CS 409 - Assignment 6
Tuesday April 8, 2014
Tanner Hoffman
200286645

New Features

For this assignment I chose to rebuild the game from scratch in Unity3d Engine (www.unity3d.com). While having a lot of initial overhead to get the controls and general functionality of the game working using Unity made it much easier to add various new features. New graphical, sound and GUI features make the game feel more polish and lead to a better user experience. The way fish are captured has also changed to add more excitement. New features include:

Graphical

- Terrain built and textured with terrain editor
- 3D grass patches scattered on the ocean floor
- Rock models scattered around the ocean floor
- Skybox
- Ability to breach the top of the water
  - render setting change when the submarine breaches the surface remove the blue fog used to create the illusion of being underwater
- New submarine model
  - flashing red light
  - animated propellers (idle and acceleration animations)
- New fish model
- Particle effects
  - sand particles when hitting the ocean floor/walls
  - bubble particles emitting from propellers (small amounts during idle and larger amounts during acceleration)
  - sparkle particles when capturing a fish
- 3D lighting
  - there is a main light source positioned in the world that creates subtle shadows and lighting on surfaces
  - secondary “fill light” is placed facing the opposite direction of the main light to make the shadows less harsh and avoid the scene being too dark

Sound

- Background music
- Terrain collision sound (hard to hear without good speakers/headphones)
- Shooting Sound
- Engine Sound
  - the pitch of the engine sample is increased/decreased when accelerating/deceleration to create the illusion of the engine running faster
- Fish capture sound

GUI

- Button to toggle mute
- Button to toggle fullscreen
- Game Over screen

Gameplay

- The game generally controls better due to the Unity's physics engine
- Collision's for the Submarine are done with the model mesh rather than a sphere or box collider
- Shooting:
  - the player can now shoot "net balls" to catch fish
  - this is the only way to catch fish now

Objective/Manual

The objective of the game is to capture a predetermined amount of fish. For testing purposes I kept the number to capture low, at 25. I also turned down the speed of the fish slightly to make testing easier. Capturing fish is no longer done by running into the fish. The controls for the submarine are the same as the original game. By pressing the space-bar the user can shoot "net-balls" that will capture a fish on contact. The net-balls move fairly slow so it is usually best to lead the fish when aiming.

Major Feature #1: Shooting to Capture Fish

The shooting feature makes use of a feature of Unity known as *prefabs*. A prefab can be made for any object that is to be reused multiple times in the game. When you create a prefab of an object you can reuse it anywhere in the game with all of the scripts, attributes, sounds etc. in tact. For the shooting feature i made a net-ball prefab. The prefab is a sphere-like object that includes a texture, collision detection, a destructor script, a fish capture script, a particle effect and a sound source.

The submarine has a *shoot* script attached to it. This script checks for spacebar input. If the spacebar has been pressed (or held down) and at least *fireRate* time has passed it will shoot one projectile. The shooting is done by using the Unity *instantiate* function that generates a prefab at a given location. This is used to create a net-ball prefab directly in front of the sub. The prefabs velocity is immediately set to *speed* in the local forward direction (i.e. the direction the sub is facing). The shoot sound attached to the prefab is set to *PlayOnAwake* which makes the sound effect immediately upon instantiation.

The net-ball prefab contains a *catchFish* script. This script detects any collisions using Unity's *OnCollisionEnter* function. If a collision occurs the script checks if the collision was with a fish. This is done by checking the *tag,* a built in functionality in Unity that allows you to give game objects labels such as "Player", "Enemy" "Waypoint" etc. If the tag of the object the ball collided with is "Fish" then the script proceeds to instantiate the capture particle effect at the position of the collision. There is also a sound source attached to the particle effect that plays the capture sound. It also calls the *Destroy* function on both the fish and the ball.

The capture particle system and the net-ball prefab both have a script that destroys them after *time* seconds. This is done to avoid cluttering the scene with net-balls and particle effects.

Major Feature #2: New Submarine

The new player Submarine includes a new model, animated propellers, bubble particles, a flashing red light, engine sounds and better collision detection. The new model was downloaded from the internet.

The model was one whole piece that did not have separate propellers. I imported the model into Blender (3D modelling software) and separated the propellers from the main body. I imported the new models into Unity and applied the textures to them.

With the propellers separated I was able to animate them. I created an animation that rotated a propeller 360 degrees to be the idle animation. I also created a second animation that did the same thing in half the time to be used when the submarine was accelerating. If there is no input to move forward (i.e. up arrow) the current animation is set to the slow rotating. When the up arrow is pressed the slow animation is crossfaded into the fast animation (the crossfade functionality is built into Unity). The fast animation remains active until the up arrow is released at which point it crossfades back to the slow animation.

The bubble particle systems are created as 4 instances of a prefab. The system uses Unity's Shuriken particle system. The system lets you tweak various parameters such as particle velocity, amount, gravity, size, emission shape, emission rate etc. I played with these settings to get a presentable particle system. I then applied a bubble texture to the particles. By default the emission rate is set to a low value. Whenever the up arrow is pressed the emission rate is increased to match the faster spinning propellers. It is brought back down to the lower rate once the up arrow key is released.

By default the engine sound effect is set to have a low pitch. Just as the propeller animation and bubble particles change when the submarine accelerates the pitch of the engine effect is changed. For every frame that the up arrow is pressed the pitch is increased by a small amount (up until a maximum). For every frame that the up arrow is not pressed the pitch is decreased by the same amount (down to a minimum).

These three effects work together to create a more realistic and game-like player model. Accelerating smoothly speeds up the propellers, emits more bubble particles and raises the pitch of the engine sound.

The red light on the submarine is create with Unity's built in point light game object. The lights colour was changed to red and it was positioned on the back of the submarine model. The C# *pingpong* math function was used to modify the lights intensity with respect to a flash *speed*. The alternating increase and decrease in the lights intensity makes it seem like it's flashing.