**Shark Bait** (Ooh-Ha-Ha)

Game Design Document

Ben Hamilton

**Overview**

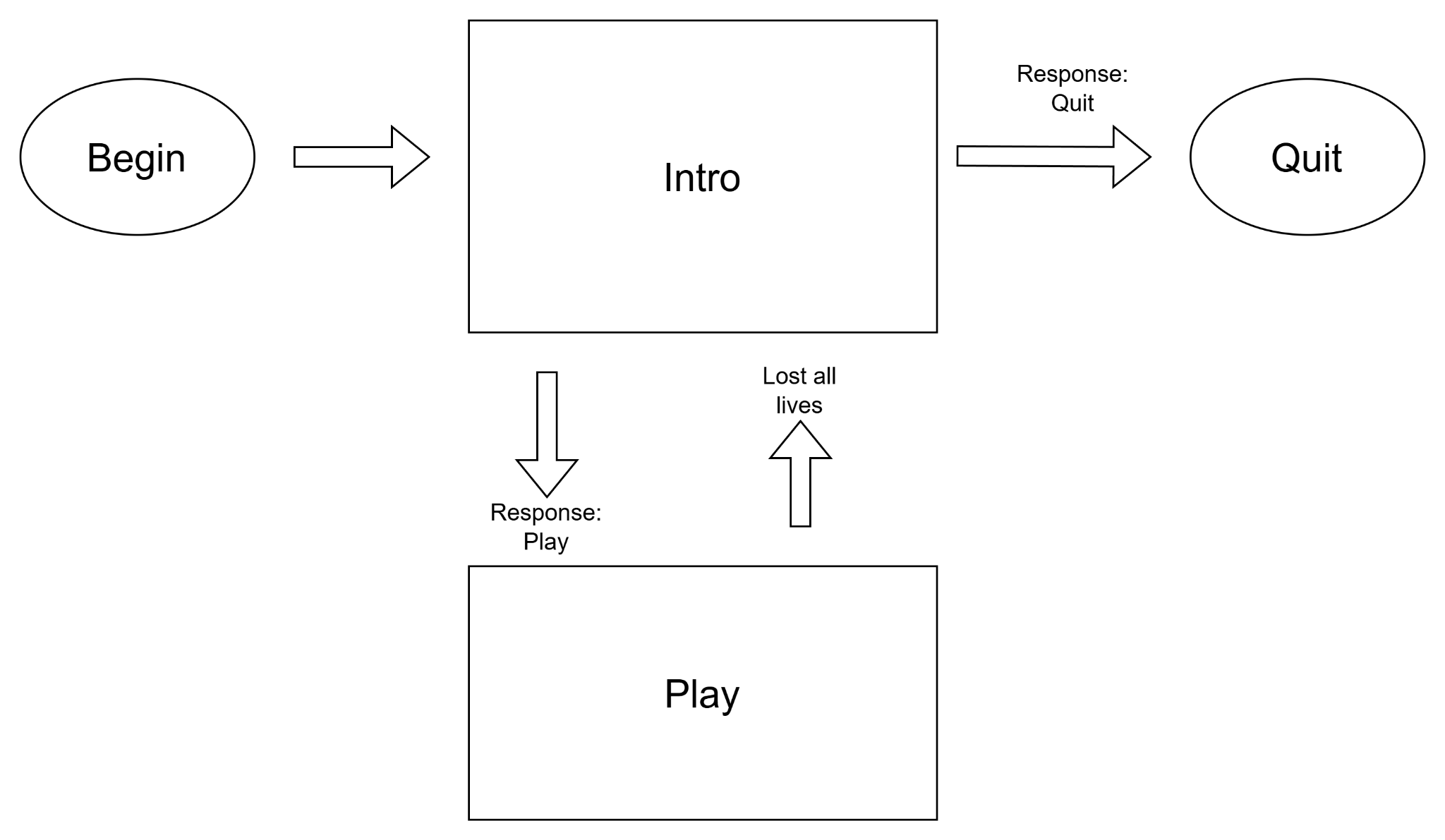
Shark Bait is a 2D arcade game built in python using pygame and simpleGE.

