CMP404 Report

1900512

Thomas Ballantyne

Usage Guide:

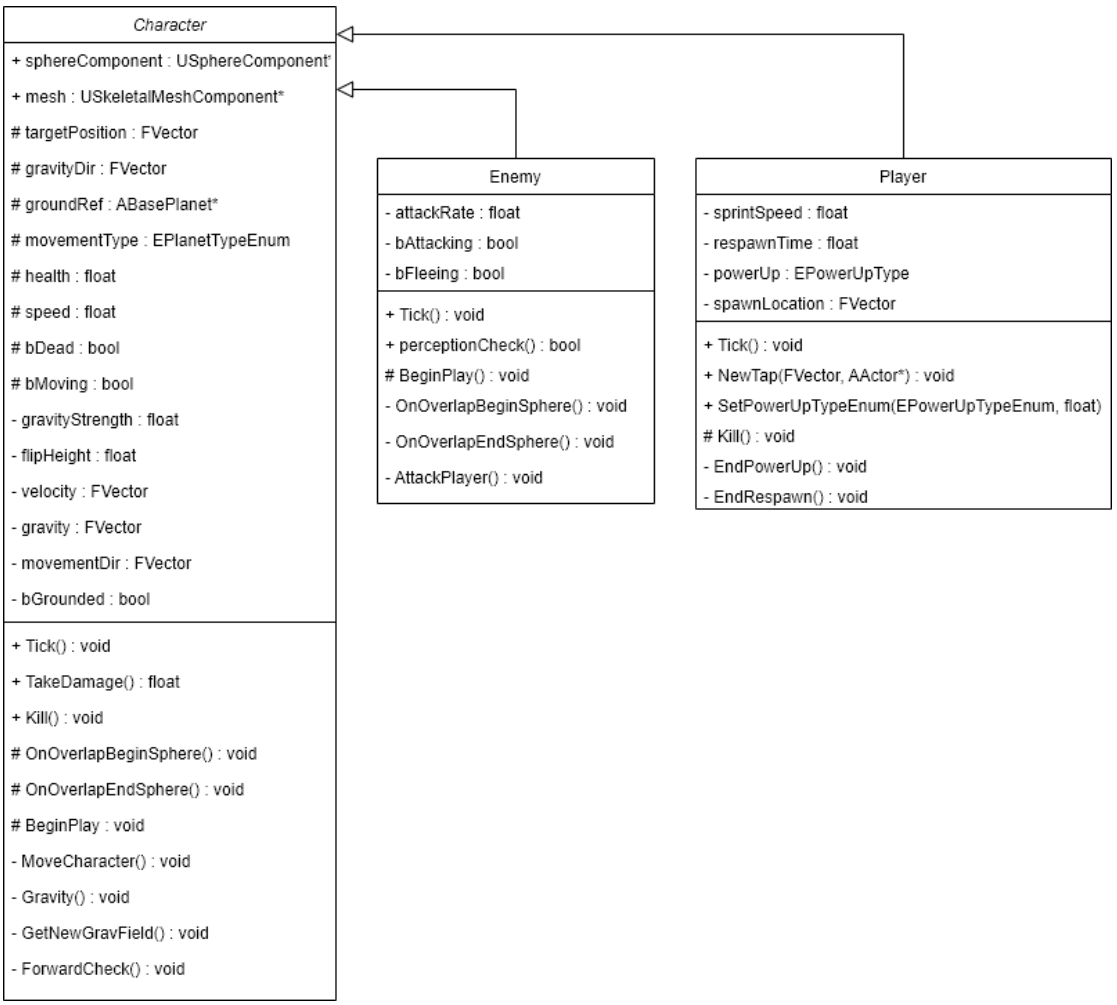
All menus can be navigated by either clicking buttons, or dragging scroll boxes & bars. From pressing play on the main menu, follow the on screen instructions of scanning for a plane and tapping to place the game world. If unhappy with the placement click the pickup button, otherwise click play. Physically moving the phone moves the camera, and tapping will cause the player to move in a straight line towards the tap position. Chests can be tapped to spawn a powerup. Power ups and collectables (such as coins) can be picked up by walking into them. By walking over springs or gas craters it will launch the player upward. Red pipes can be entered by walking towards the entrance. To complete the game, get the player to the red flag.

