KYLE

DOMENICO

Week 1: Didn't have major help with it except for ideas, but needed to do something for week 2 and week 3

I added a sort mechanism for the Shop to allow the shop to be sorted with S

The sort function in the shop organizes items by price, either from lowest to highest (ascending) or highest to lowest (descending). Each time you click the sort button (or press S), it toggles between these two orders and updates the display accordingly.

For the list this is how it works!

shopItems: list of all shop items

Sorted by GetPrice() (price of each item)

sortAscending: controls order (low to high or high to low)

After sorting, items are reordered in the UI

Week 2: I've modified some of the code into a linked list instead. I edited all the damage-related .cs files that added damage into a linked list. This means ALL the artifacts that added damage only, and not the ones that are chance-based. This also meant that I made an Infinite damage command by pressing a button. (This could also include the other artifacts because of the use of ArtifactManager.cs, but for now, we use one artifact as a starter)

So this implementation enables:

Ease of removal or damage to artifacts

The ability to add infinite damage, or later on, add the ability to add those artifact modifiers to the player.

Efficiency of adding or removing modifiers (Pure damage artifacts)

Possibility of stacking the artifact

Linked List added for bats! Bats now spawn an additional time.

The bat linked list controls the order of enemy spawns.
 Each bat is a node, and when one bat is defeated, it tells the next node in the list to spawn the next bat.

Week 3: I've made a behavior tree for the enemy's turn AI. So this would make them pick what to do based on the nodes given, and since they were turned into a behavior tree, I could implement a command that would disable their Turn AI and make them unable to attack the player. It's as if their turn was skipped. Dom and I worked and added some more nodes to this to make the game more interesting. I added some nodes so the player can pick at random to either attack twice/once, or defend and or heal.

So, how it works is that AI needs different nodes to do different things. For example, it needs a node to attack, one to defend, etc.

Then, it picks from those 3 depending on what it gets picked. Since it's randomized, it picks from the 3 different options I put on the list of nodes it can pick from.

Then it picks those for each turn.

(explanations of the other nodes here)

HealNode.cs: This is separate from the other 3; this will be a priority instead of being random. If the player damages the entity, it will prioritize

(combined randomized pick)

These 3 are picked through a randomizer

DefendNode.cs: What this does if this gets picked is that the player's damage gets halved

AttackNode.cs: Simple node, it just attacks the player based on the damage number given on the entity card

AttackTwice.cs: This will enable the AI to attack twice.

Also, for visuals, it just gets the updated counter for damage and shows it (defend) sadly not for healing

I originally wanted the tree to be a skill tree and tried implementing special abilities but decided to wipe that portion and we decided the best idea was to work together on this portion do to its size.
I worked on the Heal Node and Defense Node. And then kyle helped me fix it up on his own machine .
Kyle worked on the initial tree data structure, and we have different commits due to fixing and working on the nodes together

Difficulties we had:

Kyle: I had difficulty figuring out something for the tree, but researched a bit and saw that I could make a behavior tree instead, which is good for AIs in general. We wanted to do something else, but couldn't make it work in the end. Debugging for the most part, it was long but rewarding, and of course, time. But I also had difficulty trying to add the Artifact by pressing a button, and it was still really hard because it clones the object instead of it already being there. I was only really able to make the Icon appear.

Domenico: I had difficulty figuring out an idea for the Tree or Graph, We wanted to do a skill tree initially but found that we could implement a behaviour tree instead for the mobs, this would allow us to bypass the UI interface needed for the skill tree, and allow us to know if the mobs are actually doing what they are supposed to with the console.

I also had difficulty understanding the way that the game is set up but with time it became easier

Conclusions:

Kyle:
For the most part, editing someone's game to add more stuff was fun. If we had more time on this, we would have added more ideas to certain parts of the game. The idea of editing

someone’s game is a really good way to practice and to know what ideas you can add to an already-made game from someone else’s save file.

Domenico:
I realised that editing someone else's work is hard, and commenting and organization are super important. It's like stepping into someone's brain and trying to pick it apart. I enjoyed this activity, and it makes me want to become a better game developer