**Project Two Game Outline**

**Ben Weller and Aaron Beyer**

**CSCI 313**

**4/3/2017**

**Updated Game Specification**

**Story:**

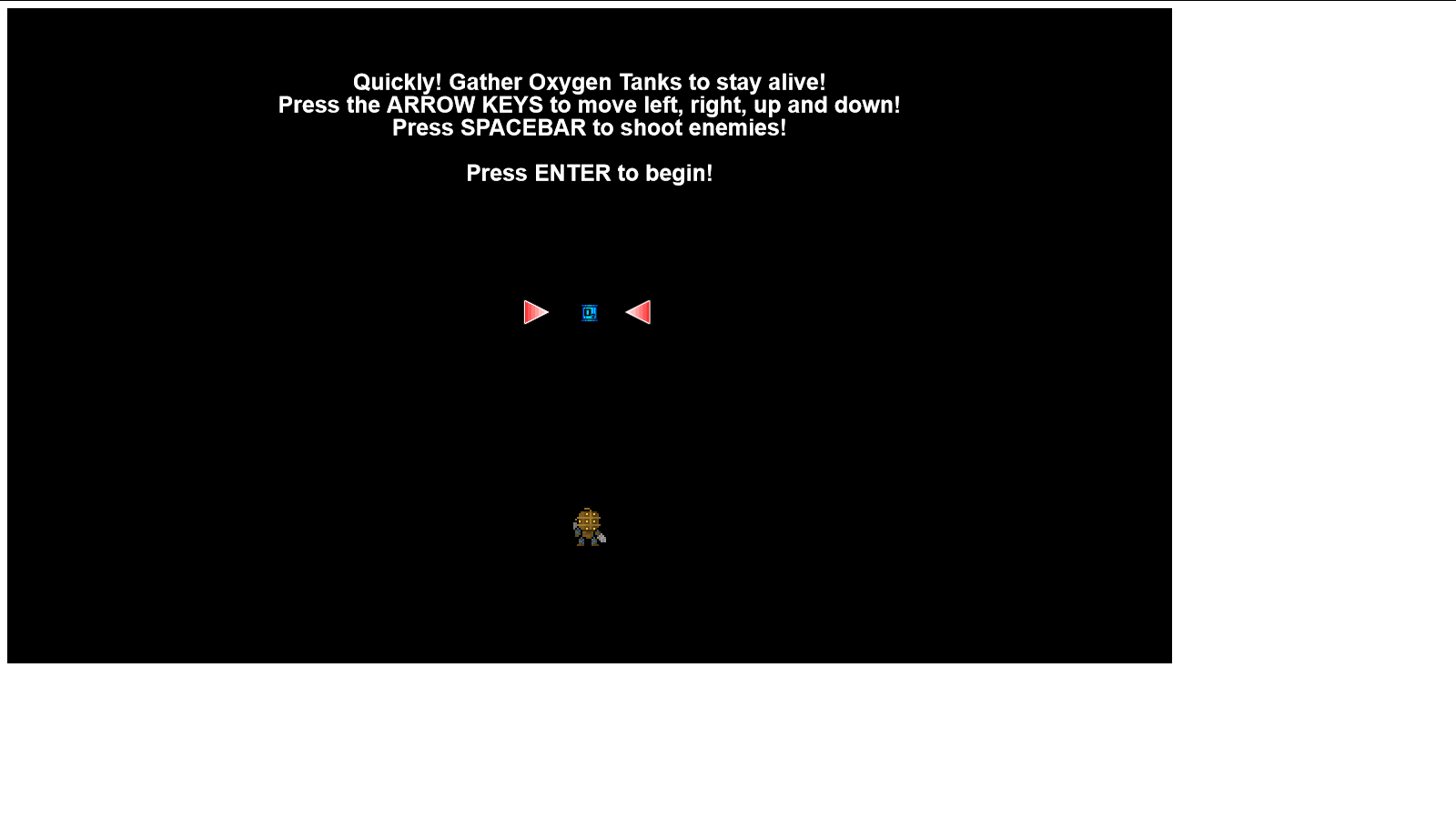
You are a scuba diver in peril, as you drain the last bit of oxygen from your tank you realize that you have enemies surrounding you, and that you need to find more oxygen and fast! With limited charges in your blaster you have to navigate your way through the series of platforms trying to find your next oxygen fix, all while avoiding enemies.

