XbtOS Write-Up

Introduction:

For this game project, I tried to create a deck-builder roguelike, called "XbtOS", or **"Live by the One Shot, Die by the One Shot**." The idea is that health does not matter; you kill everything in one hit, and everything kills you in one hit. Instead, to kill things, you need to charge up a single attack strong enough to attack the enemy. To avoid being hit, you either must dodge physically, or block the hit fully.

Motivation:

Last year, my friend came up to me with an idea for a deck-builder roguelike (he's really into those), and I thought it was a cool idea. I wanted to implement it for this class; however, I didn't know the project had switched for the first time this semester. It wasn't until the option for the game project returned that I decided to go with the original idea. It would help me learn a graphics library so that I might do interesting projects later on, and would also satisfy the project conditions; the hero and cards would be player-controlled, and the enemy would be autonomously moved.

Implementation:

The only library used is SDL2 (Simple DirectMedia Layer 2) and its extension libraries. Everything else is in the C++ standard library.

For the "Hero"/Player: The hero is not a class object, but just a collection of variables, including their current charge level, and a hero deck object. Position is not needed, because the Hero is always at position (0, 0), or the center of the grid. The hero deck class includes the Hero's full deck of actions, the Hero's current deck (which is refilled by the full every battle), and the Hero's current hand (which can hold up to 7 actions). Every Action object has 4 things it can do: move (a list of Coordinates the Hero can move to), block (a list of DAMAGE\_TYPE the Hero blocks), charge (an amount to increase the Hero's charge level by), or strike (try to attack; only successful if within 1 tile of enemy). The player also has access to 4 free actions, that don't get consumed and are always available: Pass (do nothing), Sidestep (move to a 4-adjacent tile), Charge (charge 1), and Strike (only attack). The actions are all implemented by an init function, that places them all into a global hash table of Actions, accessible by their "ID".

For the enemy: There is currently only 1 enemy per battle. Every enemy has a Power level that determines how much charge is needed to kill them, and a Moveset that determines its AI. Every Moveset is a set of Movegroups that is randomly selected from. Every Movegroup describes AI over multiple turns; it is a sequential list of Moves. Every Move has 2 things it can do: move (a range, detailing the max distance, rounded down to the nearest tile, and a direction, whether it is towards or away from the hero.), and attack (a set of threatenedTiles, and a method of attack.) A threatenedTile is simply a Coordinate with a DAMAGE\_TYPE attached to it. The methods of attack determine whether the attack is unaimed, aimed but projecting from the enemy, aimed but placed up to a range, or placed directly on the hero globally.

For the main game loop: the game loop occurs a certain amount of TPS or ticks per second; currently set at 30. Events like mouse movement or keyboard input are handled every tick. All decisions per tick happen based on what "scene" the game is currently on. The three current scenes are the title screen, the main game, and the game over screen. The title screen only has a play button, that switches to the main game. The game over screen is the same way, only with a "play again" button. The main game has an 11 x 11 grid where the representation of a "battle" occurs. Every battle initiates with a random enemy. When a battle initiates, the entire player deck is reshuffled and refilled in a random sequence. Every battle is in turn-based sequence--enemy and player switching off constantly until one dies. Every player turn, actions are drawn to fill the hand up to 7 actions or until the current deck runs out of actions. When selecting an action, if it has movement options, you must pick a movement tile before you can continue with the action. You can delete an action from your hand (say, when the enemy doesn't use a damage type for one of your block cards) with "Discard". When you play an action by selecting "Play Action", your turn ends, and the enemy will continue with the next move in the current moveGroup, or randomly select a new moveGroup. If you end your turn within a threatened tile without blocking its damage type, you are sent to the game over screen. Striking has priority over being killed by the enemy. If you successfully kill the enemy, a new battle will start shortly with a new random enemy.

Outcome Showcase:

Currently, not all planned features are implemented; however, what currently exists is enough to satisfy the project requirements. Namely, the "deckbuilder" feature is not present; after every successful battle, the player should be able to pick from a random set of actions to add to their own total deck. Battles should be selectable from a battle map scene, increasing in difficulty; at the moment, it is completely random between a Slime and a Goblin. And finally, of course, more actions and enemies need to be added.

Title Screen:

Graphical user interface

Description automatically generated

Game Screen:

A picture containing timeline

Description automatically generated

Selecting an action to use: A picture containing timeline

Description automatically generated

Game Over screen:

A picture containing text

Description automatically generated

Conclusion:

The final product satisfies the project, but does not satisfy me. The game is clearly not done, and there is still much more to be added before it can be called a proper product. The code is very messy as well, and can stand to be cleaned up and refactored; right now, it places everything in-memory. It should be putting everything on the heap and using pointers, to avoid pointless duplication of data.

This project definitely helped me learn how to make a moderate, enjoyable game, and I will be able to expand upon this idea in the feature to create my vision.