Features and Operation:

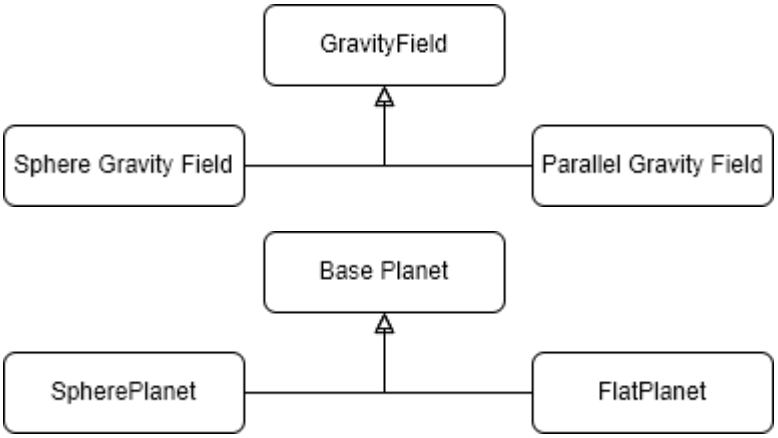
The game features a custom gravity system, which allows for the player to transition between flat and spherical planets seamlessly. The gravity fields are primitive shapes such that gravity is applied towards a centre point (such as in a sphere), or in a direction (used in the box). All gravity fields have a priority value, the highest priority gravity field that the player is overlapping with will take effect, this allows for creating smaller gravity areas within larger ones. An example of this is the upsidedown parallel gravity fields (box field), which reverse the gravity for the launch devices (e.g. spring) to get from planet to planet or the flat plane within the core of a planet (e.g. the sand planet).

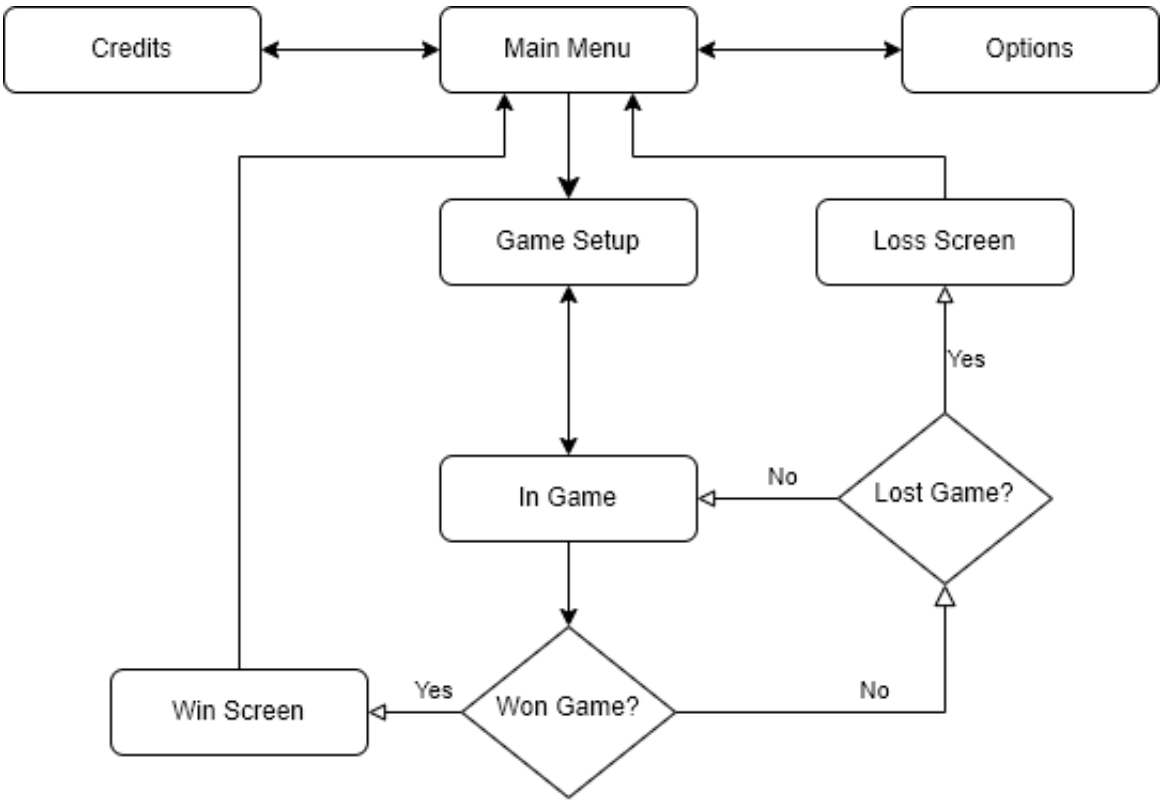
Design:

