*03/05/2021*

Hamza Oudrhiri

K1910115

**Media specialist project**

***Group 4***

Individual Report- Game programmer

*Table of contents:*

1. **Introduction**
   1. Use of PUN v1
   2. Game lobby
   3. Turn Based Strategy Framework
   4. Dividing the game implementation in smaller parts
2. **Features and practices**
   1. Unit movement and Attacking System
   2. Turn manager
3. **Results, Feedback and Conclusion**
4. **References**
5. **Introduction**

As a child, I had good memories of playing Final Fantasy Tactics Advance *(1)* on the computer using a GameBoy Advance emulator. I suggested creating a similar game to my first team partner. After coming up with several ideas, we eventually decided to make a Turn Based Strategy game. This type of game consists of two teams that hold several units of different types, all placed on a map which is itself composed of tiles. The players can move their units and attack the enemy’s units.

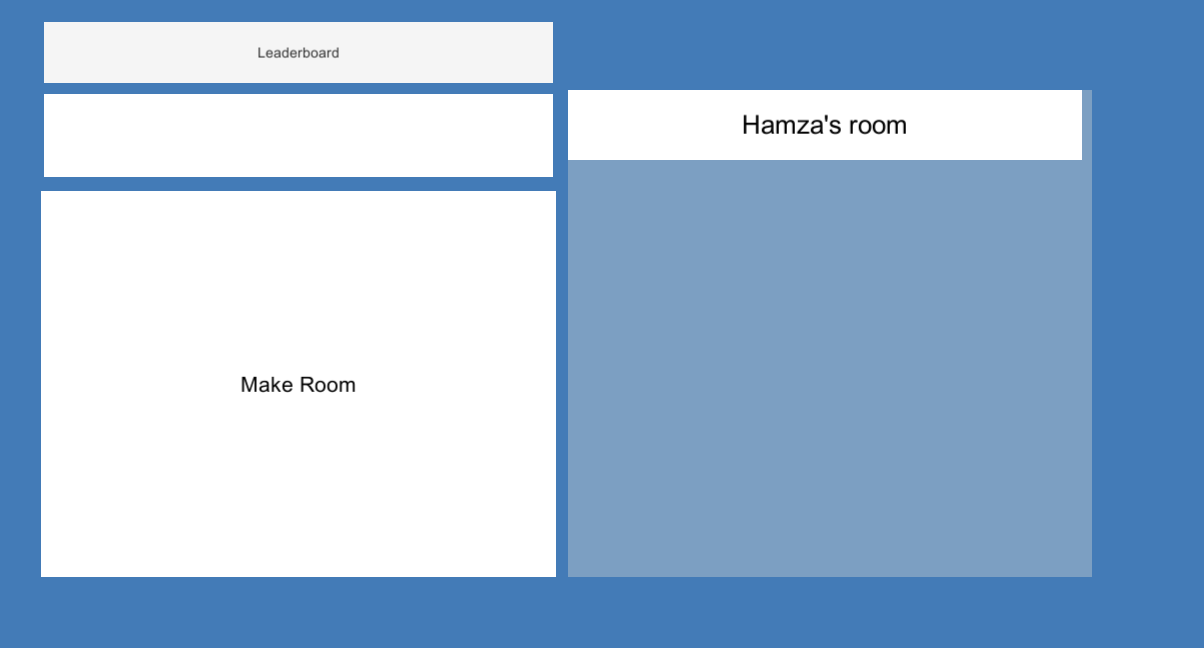
* 1. Photon Unity Networking

For this project, we were assigned to develop the multiplayer side of the project on a pre-built networking framework, we had to develop it using PUN (Photon Unity Networking). The most recent version of PUN is version 2, but for our project that consists of a turn-based strategy game we were told that the version 1 contains a pre-built turn-based manager. We initially implemented the game using the pre-built turn-based manager but in our 2nd iteration of the game we created our own manager. The networking aspect of the game was entirely making use of RPCs (remote procedure calls), which are methods that are called on remote clients.

* 1. Game Lobby

The game lobby was also created using the PUN v1 framework and a tutorial found online *(3)*. The user can create a new game room which he names inside the text field, once the ‘Make Room’ button is clicked, the user transitions to the room lobby, where he can see the names of the users currently in his room and the option to start the game.

The lobby contains a list of rooms on the right panel, upon clicking a room name the user joins that room.



*Figure 1: Lobby with an existing room called Hamza’s room*



*Figure 2: Room lobby showcasing the players currently inside the room*

* 1. Turn based Strategy Framework

We initially wanted to develop the game using the Turn Based Strategy Framework *(2).*

It was very user friendly, and included tools to set up the project for a turn-based strategy game. We managed to create a tile map and units of different teams that were able to move on the map.

we ran across an issue when we tried to sync the units’ movement across the network as for RPC to be called, a Photon View had to be attached to the gameobject that called the RPC, and this was unfortunately not possible for our case.