In this game the player will play as a shark that swims across the bottom of the screen eating fish that are falling from the sky (presumably by fishermen, but maybe it’s rapture who knows, this is a game design document not a lore bible.) The shark will move with left and right arrow button presses in the corresponding direction. The fish will fall from random points at the top of the screen at variable speeds, 3-8 pixels per second, and will make a crunch sound when they contact the shark. Otherwise, they will reset by hitting the water and fall from a new position. There will also be dynamite that when hit by a shark will take one life. Upon losing 3 lives, the game ends.

The game will begin with an intro screen that will let you play the game or quit the game and has a scoreboard to see the best run.

Upon the game ending, the player’s score will be checked against the best score, which will be modified if the new score is better.

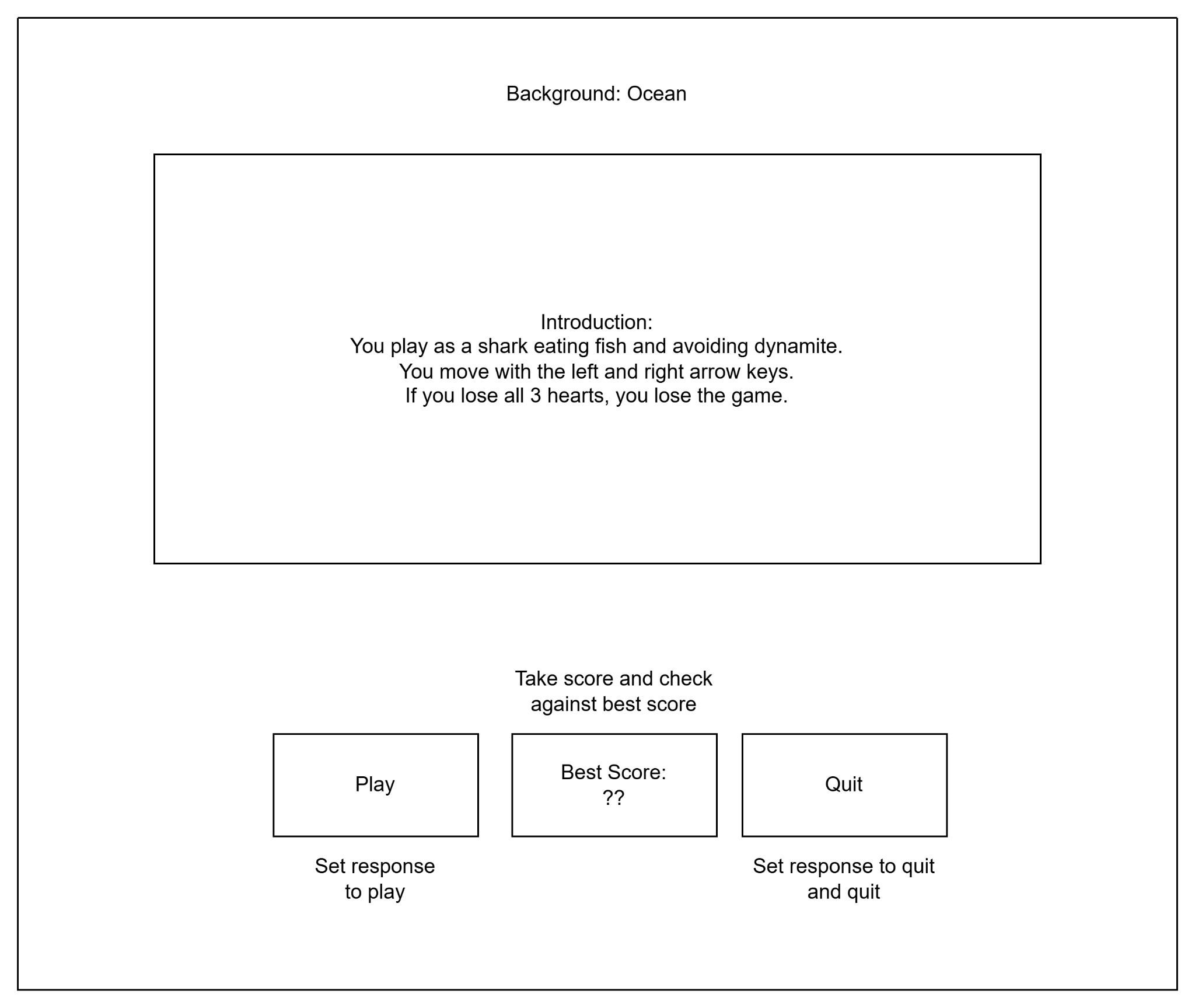
**State Transition Diagram:**



This game has two states: intro and play. These will be two subclasses in simpleGE’s scene class. There will be 2 buttons and a panel of instructions for the player to follow. If the player clicks on the play button this will set a response variable to play and take the player to the play scene. If the player selects quit, the variable will be set to quit and kill the game.

The play scene ends when the player loses all three of their hearts. This will send the player back to the intro screen with a new score that will be checked against the high score to determine if there is a new high score.

