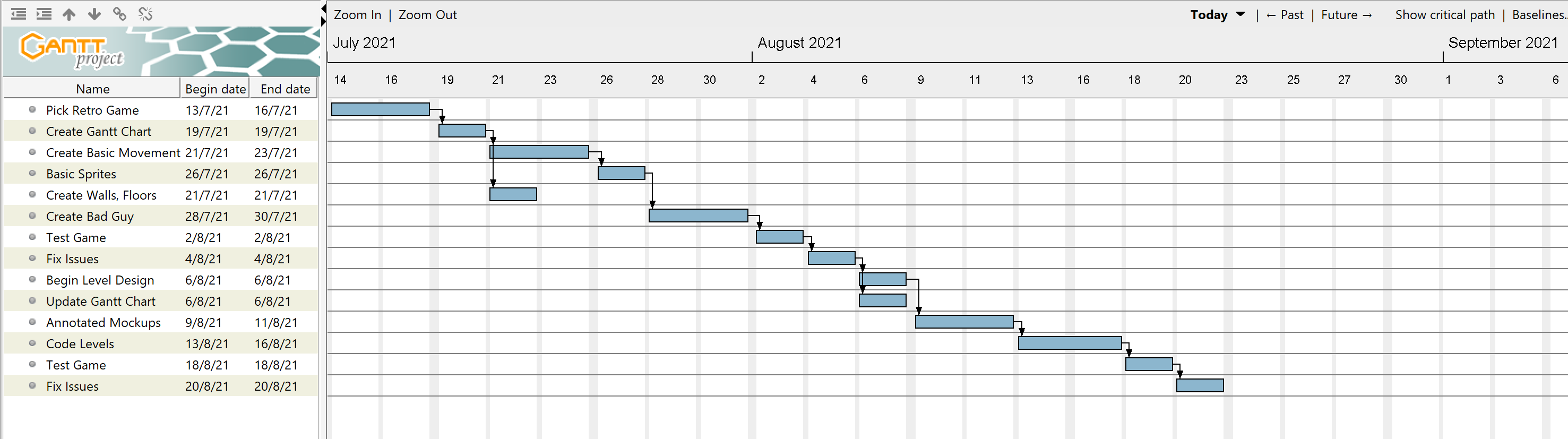
Journal

Gantt Chart 

The Gantt chart shows all the tasks that need to be created for the project to be created, on time. It runs some tasks simultaneously, because multiple people in our group can work on the different tasks. It also only runs tasks on Monday, Wednesday, and Friday as these are the days that we have Computing Class.

