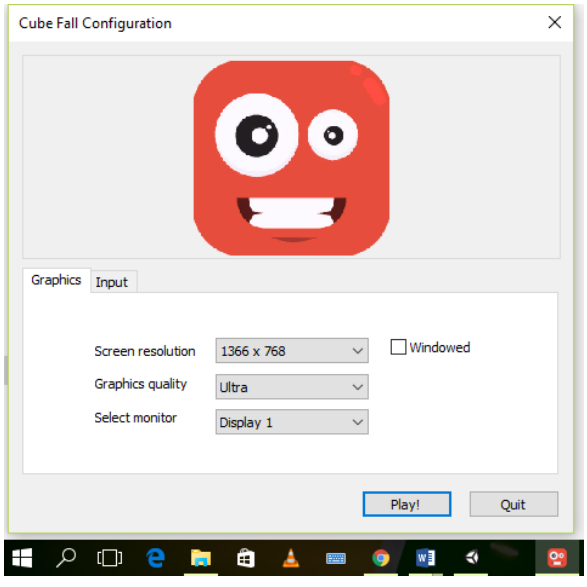

Fig no.1- Configuration window

Immediately as the game starts, the Cube will fall from the top-middle part of the screen on a platform. The Cube will fall on a regular standard platform. The background image will scroll upwards continuously as the game progresses. Inorder to move from another platform lookout for the next incoming platform. Use the controls to move to the left or right side of the platform and land on a platform as it falls.

The Cube/Player falls must land on a platform to progress in the game. Random platforms keep spawning from the bottom of the screen. The platforms are spawned by the platform Spawner at the bottom. They will spawn from different random locations from the bottom of the screen.

The platforms does no harm to the player/cube except the Spike platform. So you should definitely avoid landing on the Spike platform. The spikes at the top are also as deadly as it looks and in under no condition should come into contact with the Cube/player. As you keep falling from one platform to another, if you are not able to land on a platform and fall into the void below, the player will be executed and the game will automatically be restarted from the beginning.
Click on the red colored button located on the top-left side of the screen to exit the game. The game will immediately be dismissed.

3. CONCLUSIONS

A paper in game designing means you need a lot of experience. In this section we summarize our experience gained by paper team during development of the game. Working with the game engine was a completely new experience for us. Normally we are working with different languages and software's. It is a very sensible work and it demands much time and understanding of both Unity and C#.

2D games in my opinion are the precise way of recalling our much fondly childhood digital games before the 3D games that we play today even existed. They are pretty easy to play and equally challenging. Meanwhile providing a great and fun gaming experience. Likewise we hope to provide a positive gaming experience through this game paper.

Although without any doubt, it is totally exciting to make a game. This whole paper was a mesmerizing journey of learning by having fun. We came to understand that there is a whole lot of hardwork and immense effort behind every game we enjoy playing, while building profound respect to the people working in creating these games.

ACKNOWLEDGEMENT

The project is undertaken by **Binoy Gogoi** as a 10th semester project on "Game Designing using C-Sharp", under the guidance and supervision of **Mr. Pranav Kumar**.

Our primary thanks goes to him who poured over every inch of our project with paint stalking attention and helped us throughout the working of the project. It's our privilege to acknowledge our deepest sense of gratitude to him for his inspiration which has helped us immensely.

We are extremely grateful to him for his unstilted support and encouragement in the preparation of this project.

We show our gratitude to our Dean **Dr. Sajal Saha** and HOD "**Dr. Manoj Kumar Muchahari**" and our Project Co-ordinator **Dr. Purnendu Bikash Acharjee** for providing the best of facilities and environment to bring out our innovation, talent and spirit of inquiry through this project.

REFERENCES

[1] BJohnson, M., & Henley, J. A. (2014). *Learning 2D game development with Unity: a hands-on guide to game creation*. Pearson Education.