The two largest systems required for the application was the player and enemy movement around small planets with their own acting gravity similar to the ones found in Super Mario Galaxy (Nintendo Entertainment Analysis & Development, 2007). Since the player and enemy would largely share lots of functionality in terms of interaction with gravity and other core features such as movement and how they communicate with animation blueprints. For this reason a parent class for the character was used to implement that functionality, this saves duplication of code and makes the classes more readable and easier to understand. Unreal Engine's framework was built on to make communication between classes in the application manageable and easy to understand. The playercontroller was used for input, such as showing and hiding the cursor, and created widgets for the player. The game instance was used to store information that would be required from when the game was launched to when it was closed, as information like volume and other settings should not be reset when the level is restarted. The game state similar to the game instance was used to track data which was only relevant to the current game playthrough, such as coins collected and player lives. The game mode was used to handle core game functions in relation to AR, as well as handle tracking the win condition of the game and deciding when gameplay should end/transition.



The gravity system was designed to be expandable so that more types of gravity fields could be implemented with ease, for that reason a base gravity field class was made which had the main functionality of the class. Such as the public method which would take the player's location and return the direction of gravity (this will need to be done every frame for more complex shapes). The planet classes follow an identical format, as all planets have a gravity field, however not all gravity fields are connected to a planet. Planets will make use of Blueprint children in order to change the mesh and set the priority of the gravity field that belongs to it.





Exploitation of AR Technology:

As mobile Augmented Reality games are confined to a small screen as a viewport, it is likely that the player will go off screen at points of the game causing the user to lose track of them. In the implementation of the game when the player is far off screen a 3D arrow will appear to point in the direction of the player. This will hopefully prevent the user from becoming disoriented and lost while playing the game.

A large number of games have the player in the centre of the screen at most points of the game, the very nature of this game means that is not always true. Also the player is often quite small meaning animation can sometimes be hard to see, which could cause issues if the player is being attacked and doesn't realise. Considerations for this were sound effects and visual effects to represent damage, however a more powerful tool for this application was haptic feedback. For the implementation a vibration was set off on the phone whenever the player took damage, this gave a clear indication to the user which would effectively draw attention back to the player.

Finally, the main concept behind the game was a tiny galaxy existing within the real world, So the ability to place the game world anywhere was a key concept for the game. The most reasonable Augmented Reality technique to achieve this was markerless tracking, as this would allow for the user to decide any flat surface they want as the play area then they would be able interact with the world as if it has become a part of the real world. This was achieved by getting the user to scan their surroundings to generate planes, and the user would indicate where they want the centre of the gameplay to take place. An ARPin (Anchor)

would be placed there and the planes would stop being generated and removed (as this is expensive), the pin would then keep the world at the correct transform.

Reflection:

Approach:

When approaching the design of the application a theme of using gravity that allowed for interesting movement around a world that may not exist in the real world, and integrating it into the real world. For controlling the character I decided that a point and click movement system would be the best option for the user interfacing with the application, as a standard on screen control system for the player felt wrong when the player was on non flat planets. Generally point and click games take place on top down view type of game, where the mouse controls the movement of the viewport in a horizontal plane. By leveraging the AR system, the phone's movement can replace this function, and add a third dimension as a phone can be moved in all 3 dimensions compared to a mouses 2.

Effectiveness of design solution:

The solution produced was effective in terms of minimal code duplication and hierarchical design. As the characters, gravity fields, planets, and collectables were all designed using inheritance in order to reuse code that would otherwise have been duplicated. This will make expending the code in the future easy. However there was a limitation involved with using streaming levels, loading and unloading the level can be time consuming depending on the hardware being used. This lag cannot be hidden with loading screens/zones like with most games as the level only needs to be placed once a plane has been scanned and the user taps the screen to spawn it. However, one lacking area of the application was accessibility as no other players had tested the game, the scale of the world was set to my own height. Which may lead to issues to people who are shorter having issues accessing certain parts of the level. If there was more time for the project potentially getting the user to enter their height or calculating their height through AR means to scale the level height based on that.

