PSYBORG

Game development using Unity 3D

Anmol Sinha, Arya Dwivedi, Bhavishya Raj and R. Shravan
School of Engineering
Jawaharlal Nehru University
New Delhi-110067

***Abstract*—'Psyborg' is a 3D game we developed with Unity 3D in a team of 4 Computer Science and Electronics students with varying skills and expertise. A game development pipeline explaining the steps and the architectural diagram is presented. In addition, this paper explores the tool used in game development, challenges faced by the students and outlines the experiences gained during the project's implementation.**

Keywords—Unity 3D, open-world, formatting, game production pipeline, styling, insert, Waterfall model.

I. INTRODUCTION

The demand of more realistic look and feel in the games by the consumers is increasing the quality of graphics and the realism of games to higher level, every day. This has resulted in improvement of renderings, outstanding content, more realistic animation and more authentic behavior of artificial intelligence. Therefore, the work of artists and animators for the success of a game is now crucial than ever. Interactive computer graphics thus permits extensive, high bandwidth user computer interaction. This significantly enhances our ability to understand data, to perceive trends and to visualize real and imaginary objects, indeed to create a "virtual world" that we can explore from arbitrary points and views. It makes communication more efficient, graphics makes possible higher quality and more precise results or products, greater productivity, and lower analysis and design cost. This paper explains a basic strategy for the professional development of a content-intensive video game in a large team and shares experiences from our project. In the following, we will shortly introduce the tools that were used, schematize the workflow of our game development processes and finally explain some selected challenges that had to be overcome during the creation of 'Psyborg'.

'Psyborg' is an open-world adventure game with role-playing and stealth elements. While the openness of the game world is an important facet to games featuring open worlds, the main draw of open-world games is about providing the player with autonomy - not so much the freedom to do anything they want in the game (which is nearly impossible with current computing technology), but the ability to choose how to approach the game and its challenges in the order and manner as the player desires while still constrained by gameplay rules. The main appeal of open-world gameplay is that they provide a simulated reality and allow players to develop their character and its behavior in the direction and the pace of their choosing. The core gameplay consists of elements of third-person shooter and driving games, affording the player a large, open-world environment in which to move around. On foot, the player's character is capable of walking, running, sprinting, swimming, climbing, and jumping as well as using weapons and various forms of hand-to-hand combat. The player can operate a variety of vehicles, including automobiles, buses, semis, boats, fixed-wing aircraft, helicopters, trains, tanks, motorcycles, and bicycles.

Storyline*: Ligma, our beloved Cyborg is serving Shushi at Mihoyo Fishy-fishe as every day. But there seems to be something different today, she is not being rewarded the usual customers' love, something in her is dark, malign. Something asks for blood, something she has never felt before. She is feeling urge to do things she has never done, things for which she wasn't made. Can she be the Ligma she wants to be? Or will she become another 'Psyborg'?*

A. Problem Statement

To design and implement an open-world adventure 3-D game using Unity named 'Psyborg'. An open world is a type of video game level design concept where a player can freely roam a virtual world. The level will include everything that should be available in an action-adventure game like GTA.

The main appeal of open-world gameplay is that they provide a simulated reality and allow players to develop their character and its behavior in the direction and the pace of their choosing. The core gameplay consists of elements of third-person shooter and driving games, affording the player a large, open-world environment in which to move around.

B. Motivation

Studies show that video games help not only to improve social skills but also to learn basic educational principles. Even gaming companies like Mojang offer educators a transformative way i.e., MinecraftEdu to engage students using Minecraft and ignite their passion for learning. Video games have always been a big part of our childhood. The opportunity to develop games similar to what we have played in our childhood is in itself a big motivation. It would also allow us to learn more about how games are made and, also help us to increase our knowledge of Computer Graphics and enhance our grasp of it.

The main motivation to develop an open-world game came from the freedom which the player would experience while character development. The lifelike experience would help players in building life-altering experiences without having any worries. An open world also provides the player with a much more immersive experience and an opportunity to learn what goes to develop such an experience.