**Instructions Screen:**



This scene’s elements:

1. Instructions: a simpleGE multilabel detailing how to play the game and the rules
2. Best Score: a label that displays the best score
3. btnPlay: A button that says play
4. btnQuit: A button that says quit

Attributes:

1. bestScore: an integer indicating the best score, pulled from a save file, passed into the class initializer and displayed as best score
2. A string indicating player choice, determined by one of two button options.

**Create Save and Load Function**:

Define saveScore function with bestScore argument

Create or open score.json

Put the value of bestScore into the json

Close the json

Confirm the number was saved

Define loadScore function, no argument

Open or create score.json

Pull out the value within and give to bestScore

Close json

Return bestScore

Define updateBestScore, with newScore argument

best calls loadScore()

If newScore is greater than best

Best gets newScore

Save best

Return best

Initializer creates attributes:

Initiate with score argument

Set image to campus

Set response to “Play”

Create instructions multilabel

Add instruction text

Set instructions size and center (320, 240) and (500, 250)

Copy score to bestScore parameter

Create lblScore

Set text to “Best Score: {bestScore}”

Set center to (320, 400)

Create btnPlay

Set text to “play”

Set center to (100, 400)

Create btnQuit

Set text to “Quit”  
 Set center to (550, 400)

Add lblInstructions, lblScore, btnQuit, and btnPlay to sprites

Now event handling for instructions

process():

If quit button is pressed:

Set response to “Quit”

