CMP 404

Applied Game Technologies

Marcus McGinn

1800573

Introduction

For this assessment students were tasked with the challenge of designing, developing and evaluating a game application. The game had to be specifically bult around and utilise, augmented reality (AR). To accomplish this, a game was made using the Unreal Engine to be run on Android devices. The game is a tower defence game which mainly relies on the AR technique of image detection. Image detection is a technology that enables games to identify and track variables in images and their locations in the real world. (Maruti Techlabs, 2021) This location can then be used to make game elements look as though they appear in a real environment.

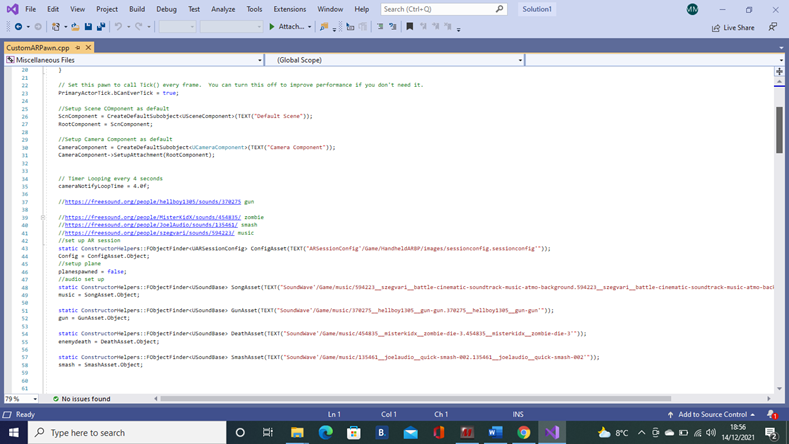
To play the game you will require the following images along with an android phone compatible with google AR Core. When the game is launched the user will need to press the start button and point the camera at image number one. Once the game application has detected the image the game will properly start. The image used to spawn the tower can be seen below. The game then spawns the start point of the game on that location which is a tower they must defend. Once the tower has been spawned the game will properly begin. Enemies will start to spawn and walk towards the tower, once they reach the tower, they will start to damage the tower and when it is destroyed the game ends. To fend off the enemies the player can shoot a projectile at the by pointing the camera at them and tapping the screen. The goal of the game is to defend the tower for as long as possible, the longer the game goes on the better.

A picture containing text, old

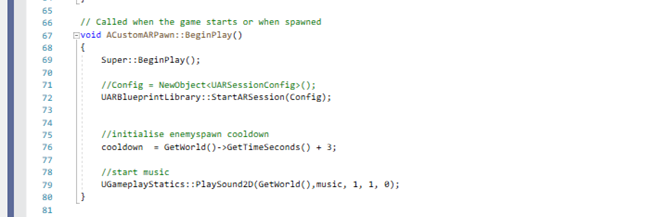
Description automatically generated

Implementation

To build this application, a combination of Blueprints and C++ were used depending on what they were best suited for. The majority of the games functionality was handled with C++, with the blueprint being used to handle the enemy AI and menus. The files used in the construction of the project include: CustomARPawn, ARPlaneActor, EnemyActor, BulletActor, WallActor, Gamecontrol and TowerActor. All of these inherit from the Actor class in Unreal, with a few exceptions which will be explained further. The reason for this is that the actor class is for objects in the game which have functionality but are not controlled by the player. The exceptions to this are the CustomARPawn which inherits from the pawn class because it has functionality and is controlled by the player, in addition to the Gamecontrol, which is a Gameisntance which is used to communicate between the c++ and blueprints. The CustomARPawn was used as the main file to control the functionality of the game.