C. Key Challenges

a) Unity: Though Unity is a beginner-friendly engine and remains one of the preferred engines for game development, it also comes with some consequences. Creating the whole gaming world in Unity requires a lot of memory, thus making the execution of the game much slower and creating additional difficulties like more time consumption in loading the scene, importing assets, etc.

b) Complex interface: Unity is loaded with features, and a quick look at its interface can be really confusing, especially when used for the first time. Its power, though, makes up for the complexity. For instance, one can use 3D graphics in their 2D project, for a sleek and modern look. Creating basic 3D objects is also quite easy, as well as combining them into more complex ones.

c) Scripting Challenge: Scripts may be dependent on either time or player interaction. There are many things that should be considered carefully when creating the script. Some minor inaccuracies can stop the execution of the game and could be very time-consuming to debug a minor issue in the script that caused the game to stop.

d) Using tools: Importing various assets and tools, designing gameplay, setting topography and adding audio/video components is a big task for a beginner in unity. It takes a lot of time to get comfortable with it.

e) Management of complex interactions between animations: Adding animations could be tricky, considering various factors like keyframes, previewing animation clips, transitions and interaction between them. These animations are controlled by an "Animator Controller". An advanced Animator Controller might contain dozens of humanoid animations for all the main character's actions, and might blend between multiple clips at the same time to provide a fluid motion as the player moves around the scene.

II. Tools

The main tools used throughout the project were Maximo for modeling, animating and rendering and the Unity game engine for implementation. In addition, image, audio and video editing tools as well as drawing, sculpting and communication tools were used.

A. 3D animation (using Mixamo and Unity assets)

Mixamo is a freely available software, used to download characters and animations in multiple formats, ready to use in motion graphics, video games, film, or illustration. Mixamo's online services include an animation store featuring downloadable 3D models and animation sequences.

B. Game engine (Unity 3D)

Unity 3D is a game engine and complete integrated development environment (IDE) with an integrated editor, asset workflow, scene builder, scripting, networking and more. It also has a vast community and forum where any person wanting to know and learn to use Unity can go and have all their questions answered, which is very helpful in case of any queries. Unity 3D engine was used during this project as it has the capabilities the group needed for deploying the game made to as many mobile devices as possible. Unity can deploy the game made to Windows, UNIX, Mac, iOS and Android. Also, it is very easy to use and to learn. Developing with Unity is mainly based on drag and drop with occasional adapting of scripts rather than writing code. Apart from shaders and effects which can simply be turned on in some game settings, Unity provides numerous scripts which can be dragged onto 3D models. These scripts act for example as character controllers, follow up cameras or other important features.

III. Methodology

The development cycle that was chosen for the game development was Winston Royce's Waterfall Model. The waterfall model is where the development is broken down into different stages. Once a stage is completed, the user goes on to the next stage. The waterfall model allows going back to a previous stage, which is known as splashing back. This model was found the most suitable for the development as uncertainty and complexity were both low for the user interface design and user requirements.

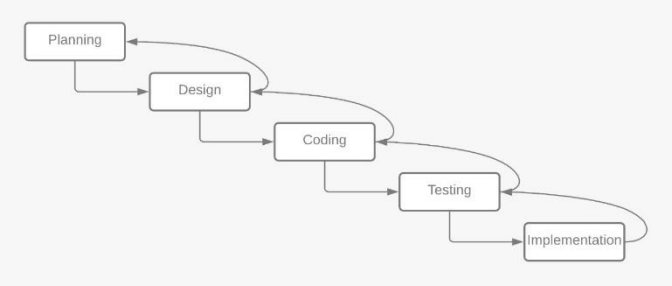

Fig. 1. Pipeline

A. Phase 1: Planning

Planning phase of the game includes characters to be used in the gameplay. Basic Structure and layout of the game, creating the storyline of the characters involved (the main character, 'Psyborg'), planning the number of scenes to be built, and what assets to include. This game requires a lot of storage space, so it is broken up into different scenes, so that only one portion of the game needs to be loaded at one time. An introduction scene (where the main character is introduced, along with an associated storyline), and the open world scene (where the city is loaded and the game begins to execute), were created to carry out the gameplay.

B. Phase 2: Designing

a) Assets: A completed game is normally made with the required game assets. The game assets consist of art assets and script assets.

