Swimming Blobby Man in HTML5/Javascript

Ramya Babu

December 15, 2013

1 Introduction

This project expands upon previous years' work of the blobby man renditions. This project is different due to the language used to program and the action that the blobby man takes. Previous versions include running, working out, and biking. This version will depict a swimmer. The purpose of this project was to illustrate the dynamics of swimming for different styles. This project will be helpful to the public because it can help athletes avoid injuries and improve upon their current lap times.

This paper is a description of details of dynamics of the swimming strokes that were modeled. It also goes into the necessities for coding with the three.js library and the WebGL renderer in HTML5. The first section will provide a brief introduction to swimming and its various styles. The last section will cover the technical aspects with using the three.js library.

2 Swimming

There are various different styles or forms of swimming. Each style is formally known as a stroke. There are four traditional strokes: freestyle, backstroke, breaststroke, and butterfly. In addition to these, other non-traditional strokes have been used such as the corkscrew and side-stroke. This project only focused on the traditional strokes.

2.1 Freestyle

Freestyle is the most popular of the four strokes and it is the easiest stroke for beginner swimmers to learn. It is a simple flutter kick and a windmill arm movement, with the individual facing down (on the belly). The difficult part is coordinating breathing, as the face is under water most of the time.

The leg kick is a flutter kick where the legs are kicked alternatively (one after the other). The knees are slightly bent and the feet and ankles are relaxed. The emphasis of the kick is on the down stroke as it provides the propulsion forward.

The arms are moved in a windmill motion alternatively. Each arm is pulled through the water with equal strength and arm reach to ensure straight motion. The hands should be cupped but the wrist and hand should be kept relaxed during recovery.

One hand is raised to begin the stroke. While doing so, the shoulder rises and the head is turned enough to leave the water to breath. Take as many breaths as needed. Next, the head is returned to the water while air is exhaled through the nose. The procedure is repeated, with the head turned the other side in coordination with the beginning of the opposite arm stroke.

2.2 Backstroke

This stroke is similar to freestyle as it also uses an alternative flutter kick with windmill arm motions. The big difference, as its name suggests, is that it is done on the back. Unlike freestyle, the emphasis for the kick is on the up stroke to provide the propulsion needed to move. As with freestyle, the arms move in a windmill motion. However, the thumb exits the water first and enters the water last. The breathing with backstroke is not difficult as the head is always kept above the water.

2.3 Butterfly

Butterfly is a difficult stroke that is not recommended for beginners as it needs perfect timing and a great deal of strength. For this stroke, the legs move in unison in a dolphin kick (like a mermaid). The arms move together to push the water downward and back, while the torso undulates like an earthworm as the body moves forward through the water. The breathing is done at the end of the arm stroke at which point the head is out of the water.

For the leg kick, the legs are bent slightly at the knee and kept together. A downward thrust is made by whipping the feet downward and straightening out the knees. Ideally, there should be two kicks for every one stroke.

The arms move together and are pulled through the water with the hands cupped. The stroke begins with the hands out in front with the palms facing outward away from each other. The hands are then pressed down and outward. The arms are swung forward in a sweeping motion to complete the stroke.

2.4 Breaststroke

Breaststroke is the most difficult stroke and requires exquisite timing. In competitions, swimmers are disqualified if any stroke is missed. This stroke involves a form where the body bobs up and down as it glides forward through the water. The breathing is done in time with the arm stroke.

For the leg kick, the knees are first brought toward the chest. The legs are then thrust outward and apart and straightened. Next they are snapped together to push the water and provide the propulsion for the forward movement. The kick is also known as the frog kick.

The arms start out overhead. The water is then pulled on, bringing the arms down toward the chest. The hands are kept cupped throughout and the arms are returned to starting position.

3 Three.js

The Three.js library is a useful for creating 3D objects in HTML within the browser. With this library, one can create cameras, objects, lights, materials, and more.

3.1 The Basics

Before creating objects with the three.js library, there are some features that must be present to visualize and see the objects. They are a scene, camera, and renderer. A scene is the space to which objects can be added. A camera is needed to see the objects in the scene, and a renderer is needed to add the objects to the scene.

The scene is set and created using the following line of code inside the script tags.

```
var scene = new THREE.Scene();
```

The Three.js library offers a few different camera options. They are the basic camera, the perspective camera, and the orthographic camera. This project used a perspective camera. The general construct for such a camera is as follows:

PerspectiveCamera(fov, aspect, near, far)
where
fov — Camera frustum vertical field of view.
aspect — Camera frustum aspect ratio.
near — Camera frustum near plane.
far — Camera frustum far plane.

This project initialized the camera using the following lines of code. The first creates the camera, which the second positions it.

```
var camera = new THREE.PerspectiveCamera(45, window.innerWidth/
window.innerHeight, 1, 5000);
camera.position.z = 1000;
```

Last but not least, a renderer is needed. With the Three.js library, HTML's canvas renderer or the WebGL renderer can be used. This project used the WebGL renderer. The renderer is initialized and created using the following:

```
var renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);
```

Objects are added to scene using the renderer. For example, the camera is added to the scene by writing:

```
renderer.render( scene, camera );
```

3.2 3D Object

Now that we have the essentials out of the way, 3D objects can now be made. The Three.js library provides classes to create many different 3D. Each 3D object from the javascript library requires a geometry shape and a mesh material. The main geometrical shapes were used in this project are the sphere geometry, the cylinder geometry, and the cube geometry. The commonly used meshes that Three.js provides are the basic mesh, lambert mesh, and the phong mesh. This project used only the basic mesh.

The code is reproduced below for adding a sample sphere:

```
var sphereg = new THREE.SphereGeometry(20,100,100);
var spherem = new THREE.MeshBasicMaterial({color:0xFFFF30});
var Sphere = new THREE.Mesh(sphereg, spherem);
```

The sphere geometry requires three input parameters. First is the radius of the sphere. Second is the number of horizontal segments. The third is the number of vertical segments. The only parameter needed for the basic mesh material is a color which can be given in a variety of formats. Above, it is given with the hexadecimal number for the color. The sphere is then created by making a mesh with the geometry and the material as parameters.

3.3 Movement

This project created movement using rotations and translations. In addition to these, mouse controls were added using the TrackballControls class of Three.js.
Rotations were done around either the x, y, or z axes using of the following lines of code:

```
[Object].rotation.x = [Degrees of rotation in radians];
[Object].rotation.y = [Degrees of rotation in radians];
[Object].rotation.z = [Degrees of rotation in radians];
```

The translations were done using either translation matrices or abbreviated versions. Code reproductions are provided below, first with a translation matrix. The second uses the abbreviated version. However, the abbreviated versions can only translate the object in one direction.

```
[Object].applyMatrix(new THREE.Matrix4().makeTranslation(x,y,z));
[Object].translate[X/Y/Z]([distance]);
```

The TrackballControls were added with the following lines of code during initialization:

```
controls = new THREE.TrackballControls( camera );
controls.rotateSpeed = 1.Ø;
controls.zoomSpeed = 1.2;
controls.panSpeed = Ø.8;
controls.noZoom = false;
controls.noPan = false;
controls.staticMoving = true;
controls.dynamicDampingFactor = Ø.3;
controls.keys = [65, 83, 68 ];
controls.addEventListener( 'change', render );
```

And this line of code was added into the animate function to update the controls whenever the mouse was used.

```
controls.update();
```