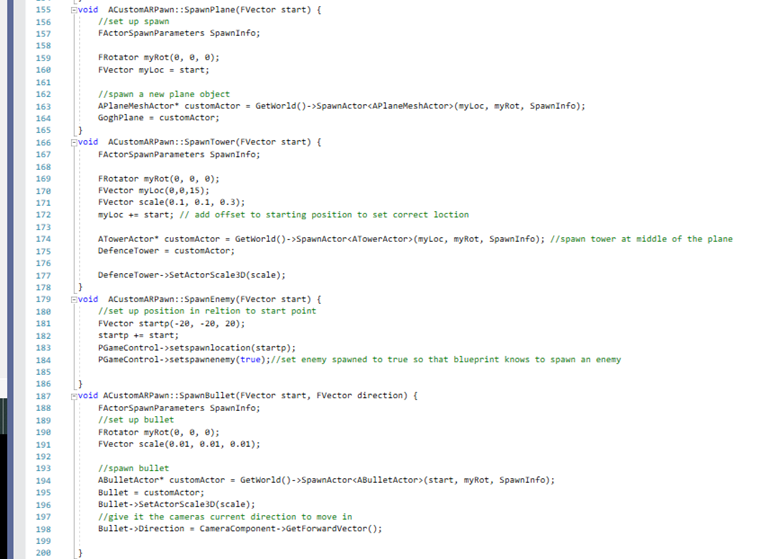


In the constructor several of the important aspects of the game are set up, including the camera component which is used the keep track of the players camera and the several audio tracks used in the game. The “begin play” function which runs when the game starts, begins the AR session, starts the game’s music and sets up the cooldown variable which will be discussed later.



The tick function runs every frame and its main purpose is to decide when things happen in the game. First it checks the game is running, if yes, then it will activate the FindCandidateImages function which will be discussed later. Next the tick function checks if the plane has been spawned so that enemies can be spawned onto it, however before activating the SpawnEnemy finction the cooldown variable is checked. The cooldown variable is used to prevent multiple enemies from spawning too quickly and getting stuck on each other, the way this works is when the game starts the value of cooldown is set to three seconds further ahead than the games timer. Every frame the cooldown value is compared to the timer and when the timer catches up, an enemy can be spawned and the cooldown is set to 3 seconds into the future again. This results in a three second interval between each enemy spawning. The setupPlayerInputComponent is a very simple function which prepares the touchscreen to be used by the game. In the OnScreenTouch fuction, the game is checked to see if it’s still running. In the event the game is running, the function will update the camera data and spawns the players bullet and plays a gunshot sound effect. This bullet is given the cameras forward vector so that it will continue in the direction the camera was facing.

Next to discuss are: SpawnPlane, SpawnTower, SpawnEenemy and Bullet spawn functions. Each of these are very similar in functionality and will therefore, be covered at the same time. They each receive a vector. This vector is where the game starts from and then they are offset by a certain amount to position them correctly in the game world, they are also used to set the size of objects in the game by setting their scale. The end result of these offsets is that the plane is spawned and the other objects are placed relative to the plane. The tower sticks out at the the centre of the plane and the enemies will be spawned to the left and near the front end of the plane. There are a few ways in which these functions differ for example, the BulletSpawn function receives two vectors; one is the camera location and one is the cameras forward vector. This is done so the bullet appears at the players location and the forward vector can be used to move it. The SpawnEnemy also differs in that it does not actually spawn the enemies, instead, it changes a variable which signals to the blueprint to spawn another enemy. The reason for this will be covered in the blueprints section.



The final function in CustomARPawn is the FindCandidateImages function which is used for image detection. The function checks for any images in view of the camera, checks what the image is and then checks if the detected image matches the tracked image of Van Gogh. When it has detected the Van Gogh image the function will find the location of the image and store it to be used elsewhere in the programme. This value is used to dictate the locations of the objects in the game.

The Cpp files for the other actors have very little in terms of functionality. They all set up the main components of the object which are the meshes, collision boxes, materials and scene components. The Bullet is a bit different as its tick function is used to update the bullets position and make it travel through the game. The tower also has some functionality because it has a Hp variable which is used to control the game and it communicates with the blueprints to tell them when to end the game. The Tower also has its own collision responses where it takes damage while enemies are touching it. To prevent it from dying too quickly a cooldown variable was used so it can only take damage every three seconds. The final c++ file of note is the Gamecontrol file which is a gameinstance used to pass variables from c++ into bluerints.The variables being passed on to blueprints are to tell the blue prints when the game starts, when the game ends, when to spawn enemies and where to spawn them.

The reason bluerpints were used in the project is because in the case of AI the developers of Unreal have stated that it is ill advised to make an AI in C++ as its not how unreal was intended to be used and in the case of the games Menus it was much easier to make widgets in the Unreal editor as they can be seen while they are being developed, to do the same in unreal would require trial and error and would be time consuming. (Mieszko, 2017) The enemy has three main components, the AI itself and the spawner and the NAVMesh. The AI inherits from the character class and uses the default unreal character mesh. The enemy simply checks if there is a tower in the scene and moves towards it. The enemy also checks if its been hit by a bullet, if it’s hit it destroys itself. Graphical user interface

Description automatically generated

The spawner is an Actor which is placed in the level, it uses the variables from Gamecontrol to spawn the enemies in the scene. The final aspect of the enemy is the Nav mesh which is used to tell the AI where it is able to move around in the game.