- *Art assets:* Art assets contain 2D/3D models, textures, pictures, audio etc.
- *Script Assests:* Script assets are used to make all game objects to be performed. They are programmed with codes. Every script must be bonded with the suitable object. In this game, assets and textures were downloaded and then imported to the unity editor, and the standard assets packages were imported from Unity Assets Package. Packages are collections of game assets which can be imported into a project without significant dependency.

b) Adding GameObjects: Game objects are the objects which can be in sight and arranged in the game scene. It can be said that game objects are the essential elements which constitute the whole game.GameObjects are the building blocks for scenes in Unity, and act as a container for functional components which determine how the GameObject looks, and what the GameObject does. 3D objects like cubes, audio components, text to be displayed at the beginning of a new scene, are some of the game objects that we added in 'Psyborg'.

c) Animations: Many types of animations were used in the game depending on the scene. 'Walk', 'Run', 'Idle', 'Frightened', 'Aiming Pistol', 'Dying' etc. These animations were downloaded from the 'Mixamo' website. Mixamo provides a range of characters and animations, which were useful throughout the game.

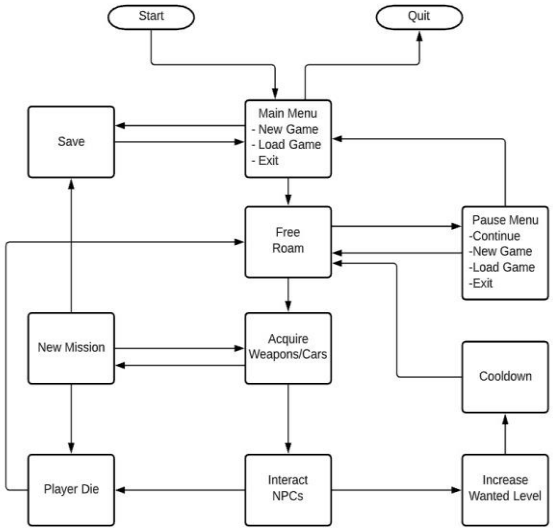

Fig.2. Architectural Diagram

C. Phase 3: Coding

Scripts were attached to the GameObjects in the scene in-order-to be called by Unity. The language that is used in Unity is called C# (pronounced C-sharp). All the languages that Unity operates with are object-oriented scripting languages. Like any language, scripting languages have syntax, or parts of speech, and the primary parts are called variables, functions, and classes. C# scripts were used for the coding phase of the game to control objects, scenes, and implement game logic.

a) Characters

- *CamControl:* To retake and control 3rd person player's camera.
- *CharControl:* To control the player's movement.
- *Firing Pistol:* To determine what happens when the pistol is fired.
- *NPC Interaction:* To have the conversation between NPCs or Non-playable characters.
- *NPC AI:* To control the behavior of NPCs.
- *NPC Alert:* To control the behavior of a NPC, when threatened.
- *NPC Death:* NPC behavior when shot dead.
- *NPC Destination:* To select where NPC goes while walking randomly.

b) General

- *RotationObj:* To rotate objects such as pistol spawns.

c) Missions

- *M001:* Mission 01 script
- *M001_A:* Mission 01 cut scene.
- *Global Ammo:* To spawn and use ammunition for guns.
- *Global Hints:* To display hints in hint box.
- *Global Wanted:* To control wanted levels.
- *Location Display:* To display location information when player goes into a new one.

d) Weapon

- *AmmoCrate:* What happens when the player picks up ammunition.
- *GunPickup:* What happens when the player picks up gun.
- *AA_Opening:* Determines the sequence of ___ in game.

Here is one of the samples of scripts that we used in creating this game:

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

public class A01_CamSwitch : MonoBehaviour {

    public GameObject firstCam;
    public GameObject secondCam;
    public GameObject Text;

       void Start () {
         StartCoroutine(CamSwitcher());
       }

    IEnumerator CamSwitcher()
    {
        yield return new WaitForSeconds(3);
        Text.SetActive(true);
        yield return new WaitForSeconds(3);
```