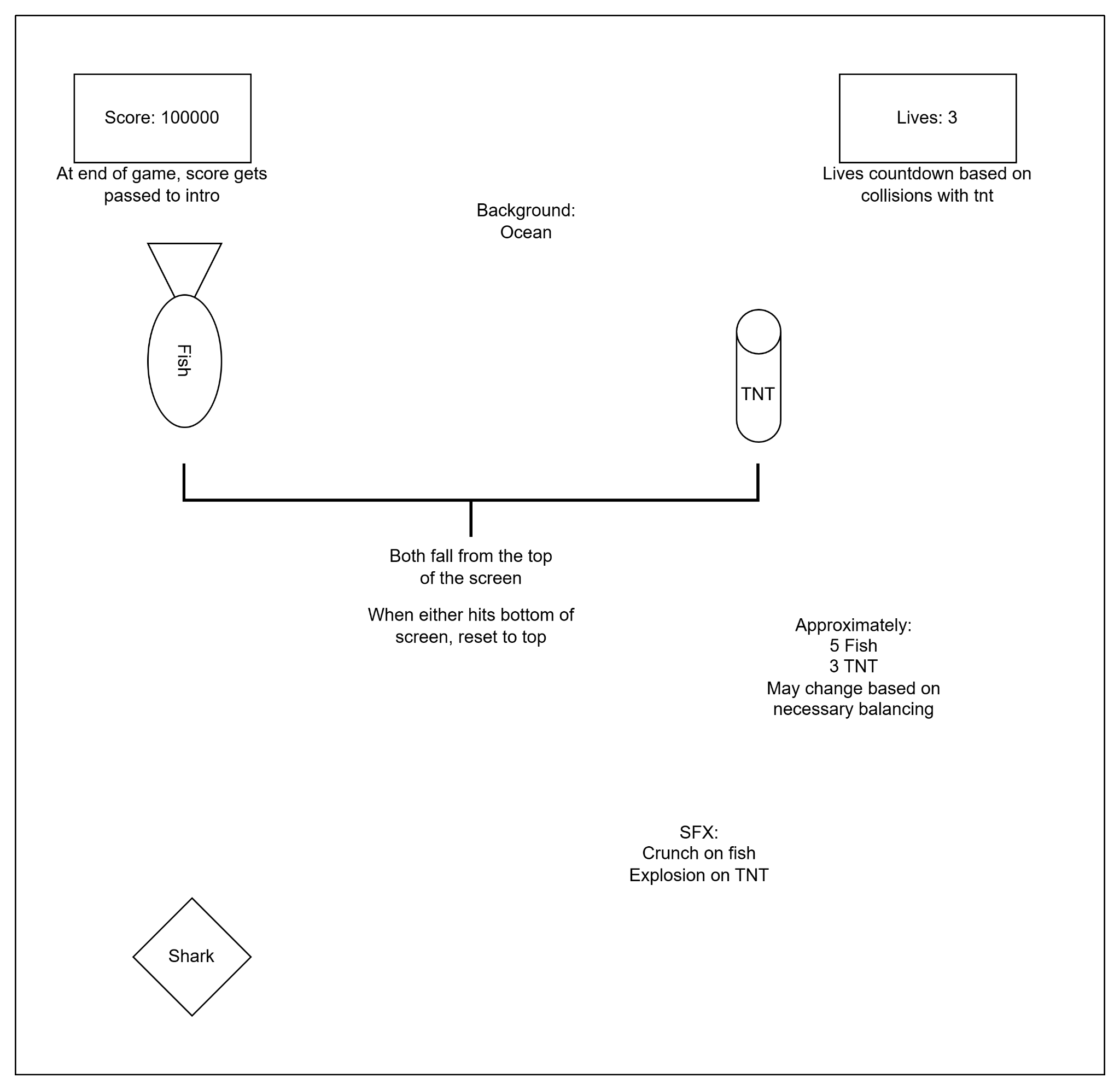
Stop the scene

If play button is pressed:

Set response to “Play”

Stop the scene

**The Game scene:**



The game has 5 visual attributes:

1. Shark - an instance of the shark class
2. Fish - an instance of the fish class
3. TNT - an instance of the bomb class (basically just the fish class)
4. lblScore - an instance of the LblScore class
5. LblHealth - an instance of the LblHealth class

It also has some non-sprites:

1. Score - an int containing the current score
2. Health - an int containing the current health level
3. sndFish - a crunch sound from the simpleGE.Sound class
4. sndBomb - an explosion sound from the simpleGE.Sound class

Initialize the game state:

Def init(self)

Set image to Ocean.png

Set hearts to 3

Set score to 0

Initialize sndFish to crunch sound

Initialize sndBomb to explosion sound

Create instance of shark

Create a list of 5 fish instances

Create a list of 3 tnt instances

Create an instance of LblScore

Create an instance of LblHearts

Add shark, bomb, fish, lblscore, and lblhearts to sprites

The events for game:

Process:

For each fish in the fish list

If the fish collides with shark

Play the crunch sound

Reset the fish

Add one to score

Update LblScore

Do the same for bombs but subtract one score and update LblHearts

If hearts = 0

Print final score

newScore gets the final score

Self stop

**Let’s talk components of game:**

**Shark**

Subclass of SimpleGE.Sprite

CC shark png

Size is 50x50

Transparent background

Movement speed is 5

Init:

Set image Shark.png

Size 50x50

Set position (320, 400)

Set moveSpeed to 5

Process:

If left key is pressed

Move left

If right key is pressed

Move right

**Fish:**

Subclass of simpleGE.Sprite

CC Fish png

Transparent background

Resets if it hits the bottom of the screen or shark

Falls down screen at 3-8 pixels per second

Init:

Set image Fish.png

Set size 25x25

Reset

Reset:

Set y to 10

Set x to random integer between zero and screen width

Set dy to random integer between 3 and 8

checkBounds:

If bottom of sprite is larger than screen width:

Reset

**Bomb:**

Everything is identical to Fish subclass. The only difference is at a game level so I’m going to assume you can figure out the algorithm for this.

**LblScore**

Subclass of simpleGE.Label

Init:

Set text to “Score: 0”

Set center to (100, 30)

**LblHearts**

Same as LblScore

Init:

Set text to “Lives: 3”

Set center to (500, 30)

Main function:

The main loops that controls the whole thing

Includes the following variables:

1. Instructions - instance of instructions class
2. Game - instance of game class
3. keepGoing - the GOAT
4. newScore - current score
5. Best - the best score

**Main:**

Set keepgoing to true

Set newScore to 0

Call loadScore for best

Start while looping:

Instructions = Instructions(best)

Start instructions

If instructions.response is “play”

Game gets Game()

Game start

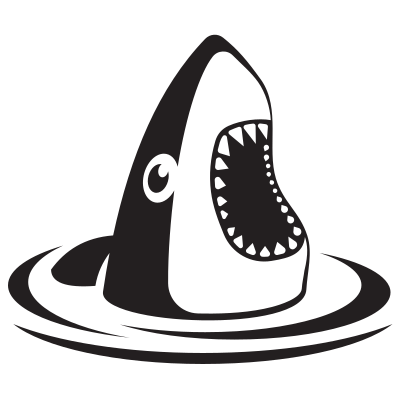
newScore gets game.score

Best calls updateBestScore on newScore

Otherwise:

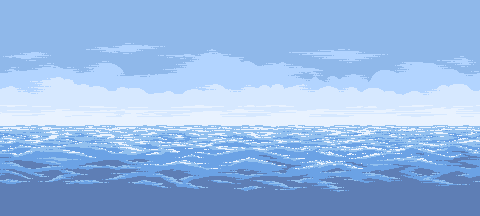
keepGoing is false

**The assets I used:**



Surprised Shark by liftarn (Creative Commons)

<https://openclipart.org/detail/319499/surprised-shark>



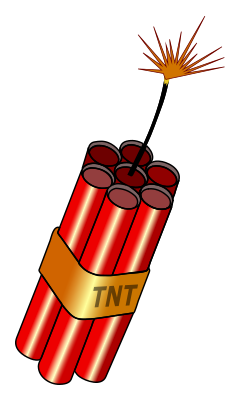
Ocean Background by KnoblePersona (creative commons)

<https://opengameart.org/content/ocean-background>



fish by johnny\_automatic (creative commons)

<https://openclipart.org/detail/518/fish>



Lit Dynamite by j4p4n (creative commons)

<https://openclipart.org/detail/336201/lit-dynamite>

Big Crunch 2

Connersaw8 (freesound)

<https://pixabay.com/sound-effects/big-crunch-2-90138/>

Explosion

Ahmed\_Abdulaal (free for use)

<https://pixabay.com/sound-effects/explosion-312361/>