**Scene:**

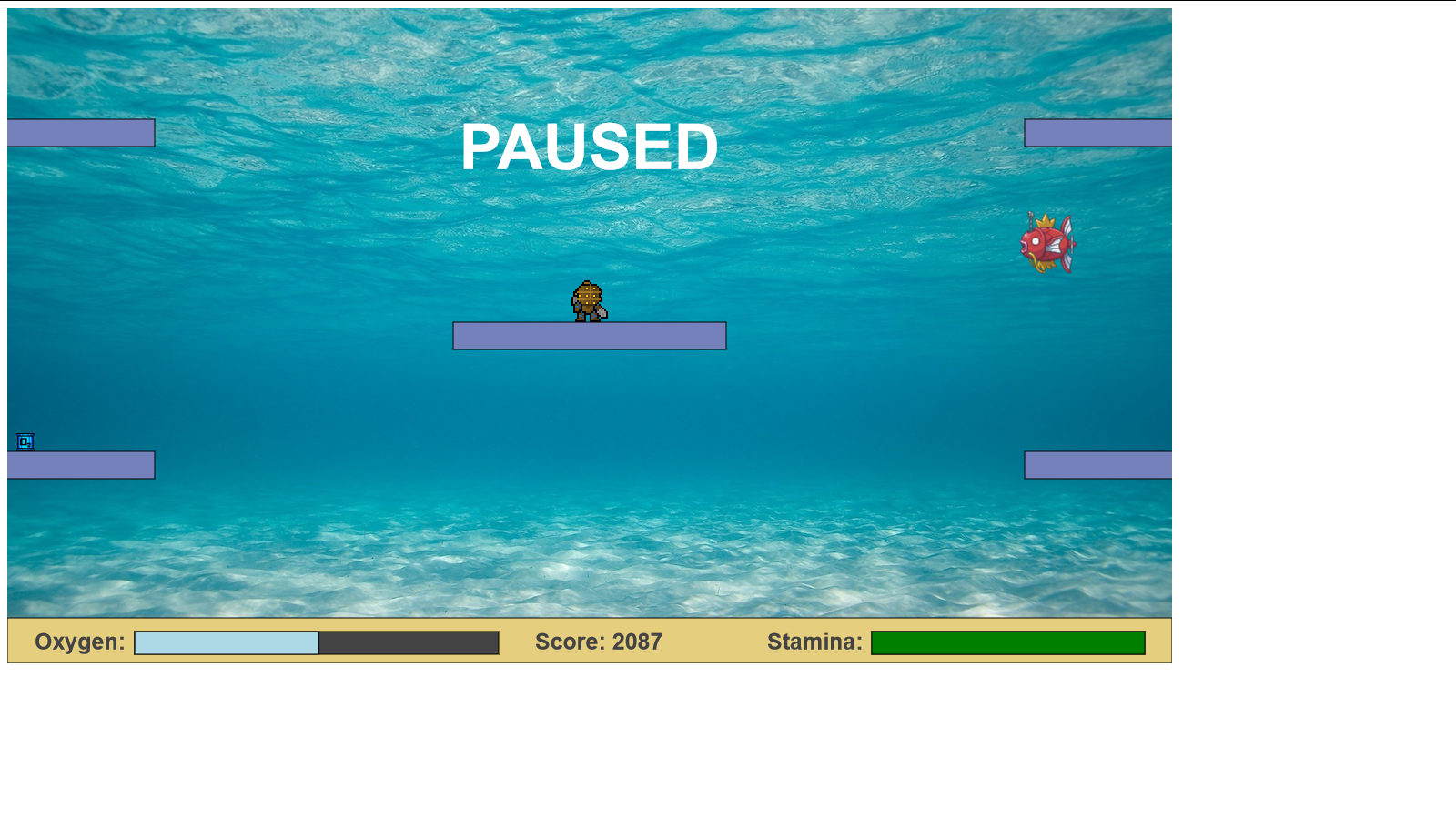
The scene is placed underwater, there is a “sandy” bottom and five platforms located in the arena. There is one center platform and two platforms that wrap around each side of the screen.

