Game description

Our game is a maze game called Haunted Hospital. There will be a main character who will need to collect all rewards (key cards) in order to open the exit door. There will also be a starting point of score 100 that keeps decreasing as times goes. There will be bonus rewards(candy) which can help increase the score and traps (blood stain) which decrease your score. If the score goes to 0, you lose. Throughout the game, there will be two types of enemies which are ghosts(can go through walls) and demons (can't go through walls), which can cause you to be lost directly when colliding.

Original plan

While our original plan is a completely different theme, which is an underwater theme with fish, pearl, and shark. However, in the process of building the UI design, it is hard to find the animation of the icon, specifically like fish going to the right, left, up, down. So, we decided to change our theme which currently can show a very good animation where the main character icon will change showing movement of right, left, up, down.

What we learnt

1. Time management
   One of our main issues is time management, where we often work close to the deadline, which makes our results not maximize. As the result, we working mostly at the same time, and committing is such an issue in our beginning phase, where some of us not really familiar with git. The issues are relating to pull and merge, then push, which we find challenging.
2. Good planning
   We realize that our code is a bit unorganized and too much spaghetti code where there are high dependencies, resulting in lots of refactoring. These issues can be avoided if we plan everything properly from the beginning.
3. Communication
   In the beginning of the project, we don't communicate quite well, resulting in overdoing some tasks where we make things that others are already doing. Throughout the semester, we manage to have good communication and keep updating what has been done or changed using Discord, in order to avoid miscommunication and updating the current situation of the code.
4. Pair programming
   We also find that pair programming is very useful when doing the project, where we always make sure that after we write our code, especially important parts of the code, we always ask someone to check out code and give some feedback, if any. In this way, we can get a better code and each of us is understanding what is going on in our code.