Game Plan + Description

Description:

Our game is called Treasure Hunter, and it will be a 2D, maze-navigation game. The player will control the main character, navigating the maze and collecting key items while avoiding enemies. The game follows a brave but greedy scuba diver as he goes on the hunt for sunken treasure at the bottom of the ocean. The game begins as the diver enters an underwater cave filled with treasure. Lurking within this cave are deadly sharks that stalk the diver and tangled clusters of seaweed that cause them to drop their precious treasure. Appearing sporadically within the cave are large treasure chests that the diver can collect for a hefty bonus to their total treasure collection. Once the diver acquires all of the regular pieces of treasure, the cave exit becomes accessible, and the diver can escape with their loot. If the diver gets caught by a shark, the game is over.

Plan:

Phase 1: Our plan for creating this game will begin with outlining the various Classes, Interfaces, and Relationships needed to deliver a playable and stable experience. This phase includes creating UML and use case diagrams, UI drafts, and asset acquisition/creation. These plans/diagrams will lay the foundations for our eventual implementation and development of the game. The specifications laid out in this phase may be altered/strayed from as development takes place.

Phase 2: The following phase will involve implementation incorporating the design plans laid out in phase 1. Namely, creating a Map driver/engine to constantly monitor and update the game state depending on the elements' placement on the map. Additionally, creating a pathing algorithm for the moving enemies (sharks) in order for them to follow the player character and create challenging gameplay. The requirements/goals of this phase may change as the implementation progresses.

Phase 3: The third phase of the project will consist of testing the systems we implement in phase 2 to ensure a stable experience for the user. This will include specific scenario/situational testing to find and remove bugs that could affect the desired flow of gameplay or cause serious failures within the application.

Phase 4: Depending on the completion speed of the previous phases, we may attempt to implement a few more features to add robustness to the game. These could include new difficulty settings and/or new map layouts/designs. We may also attempt to create a trailer for our game in order to present it at the end of the semester.