**Gameplay:**

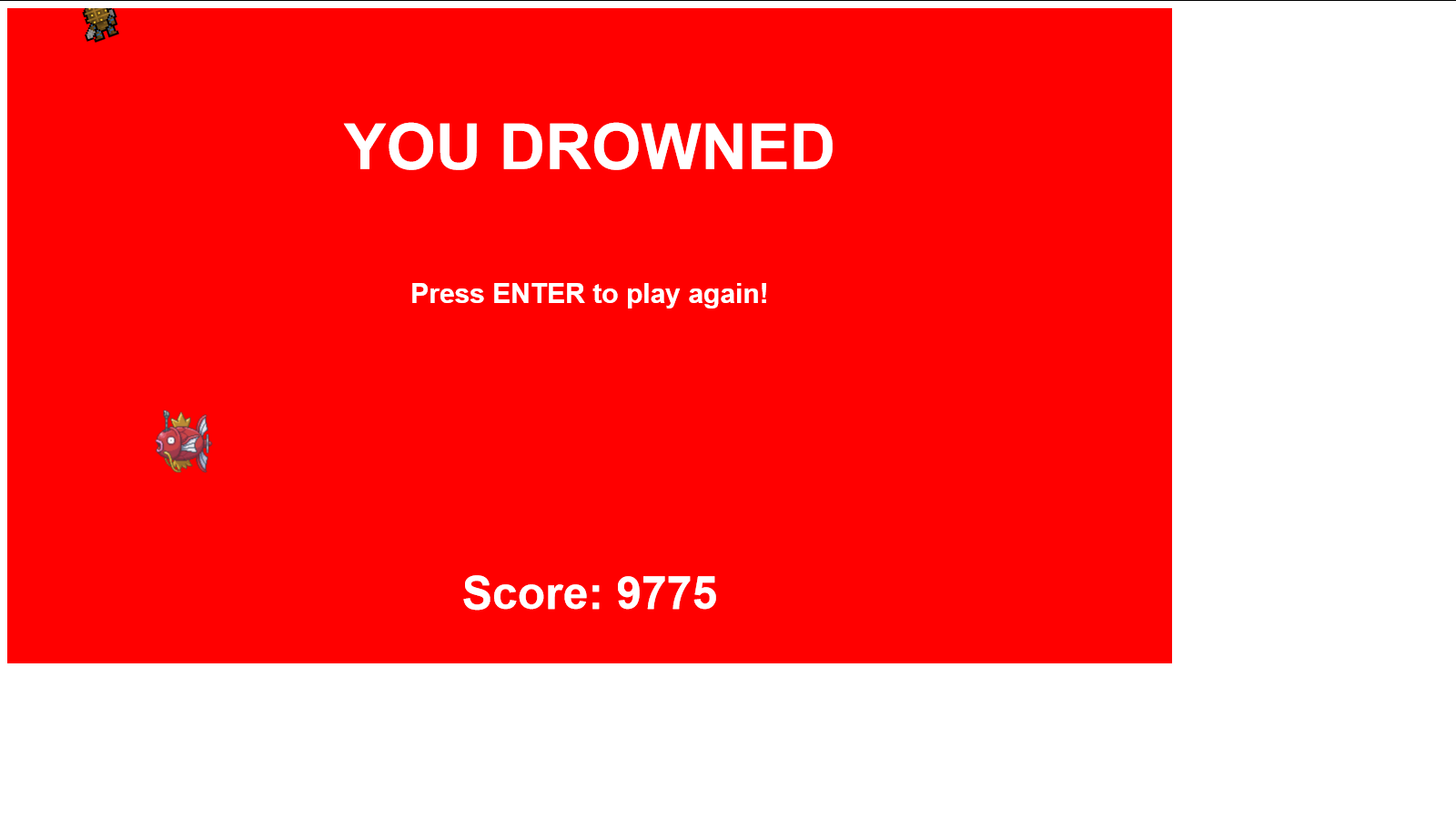
**Intro Screen:**



**General Game Example:**



**Death Screen:**



The game is played using the WASD/arrow keys for movement. To pause the game use the enter key, and for firing bullets use the spacebar to fire a bullet in the direction the diver is facing.

    The diver can wrap around the edges of the screen. He can go up through the bottom of a platform but will stand on a platform unless your direct him to go down through it. The diver can’t go through the “sandy” bottom and appear on the top of the screen, nor go through the top of the screen and appear on the bottom.