Problems Encountered and Solutions Devised:

During development of the project there were many key problems that needed to be solved in order to produce the final game. The first problem encountered was getting a functional and good looking movement and rotation when under the influence of non standard gravity. Mainly, Unreal Engine 5 does not support navigation on spheres, so a custom implementation was needed. The way the movement direction was calculated was different depending on the shape of the planet the player was on, this was simple on a flat plane but more complex on a sphere. Due to the timescale of the project a simple solution was devised, which was to take the target position and then calculate a vector parallel to the sphere at the player's position, in the direction of the tap. When the player would change gravity field and the player would rotate to the new gravity orientation, if the fall of the player was from a large height, the rotation would look strange. To make this look more natural inspiration was taken from Super Mario Galaxy (Nintendo Entertainment Analysis &

Development, 2007) where the player would only rotate when closer to the ground, this gave a satisfying rotation.

Another problem encountered was how the game world was placed in the real world, and how placing the world at a rotation caused issues for setting locations. As the world would contain a large amount of actors and would need to be placed during run time. Two possible solutions were devised, the first being using a blueprint to contain the entire world and the second was to use streaming levels. Both methods would make use of plane tracking and AR pins in order to transform and orientate the worlds. Due to the large number of actors in the world and the ease of editing, level streaming was chosen, as it has all the necessary functions as the level can be loaded during runtime and transformed. However, this caused a knock on issue, the world's origin would no longer be 0,0,0, in world space at the base rotation, as the ARPin would offset the entire world. This effected teleporting the player when they went through a pipe, as the destination location couldn't be set using global location as that location would be relative to the persistent levels origin not the stream level. The solution used was to make use of unreal engine's USceneComponent and blueprint instances, as on the actor placed in the world, the offset of the USceneComponent can be set locally to the actor. This location is then affected by the streaming levels transform when the level is spawned, fixing this issue.

Opportunities for innovation:

One logical innovation that could be added to the game is multiplayer. However, multiplayer through a single phone isn't suited for this kind of game. As two players trying to use the phone at the same time isn't ideal, and passing the phone between players removes the sense of playing together. That's why I think cloud anchors could be implemented in the future. For Example, "Anchors" are access points represented by an identifier, where data is shared for multiple visitors, which gives to Augmented Reality creations the possibility to create a multiuser reality shared on the Cloud (Antunes et al., 2019). Furthermore, according to Huynh et al., (2009) By giving each player their own view of the hybrid space, handheld AR games allow us to make use of the core features of traditional board games. Overall, a cooperative AR game could be made which makes use of cloud anchors in order to share a single player experience between two players by making good use of the core features of the game.

References:

Literature:

Antunes, J.L., Bidarra, J. and Figueiredo, M., 2019. AR With Cloud Anchors: A Way to Improve HCI and Interactive Art. International Journal of Creative Interfaces and Computer Graphics (IJCICG), 10(2), pp.29-40.
Huynh, D.N.T., Raveendran, K., Xu, Y., Spreen, K. and MacIntyre, B., 2009, August. Art of defense: a collaborative handheld augmented reality board game. In Proceedings of the 2009 ACM SIGGRAPH symposium on video games (pp. 135-142).

Games:

Kunabi Brother GmbH (2017) Euclidean Lands [Video game].
Paladin Studios (2018) My Tamagotchi Forever [Video game]. BANDAI NAMCO Entertainment Europe.
Super Mario Galaxy (2007) Nintendo Entertainment Analysis & Development [Video game]. Nintendo.

Assets:

Ultimate Platformer Pack, Quaternius, 2021, CC0.
Modular Platformer, Keith at Fertile Soil Productions, 2020, CC0.
Poly Desert Freebies, RunemarkStudio, 2019, CC0.
UI Audio, Kenney.nl, 2015, CC0.
Free Footsteps Sound Effects, Mayra, 2020, CC0.
Nature Sounds Pack, Antoinemax, 2021, CC BY 4.0.
RPG Music Pack, SVL, 2020, CC0.
Wooden Crate Being Broken Into Pieces, freesfx.co.uk, CC BY 4.0.