**Fish Game**

**Revision History**

|  |  |
| --- | --- |
| **Version** | **Description** |
| 1.0 | Initial Creation |

**Development Environment**

**Game Engine:**

A Custom engine is being used built on top of Raylib.

**IDE:**

Visual Studio

**Source Control:**

Github via Sourcetree.

**Third Party libraries:**

Raylib, TinyXml, ImGui

**Other Software:**

Aseprite for 2D art, and Fl Studio for audio.

**Game Overview**

Technical Goals:

I wish the create a 2D game that’s fairly challenging.

Game Objects and Logic:

|  |  |
| --- | --- |
| **Player** | Move around the map, collects **Pearls** and avoids **Sharks**. Controlled by player. |
| **Pearls** | Placed inside a **Clam**, collected by the **Player** |
| **Clams** | Holds a **pearl**. |
| **Sharks** | Move around the map, attacking the **Player** if they are in view. |
|  |  |

**Game Flow**

* Main Menu
  + Has a start button, demo button and editor button, Navigated by the arrow keys.
* In Game
  + **Player** moves around a cave.
  + **Player** tries to locate **Pearls**.
  + **Sharks**, using various movement patterns patrol for the **player**
  + When a **Shark** sees a **player** the attack it, using varying attack patterns.
  + When the **player** collects all the **Pearls** they win.

**Mechanics**

The Players main goal is to avoid **Sharks** and collect **Pearls**, The **Player** moves around using the arrow keys an collects **Pearls** by moving into them. Once the **Player** has been ***hit 3 times*** the game is over and they **Player** has lost. To win the game the **Player** must collect all **Pearls** in a level.

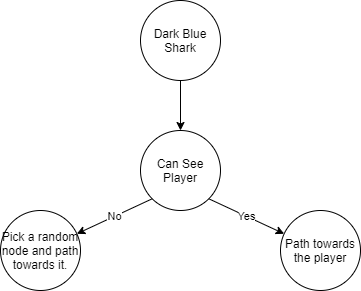
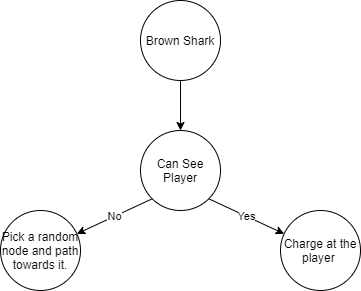
**Graphics**

* Side view.
* 320x180 Upscaled.
* Pixel art.

**Artificial Intelligence**

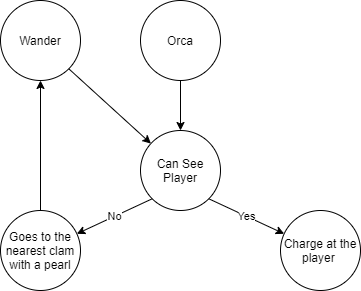
AI in the game uses A\* to find the shortest path between nodes, this is only used in some Behaviours though. A Simple state machine will be used to control the **Shark’s** logic, Since the **Player** is user controlled it won’t use any AI.

**Brow Shark**

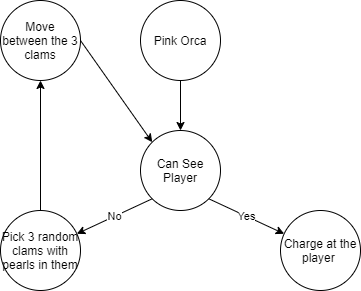
****

**Dark Blue Shark**

**Orca**

****

**Pink Orca**

****

**Physics**

No third-party libraries are being used to handle physics / collision. Collision is handled using simple AABB detection. Collisions with tiles is done by check if the target position is touching a tile and the looping to that position by 1 pixel until the player is right up against the wall, creating pixel perfect collision detection.

**Items**

Pearls serve as the goal for the game but have no use other than being collected and serving for markers for some **Shark** AI’s.

**Game Flow**

The game will have a singular medium to large level, levels are loaded through .xml files using TinyXml2, Levels are constructed using my own level editor built for the game (Changing m\_editor to true in game.h should access it.)

**Objectives**

The **Player** must collect all **Pearls** to win the game.

**Interface**

**Menu**

Main menu will be controlled with arrow keys and Z or Enter to select. The appearance of the main menu is simply the games title centred at the top of the screen with the options below them.

**Camera (Orthographic)**

The camera follows the player. At the end of a frame the 320\*180 game scene is rendered to a render texture and then scaled up to fit the window.

**Controls**

Arrow keys to move and navigate menus, Z or enter to select in menus.

**Asset** **List**

* Various Shark sprites
* Player fish sprite
* Sand tileset, Collision tileset
* Background music (mus\_water)
* Various room files.

**Technical Risks**

The main potential problem is that I spend way to much time working on the base engine and or the editor that I waste to much time not working on the game.