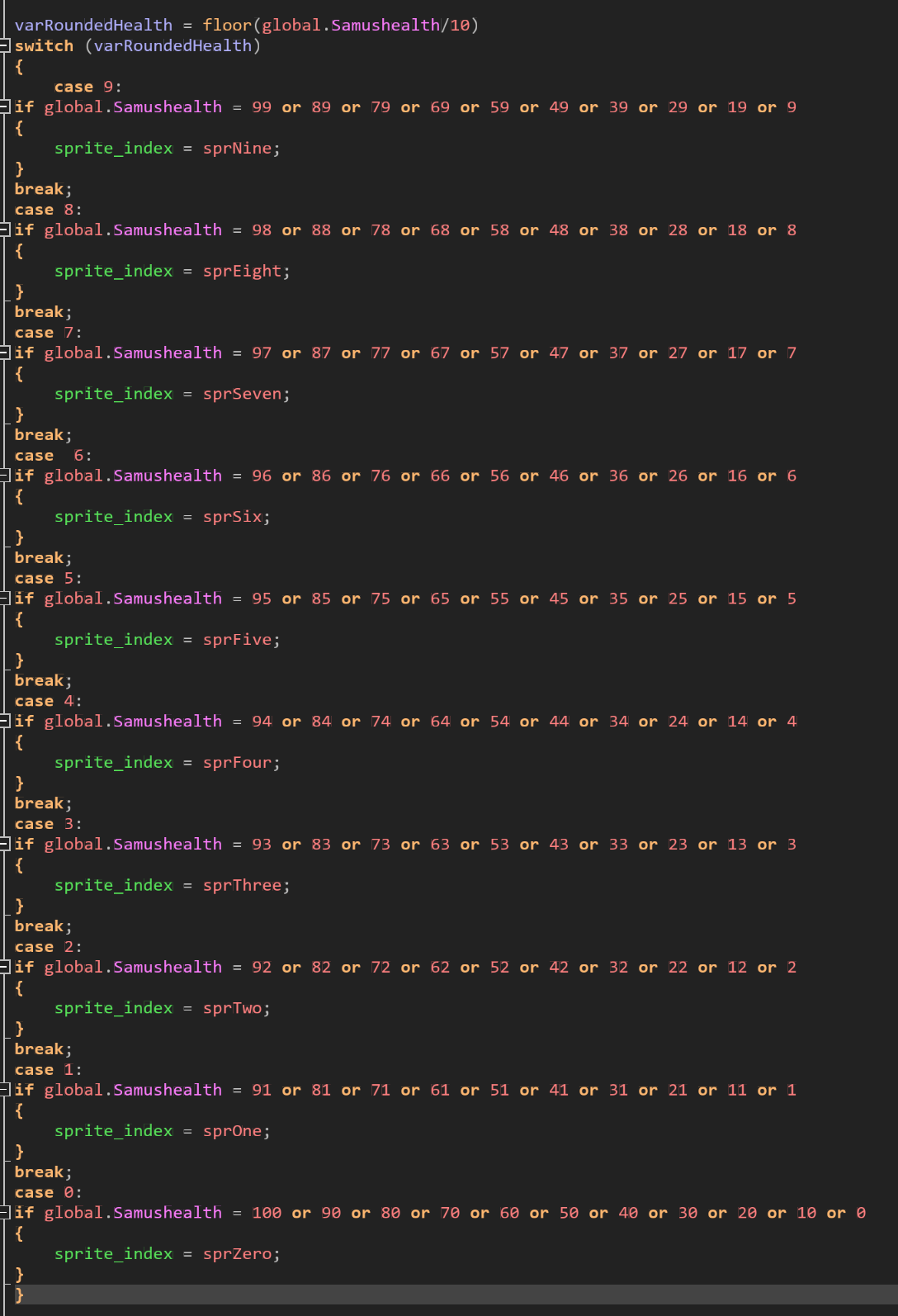
Creative and Critical Thinking

|  |  |
| --- | --- |
| Creative | Critical |
| **Creating a GitHub Repository** – Using our previous storage medium of OneDrive created a problem with two or more members working on the same project simultaneously. Because of this, we found a different storage medium in which merging different files can be more streamline and the files are still stored online so anyone can access it. | **Merging conflicts**- While using GitHub, we struggled to combine changes from multiple users working simultaneously. This often led to code being deleted, and work being overwrote. We eventually figured out how to use merging to keep all the changes without losing important code. |
| **Creating a GUI-** An appropriate font could not be found for the GUI, so it was decided to use a case statement, using sprites to create an accurate looking GUI that fit the look. This used a statement for the 10’s and the 1’s, tracking the variables and displaying them on the screen. | **Hornoad movement-** When creating the Hornoad, the AI kept running into issues getting stuck in the wall and not jumping. Using trial and error, we found code that worked around the collision problem, making the Hornoad turn around when it hits a wall |
| **Controls-** The original Metroid II was built for the Gameboy, meaning that we had no reference to go off for controls. We decided to use arrow keys for movement, as this is most common in most games and feels the most natural. | **Samus’s health & iFrames** – When Samus would collide with an enemy, he would lose all her health very quickly as the code ran in the step event and would drain her health. To fix this, we used an alarm that prevented Samus from taking damage within 60 frames of taking damage originally |
| **Sprites-** Samus would perform tasks such as shooting a missile or rolling, but the sprites would not match this. In order to fix this, we had to find many sprites that fit all the states that Samus could be in, and all of their actions. | **Samus’s sprites moving**- When samus would change sprites from standing, to shooting and more his origin and hitbox would change, meaning he would clip into the ground. In order to fix this, we had to go through all the sprites and manually align the origins so that they stay centered throughout all of Samus’s states, not just the one sprite |
| **Color­-** Since the original Metroid II was made for the Gameboy, it was completely in black and white. We considered transferring the game to colour, to give it a more modern feel however we decided to stay with the black and white to stay true to the original game and the artwork. |  |
|  |  |

**Creative thinking** refers to how you use creative methods of resolving problems that you encounter (in this case, as you create your “retro” game). You will let me know how you tried to overcome problems that you encounter which might include use of external resources, people, problem-solving methods.

**Critical thinking** refers to evaluating how well you have accomplished a task or how well you have addressed the specific requirements of the game eg how well you have met the functional and non-functional requirements of the game. You might explain how you changed your approach or the design or creation of a room to best meet the non-functional requirement of motion in the room to keep the player motivated.

**Three New Skills**

1. **Case statements for GUI**

An appropriate font could not be found for the GUI, so it was decided to use sprites instead. This meant that a case statement had to be created, to effectively track the health and ammo left. There are 2 case statements for each variable present on the GUI, one for the 10’s, and one for the 1’s. The left shows the code for the 10’s for Samus’s health.