Graphical user interface

Description automatically generated

The main menu and end screen were made using widgets and each have one button. They use the variables in tandem with the level blueprint to start and restart the game when appropriate.

Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Reflection

At the start of this project, I was given the task of developing a prototype program that demonstrates Augmented Reality by using one of the techniques discussed during the module. The game developed used image tracking to place the game into the real world and allowed the player to interact with that 3-D space in a way unique to AR. What makes this gameplay unique is that in order to effectively play the game, players must move and aim their phone in the real world. Indeed, this could not be replicated by playing simply on a computer screen. There were many issues I came across during the development of this game and finding solutions to them was a challenging task since there are not many online resources that detail how to effectively work with AR in the unreal engine. One issue I came across was the problem of getting the blueprints and C++ to communicate with each other. My solution to this problem was the gamecontrol gaminstance I developed, which allowed both C++ and the blueprints to share data. Another issue I faced was that some objects developed would work well when tested in the editor but when used in the AR session the objects would act differently than anticipated or cease to work entirely. Due to these errors faced, there are aspects of the project which had to be cut. Initially I planned to use different types of images to spawn obstacles and slow the enemies down. A placeable wall was developed however, when it was spawned in the scene, the enemy AI stopped working and wouldn’t move. The reason for this is unknown as it worked as intended when the objects were placed in the level and tested with the editor.

Graphical user interface, website

Description automatically generated

The game could be improved by further developing the planned content as well as adding more AR technique. Cloud Anchors are a technique that could be used to great effect as the image detection isn’t always stable, Cloud Anchors could be used to create a much more stable positioning for game objects. (ARCore, 2021) If the game was further expanded with additional time, geolocation-which allows a users position to be tracked, could be used to change the games assets to match the architecture at the location for a player. (Google Developers, 2021)

Conclusion

In conclusion, the purpose of this project was to develop an AR game application which effectively demonstrates the unique features of AR and explore the potential of emerging technologies and their uses. To accomplish this an AR game was built which explored the use of Image tracking and its effectiveness in game development. In the development of this game, much was learned about the intricacies of handling AR and some of the challenges developers can face was noted. The development of the game also left room for research into other AI techniques and how they could be implemented in future projects or be used to improve this one. The end result of this project proves that there is much potential in AR and opportunity for expansion in this area, in the future AR is likely to continue to revolutionise games development.

Word count 1986

Reference list

ARCore. (2021) Introduction to Cloud Anchors on Android , ARCore [online] Available at: <https://developers.google.com/ar/develop/java/cloud-anchors/introduction> [Accessed 11 December 2021].

Maruti Techlabs. (2021) What is the Working of Image Recognition and How is it Used?. [online] Available at: <https://marutitech.com/working-image-recognition/> [Accessed 10 December 2021].

Google Developers (2021) Geolocation: Displaying User or Device Position on Maps [online] Available at: https://developers.google.com/maps/documentation/javascript/geolocation [Accessed 11 December 2021].

Mieszko (2017) Answers.unrealengine.com. Creating game AI in c++ instead of using blueprints - UE4 AnswerHub. [online] Available at: <https://answers.unrealengine.com/questions/673546/creating-game-ai-in-c-instead-of-using-blueprints.html> [Accessed 12 December 2021].

Assets

Freesound (2021) Battle - Cinematic soundtrack music atmo background.wav by szegvari. [online] Available at: <https://freesound.org/people/szegvari/sounds/594223/> [Accessed 14 December 2021].

Freesound. (2021) Gun Gun.wav by hellboy1305. [online] Available at: <https://freesound.org/people/hellboy1305/sounds/370275> [Accessed 10 December 2021].

Freesound. (2021) QUICK\_SMASH\_002.wav by JoelAudio. [online] Available at: <https://freesound.org/people/JoelAudio/sounds/135461/> [Accessed 14 December 2021].

Freesound. (2021) Zombie\_Die\_3.wav by MisterKidX. [online] Available at: <https://freesound.org/people/MisterKidX/sounds/454835/> [Accessed 10 December 2021].

Unreal Engine (2021) Starter Content. [online] Available at: <https://docs.unrealengine.com/4.26/en-US/Basics/Packs/> [Accessed 14 December 2021].

Upload.wikimedia.org. (2021) [online] Available at: <https://upload.wikimedia.org/wikipedia/commons/thumb/7/71/Vincent\_Willem\_van\_Gogh\_102.jpg/450px-Vincent\_Willem\_van\_Gogh\_102.jpg> [Accessed 14 December 2021].