Report

Problem & Solution

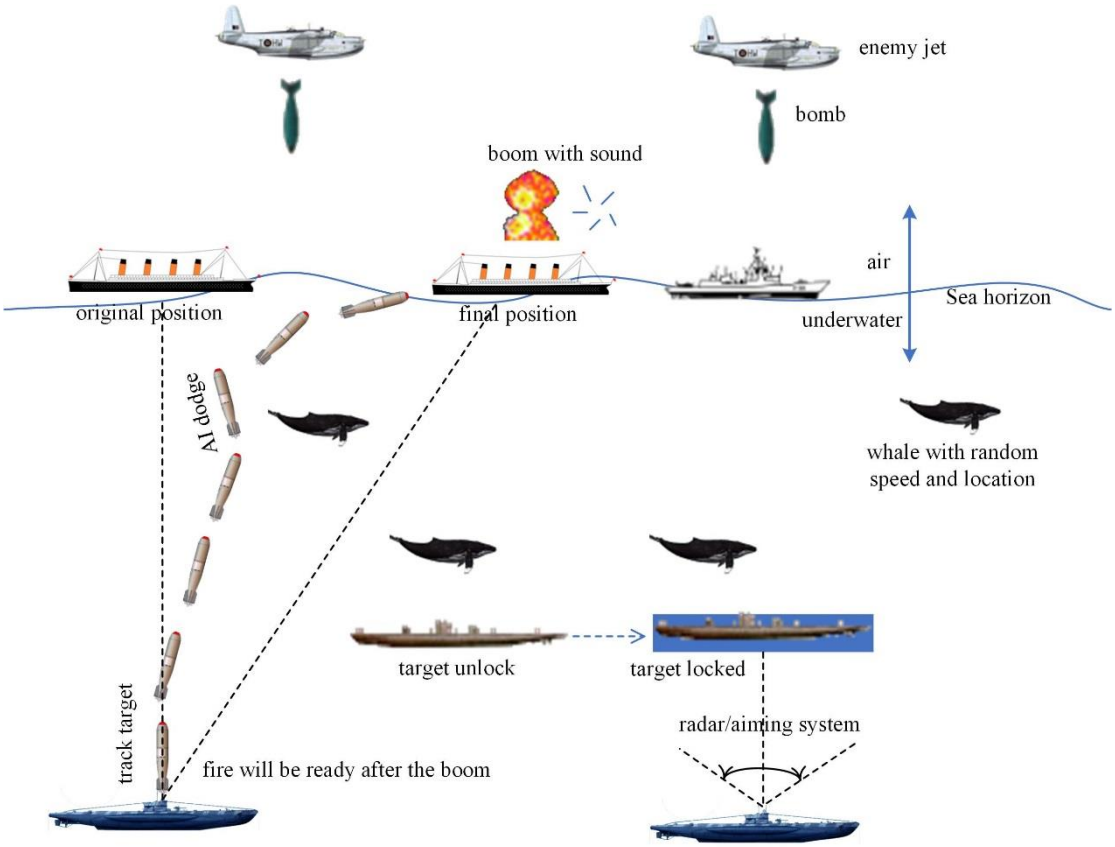
- The Aim system and AI dodge

I designed the submarine to have a radar or aiming system where the player can control the torpedo direction. Once there is an intersection between the aiming line and enemies, the target will be locked. The location of the torpedo destination will be updated accordingly. The AI potential function was adopted, ensuring the torpedo will dodge the whales when whales are too close.

- Whale Animation

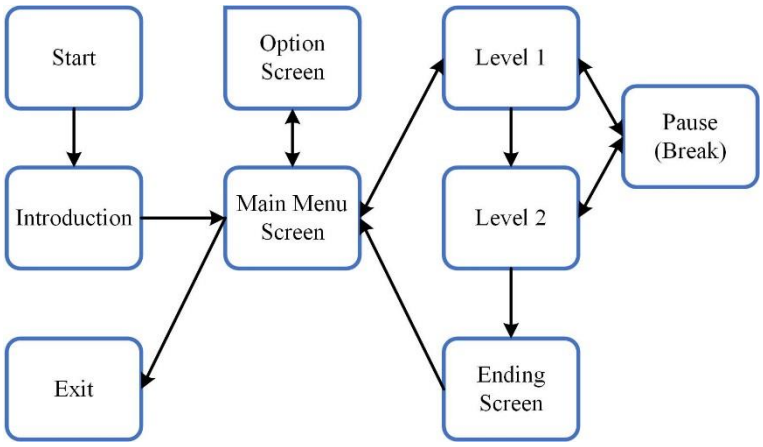
After researching on the open game art website, I could not find any animation suitable for my game. I reached out to any videos and gifs. One animation video was chosen. I extract 21 pictures from the video, each of which was used for one frame in my animation. All pictures were adjusted, resized, edited, and finally grouped together before using Sprite method for animation.

Story Board



The above figure shows the details of the game story, implying how the AI technology, tracking system, and radar system are employed. All the art works are also demonstrated.

Layering Strategy



After running this game, there will be an introduction screen, explaining this game's background and environment. The user can open the option screen for game instruction. After pushing the "Play" button, the player will enter the game Level one. If the player can pass the criterion, the player will be allowed to play the Level two game. When playing the game, "Pause" button is available to pause the game and later on resume back to the game. Succeed or not, the player will enter the ending screen after the game Level two.

Artwork

There are two sounds incorporated into this game. One explosion sound which will be activated after the torpedo hit the enemy targets. The other sound is a warning sound, which will be accompanied by a flash text, saying "You Are Spotted!", also indicating the entrance from Game Level One to Game Level Two.

Extra Coding Technique

I used one AI algorithm to successfully design the torpedo in the game for dodging the whales. Part of the codes are shown as

```
Vector2 pp = ss.getPos();
if ((pp - beePos).Length() < pushBackDist)
{
    Vector2 pb = (pp - beePos);
    float dist = pb.Length();
    float amt = (pushBackDist - dist) / pushBackDist;
    amt = amt * pushBackStrength;// amount of push back (checked with debugger)
    pb = pb / dist;
    pushBack = pb * (-1);

}
```

I used XACT to manage the sound. The original .wav files were transformed into XACT files, namely .xsb, .xgs, and .xwb files by using tools of AudConsole3.exe and Xact3.exe. They are loaded into game by

```
// Animation of Explosion & Sound Effect
audioEngine = new AudioEngine("Content/sound.xgs");
soundBank = new SoundBank(audioEngine, "Content/Sound Bank.xsb");
waveBank = new WaveBank(audioEngine, "Content/Wave Bank.xwb");
soundBank.GetCue("explosion").Play();
```

The original codes and related tools for the XACT method are explained at https://www.gamefromscratch.com/post/2015/07/25/MonoGame-Tutorial-Audio.aspx