    Enemies appear on the right side of the screen and swim to the left. Once in a while there may be a mass production of enemies that form a wall blocking your progress.

    You can shoot and remove enemies using the space bar. The diver will shoot whichever way he is facing. You can only shoot if you have an appropriate amount of stamina in the bar, if you deplete the bar, it will take longer to regenerate.

    When you collect a tank your oxygen refills. As you collect more tanks your oxygen depletion rate increases, and your score that you get per tank also increases. When you collect a tank it is moved to a different random location among seven picked.

    If you can’t collect an oxygen tank in time the drowning bar will increase, the game “slows down”. When you collect an oxygen tank after drowning it only refills half way. The drowning bar slowly drops.

    The longer you stay alive the more of a score increase you get.

**Differences:**

Our game is different in that doesn't hide the 8-bit feeling. We make no effort to hide our game behind taxing graphics or overly complex gameplay. This is a simple what you see is what you get type game. There is no complex mouse following behavior and basically all instructions are given in one simple screen readout at the beginning. It is intended to become difficult very quickly and as such doesn’t lead to boredom during long drawn out “lives”.

**Major Topics:**

**Bitmaps:** The tank and diver are bitmaps displayed on the screen.

**Shapes:**    The oxygen and stamina bar are both shapes that are modified using command objects.

**Animations:** The oxygen depletion and stamina regeneration are both examples of animation.

**User Input:** The user uses the keyboard to move the character around the screen to shoot and to pause the game (WASD/Arrow keys, Spacebar, and Enter respectively).

**Containers:** The bullets the diver shoots are containers. Additionally the enemies are in containers and as such are easily removed when hit.

**Sprite Animations:** The magikarp that swim across the screen are examples of sprite animation.

**Differences in Initial to Final Product:**

**Beginning:**

    The game was initially thought up to be a swimming game where it simply got to be so difficult that there would be no room for error and as such you had to focus strictly on the game.

**Changes:**

The screen size was initially 800x600. During the second week of the project it was proposed to make the screen bigger and thus change the gameplay. This meant that platforms would have to be moved, tank spawns would have to be changed, the opening and closing screen would be modified and the oxygen and score indicators would have to be relocated. This change was implemented because the screen was just a bit too small and without the expansion it wouldn’t have had room for enemies.

The thought of enemies was also pitched in the second week and the game radically changed. Then we had to add to include collisions between the enemy and the player, and we would have to include some form of protection for the player (thus the advent of bullets). This change was implemented to further enhance the game (it felt a little bit plain without it).

**Features:**

|  |  |  |
| --- | --- | --- |
| Task: | Name: | Time Implementation |
| "Drowning" bar | Aaron | Old |
| Implement Enemies | Aaron | Old |
| Collisions with Enemies | Aaron | Old |
| Increased canvas size to 1280x720 | Ben | Old |
| Shooting | Aaron | Old |
| Score Logic | Aaron | Old |
| Collisions with Oxygen tank | Ben | Old |
| Oxygen Tank Respawn and Placement | Ben | Old |
| Game over/game opening sequence | Aaron | Old |
| Platforms | Aaron | Old |
| Progress Reports | Ben, Aaron | Old/New |
| Implement Power-ups | Ben | New |
| Refactor/redesign code | Aaron, Ben | New |
| Possibly change sprites/bitmaps | Aaron, Ben | New |

**Code Table of Contents:**

**High Level Outline:**

Load()

Init()

game\_build()

set\_controls()

game\_start()

addEventListener(“tick”, game\_step)

game\_step()

check\_controls()

handleKeyDown(e) –Handles key events

handleKeyUp(e) –Handles key events

apply\_gravity()

onPlatform(p) –Gravity

add\_enemies()

createFish(Normal)/createFish(Normal) –Creates enemies

change\_oxygen\_and\_stamina()

oxygenBarLogic()

drowningLogic() –Does the downing logic

isGameOver() –Determines if the game is over

gameOver()

resetGame()

check\_collisions

checkBulletCollisions(index)

checkFishCollision(index) –Removes the fish if it’s been hit

createBullet() –Bullet Logic

getRandomColor()

checkTankCollision() –Checks collisions with the tank

genericCollisionMethod(arguments)

movesTank() –Moves the tank

powerUpLogic()

removeFromPowerupArray(array)

createPowerUp()

powerUpCollisions()

genericCollisionMethod(arguments)