* 1. Dividing the game implementation in smaller parts

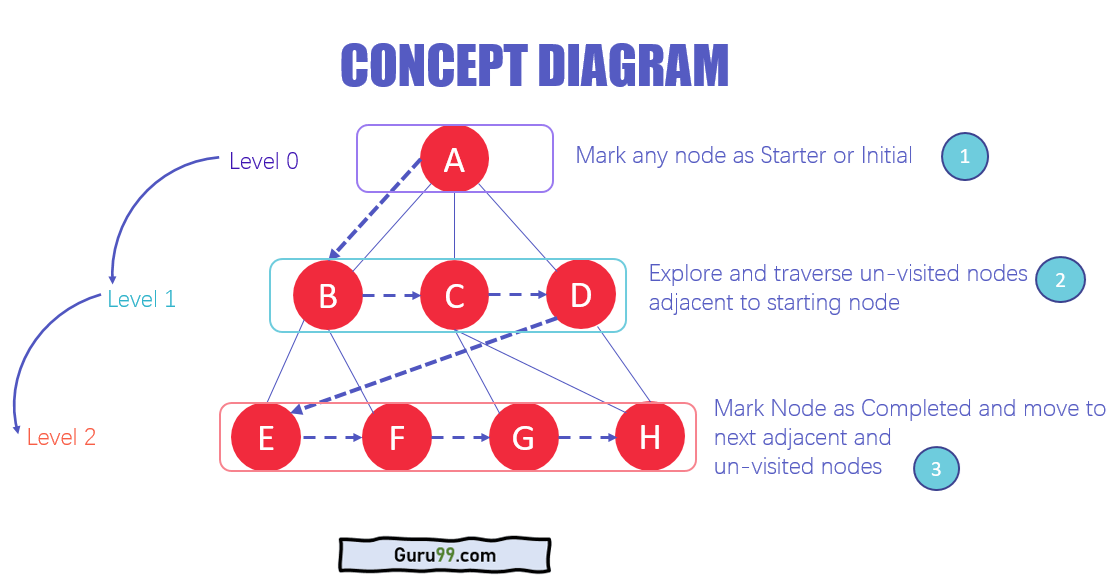
With that in mind, we decided to make the game from scratch and divided the implementation into smaller parts.

* Setting up the map
* Unit movement
* Movement sync across the network
* Attacking system
* Turn manager
* Unit stats and experience

1. **Features and Practices**
   1. Unit movement and attacking system

The first step in the development of the game was creating a tile map that allowed for unit movement. We decided to use cubes for the tiles, and gave each of them a ‘Tile’ script and a corresponding TileID as an integer, which was used in the Remote Procedure Call. Because GameObjects were not allowed to be passed as a parameter into the RPC, we passed in the ID of the tile and the function searched in the list of all the tiles which one corresponds to the ID passed as the parameter. The ‘Tile’ script involved a simple State machine, where in the Update() function we changed the color of the tile based on the state.

We used a breadth first search system to show the user which tiles a specific unit can move to. This system took the tile that the unit stood on, and found each neighboring tile, and kept searching for the neighbors until the distance from the starting tile was greater than the unit could move. We kept track of which tiles were visited with the use of a Boolean assigned to each tile.



*Figure 3: Explanation of Breadth first search concept*

Once the selectable tiles calculated, they were showcased to the user with a different color, and the user could choose his destination by clicking on a selectable tile, thanks to the use of ray casting. Clicking on a selectable tile called an RPC which was itself called on all clients.

We used a Queue system to compute the path the unit should take, starting from the end destination, the tile was pushed to the queue, and tiles continued to be pushed until it reached the unit’s current tile. Once the path calculated, the system looked at the peak of the queue and initiated the movement by moving the unit onto that tile, and then popping that tile within the queue, it kept doing so until the unit reached his final destination. Unit were able to jump to tiles of various heights by moving the unit to the edge of the tile and adding a vertical velocity.

A similar system was used for the attacking part of the game, where the system would find tiles within the attacking range and if they were occupied by an enemy unit it would flag it as selectable. The enemy units within range would then become clickable, and would inflict damage once clicked. The damage taken was calculated using both the attacker’s stats and the target’s stats.

* 1. Turn manager

We implemented the turn manager within the game controller. The game controller contained an array property that held the units of the player. The first unit of that array was selected first, and once the player moved and attacked, it would move on to the next unit, until all units were played with then it would change the turn to the other player.

It was a fairly simple script but in retrospect we realized we could have coded the turn manager more efficiently, using a list of teams and a queue that contains the current team and the next team, where when your team’s turn was over it would push your team at the back of the queue, and checking when to switch turn by analyzing if all units have moved and attacked thanks to a Boolean attributed to each unit.

1. **Results, Feedback and Conclusion**

Overall, I am pleased with what I have been able to produce, especially in regards to the movement system. There are not many resources online for turn based strategy games so I had to create solutions that were new to me. In terms of the programming and implementation, I have achieved what I set out to do, but due to the lack of feedback from the designers, the game might not feel balanced or completed. Indeed, there was not much communication between myself and the designers and most of the design of the game itself was implemented by myself.

In terms of the future of the game, the next steps would be to implement proper animations to the units, make the user interface more user-friendly, and research on ways to balance the game even further. Additionally, we could add more features such as a turn manager based on unit’s speed and also the use of items that could have special effects.

1. **References**
2. Game Design Guide (2017) *Final Fantasy Tactics & Combat initiative system*. Available at: <https://www.youtube.com/watch?v=NtUNI59UzMU> (Accessed: 26 April 2021)
3. Crooked Head (2020) *Turn Based Strategy Framework*. Available at : <https://assetstore.unity.com/packages/templates/systems/turn-based-strategy-framework-50282> (Accessed: 24 April 2021)
4. First Gear Games (2019) *Unity – Photon Networking.* Available at: <https://www.youtube.com/playlist?list=PLkx8oFug638qVMIrtqOnwmqnW6o8WDgQ1> (Accessed: 26 April 2021)