Post Mortem - 15909644

Overview:

I will be talking about things I missed in the gold release date. What went well and what didn't go well. What I learned from this. And my conclusion.

Overall, I didn't get to complete the game in time for the gold submission. Core elements missed in the final release date is the splash screen, a menu, being able to start a new game without closing the exe, not having GUI displaying the player's health, base's health, wave counter, and no sound.

The game is playable for one round, until the player dies, there is no way to restart the round unless they close and open the application again.

What went well:

- Player movement, the player can move into any lane, but will always move to the center of the lane, and velocity is consistent.
- Collision, the player can collide with the enemies and the enemy bullets. It only checks for collision when it comes near to the player, because in my game the player's Y position stays constant. The player's bullets can shoot the enemy.
- Walls, I had a 2d bool array to check if the slot if free or taken. If not, then I can add a wall at that position and add it to the wall container. The wall container will remove and add when a wall is created or destroyed. I used the same collision code with the bullets to make the enemies stop and go into a wall attack state where it would have to hit the wall and break it in order to move.
- Data driven design with enemies spawning.

What didn't go well:

- No get to do a menu.
- No get to splash screen.
- Not enough planning on the TDD to help me early on. No ULM graphs.
- Not writing good code.
- Had trouble with getting the animated sprites to work.
- Not understanding the COMP710 framework fully in terms of back buffer, which gave me trouble when it came to make the animated sprites.
- Once I finished completing the wall, with all its properties and how it interreacted in the environment, my code became messy because of factors I didn't account for such as dependencies. Therefore, I came up with a hacky solution that worked but it could have been a lot cleaner with added variables. Maybe having a container storing all the enemies hitting that wall, and once it dies it will iterate through the container to set the wall attack state to false.
- Didn't get SDL ttf to work, therefore there's no GUI showing the stats changing, such as the wave counter, player's health and base health.

Lessons learned:

- I should commit to the repository a lot more, this way I would of course have version control if anything bad happened such as losing the files or messing up the build.
- I should plan the architecture of the code better. Making things less complicated by having a solid structure to follow.
- Stick to coding standards with naming conventions.
- Should use more project management tools such as Trello board to keep track of my progress and having a to do list with a schedule behind it.
- Having a project schedule plan and sticking to it.
- Seek help as a last resort.

Conclusions:
Overall, I think the code quality got worse at the end. There are so many areas of the code where I could break it up into smaller sections and make it more object orientated. Having repeating code such as iterating through many containers. As a last resort seek help.