```
            secondCam.SetActive(true);
            firstCam.SetActive(false);
        }
}
```

This script allowed us to switch the cameras used in the game. There are several more scripts attached to implement the gameplay.

D. Phase 4: Testing

After the coding and implementation was the testing stage. This was where the game was tested. The testing ensured that the game worked as intended and aided in answering the academic question. There were three different tests done on the game. They were system testing, usability testing and beta testing. We faced many errors such as error in element's location when the view was maximized, which was due to error in anchoring. After fixing the errors, that element of the game was tested again. After every possible scenario was tested and all test results passed, the test was finished.

E. Phase 5: Implementation

During this phase, the video game was implemented so that it can be played by users. An executable file was created which would be used to start the video game. The executable file with all its dependencies are placed in a folder and then these files can be used in any target machine without the need for Unity Engine.

IV. Results and Analysis

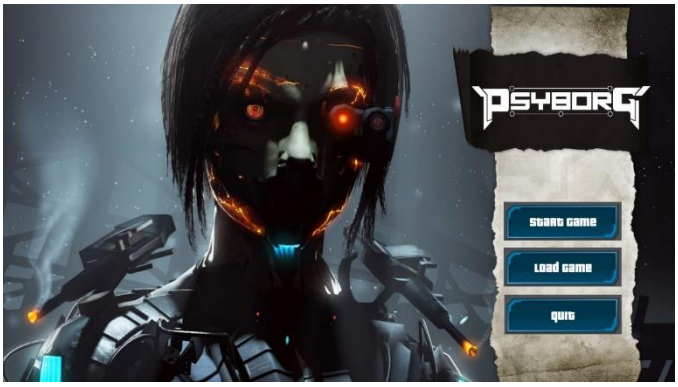

Fig.3. Main Menu

Fig.4. Gameplay

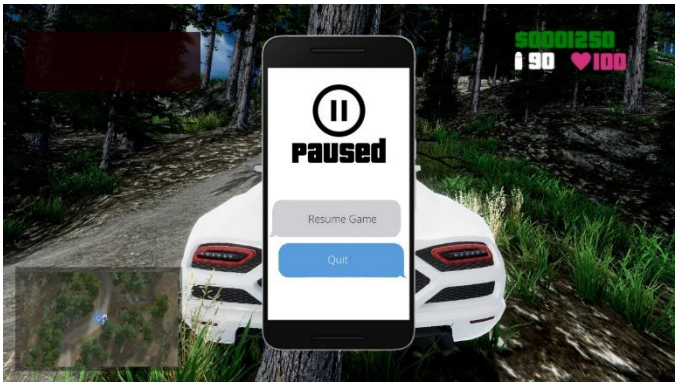

Fig.5. Pause Menu

Fig.6. Storyline using players and NPCs

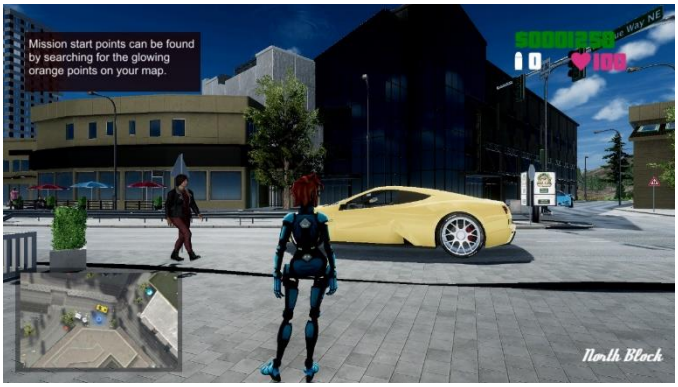

Fig.7. Game view with hint

'Psyborg' is a 3D game world with multiple axes of freedom for movement. There are random spawn locations and article inventory. We have also introduced interaction with common objects like doors, cars etc. There are life-like missions with different achievements for an enriched experience.

Acknowledgment

We would like to thank our Computer Graphics' professor Dr. Prerana Mukherjee for providing us the opportunity to develop a game as our project. In the process of completion of the project, we learnt a lot about game development process, game engine such as Unity and few tools like 'Mixamo' and

developed huge interest in game development and implementation.

Future Developments

For further development of the ‘Psyborg’, we have a set of plans which mainly include:

a) Level extension: Introducing new levels as well as new missions with an intersting script will make ‘Psyborg’ more interesting.

b) Improve Graphical Representation: Improved graphics and larger terrain map will add to the user’s interest.

c) Introduce new game features: Many new features can be addded both for the user and NPCs to make it as relatable as possible to the real world.

d) Take user’s response and improve: We would take user’s responses and crticism and improve.

References

[1] Unity 3D. Unity: Behind the scenes. http://unity3d.com/support/documentation/Manual/Behind%20the%20Scenes.html, 2010.

[2] Unity 3D. Unity: Using external version control systems with unity. http://unity3d.com/support/documentation/Manual/ExternalVersionControlSystemSupport.html , 2010

[3] Using Mixamo: For help in animations. https://helpx.adobe.com/creative-cloud/help/mixamo-rigging-animation.html

[4] Unity Technologies. Unity 3d. http://unity3d.com/