Shark Game.JACK

Concept/Idea of Your App

The main goal in **Shark game** is to eat as many fish as you can.
The player controls the shark, which goes up or down, while fish keep appearing, swimming towards the shark. He needs to avoid a hazard in order to survive.

Architecture

1. **Main.jack**
    a. **Purpose**: Responsible for initializing the game instance and running the game.
    b. **Main Logic** Sets up the shark run game instance and runs it.
2. **SharkRunGame.jack**
    a. **Purpose**: The central controller for game logic, integrating all components into a cohesive gameplay experience.
    b. **Main Logic**:
        i. **Game State Management**: Tracks the game's current state, including current score, highest score, game over conditions and speed.
        ii. **Object Management**: Handles spawning, updating, and removing fish and killFish objects.
        iii. **Input Handling**: Processes keyboard input for controlling the shark, increasing/decreasing game speed, cheat codes and restarting the game.
        iv. **Display Updates**: Updates the score, high score, and other on-screen elements in real-time.
        v. **Game Loop**: Runs the primary loop to update all elements and render them at a consistent frame rate.
3. **Fish.jack**
    a. **Purpose**: Creates a Fish instance.
    b. **Handles Movement, drawing and erasing**: Fish move horizontally across the screen at varying speeds.
4. **KillFish.jack**
    a. **Purpose**: Creates an instance of a danger fish the kills the shark, and loses the game.
    b. **Main Logic**: Simillar to Fish.jack with a different drawing function and movement logic. Handled differently in game instance.
5. **Shark.jack**
    a. **Purpose**: Creates and handles the shark's movement and interactions with the environment.
    b. **Main Logic**:
        i. **Movement**: Includes methods for moving the shark up and down based on user input, ensuring it stays within screen boundaries.
        ii. **Collision Detection**: Determines whether the shark has collided with regular fish (to gain points) or KillFish (to end the game).
        iii. **Rendering**: Draws the shark on the screen and erases it during position updates or when a collision occurs
6. **Utility.jack**
    a. **Purpose**: A helper function with an important method that tranlates (x,y) location on the IDE screen to the location variable, used by BitMap Editor functions.
    Basically simplifies the drawing location of objects.

7. **LFSR32Rand.jack**
    a. **Purpose**: Generates random numbers for dynamic game elements like fish positioning and movement speeds.
    b. **Main Logic**:
        i. **A file with Copyright (C) 2016, Mark A. Armbrust –** handles randomness in game using a Linear Feedback Shift Register (LFSR) algorithm to produce pseudo-random numbers.

Motivation:

Our motivation for the game was to create a fun yet challenging game, while learning the Jack language and creating a complicated multi-file program.

We have 3 different ideas, one was some run game but we realized it was too generic, then another idea was some childhood game "The Miner" but we had problems with the art aspect of the game.

While developing "The Miner", we stopped mid way and decided to go with the Shark Game, which was kind of a mish mash of the previous 2 ideas.

Link to Video: Video Link

Names and emails:

Omer Dan, omer.dan01@post.runi.ac.il

Amit Kaminsky, amit.kaminsky@post.runi.ac.il

Images from the game: