**Physics Simulation JavaScript Code**

// Physics\_Simulation.js

// Joshua Simmons 2018

window.onload = function() {

// Creates arrays to be populated with bodies/meshes later in the code

let meshes = [];

let bodies = [];

// Set up the renderer, scene, and camera

let renderer, scene, camera

renderer = new THREE.WebGLRenderer();

renderer.setClearColor(0x000000);

renderer.setPixelRatio(window.devicePixelRatio);

renderer.setSize(window.innerWidth-100, window.innerHeight-150);

document.body.appendChild(renderer.domElement);

scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera(

35, (window.innerWidth-100)/(window.innerHeight-100), .1, 10000);

// Creates a light to illuminate the objects

let light = new THREE.PointLight(0xFFFFFF, .5);

scene.add(light);

// Determines shadows on the plane and objects

let shadow = new THREE.DirectionalLight(0x777777, 1.75);

var d = 50;

shadow.position.set(d, -d, d);

shadow.castShadow = true;

shadow.shadowMapWidth = 1024;

shadow.shadowMapHeight = 1024;

shadow.shadowCameraLeft = -d;

shadow.shadowCameraRight = d;

shadow.shadowCameraTop = d\*2;

shadow.shadowCameraBottom = -d;

shadow.shadowCameraFar = 3\*d;

shadow.shadowCameraNear = d;

shadow.shadowDarkness = 0.75;

scene.add(shadow);

//Camera positioning

camera.position.set(0, -80, 25);

camera.rotation.x = -5;

// Set up our physics world in Cannon.js

let world = new CANNON.World();

world.gravity.set(0, 0, -9.8); // m/s²

// Creates a ground object for the shapes to land on

let groundBox = new CANNON.Body({

mass: 0, // makes the shape immobile, but still a rigid body.

position: new CANNON.Vec3(0, 0, 1),

shape: new CANNON.Box(new CANNON.Vec3(30, 30, 1))

});

world.addBody(groundBox);

bodies.push(groundBox);

// Creates the mesh for the ground object

ground = new THREE.BoxGeometry(60, 60, 2);

material = new THREE.MeshPhongMaterial({color: 0x777777});

planeMesh = new THREE.Mesh(ground, material);

planeMesh.receiveShadow = true;

planeMesh.castShadow = true;

meshes.push(planeMesh);

scene.add(planeMesh);

// This function creates a mesh when passed geometry and a material,

// and adds it to the meshes array

const createMesh = function(geo, mat) {

let mesh = new THREE.Mesh(geo, mat);

mesh.receiveShadow = true;

mesh.castShadow = true;

scene.add(mesh);

meshes.push(mesh);

}

// Generates a random number to be used as a coordinate, from -5 to 5

const getRandom = function() {

return Math.random() \* (5 - -5) - 5;

}

// Generates a random number from -20 to 20, to be used as a force

const getRandomForce = function() {

return Math.random() \* (20 - -20) - 20;

}

// Generates a random color with RGB components

const randomColor = function() {

return new THREE.Color(Math.random(), Math.random(), Math.random());

}

// Creates a box at a random position, with a random color

const createBox = function() {

let x = getRandom();

let y = getRandom();

let box = new CANNON.Body({

mass: 10,

position: new CANNON.Vec3(x, y, 20),

shape: new CANNON.Box(new CANNON.Vec3(1,1,1)) });

world.addBody(box);

bodies.push(box);

let boxGeo = new THREE.BoxGeometry(2,2,2);

let material = new THREE.MeshPhongMaterial({color: randomColor()});

createMesh(boxGeo, material);

};

// Creates a sphere at a random position, with a random color

const createSphere = function() {

let x = getRandom();

let y = getRandom();

let sphere = new CANNON.Body({

mass: 10,

position: new CANNON.Vec3(x, y, 20),

shape: new CANNON.Sphere(1)

});

world.addBody(sphere);

bodies.push(sphere);

let sphereGeo = new THREE.SphereGeometry(1, 24, 24);

let material = new THREE.MeshPhongMaterial({color: randomColor()});

createMesh(sphereGeo, material);

};

// Creates a cylinder at a random position, with a random color

const createCylinder = function() {

let x = getRandom();

let y = getRandom();

let cylinder = new CANNON.Body({

mass: 10,

position: new CANNON.Vec3(x, y, 20)});

let cylinderShape = new CANNON.Cylinder(1, 1, 2, 20);

// This code reorients the body, so that the Three.js rendered mesh

// matches the orientation of the Cannon.js body.

let quat = new CANNON.Quaternion();

quat.setFromAxisAngle(new CANNON.Vec3(1,0,0), -Math.PI/2);

let translation = new CANNON.Vec3(0,0,0);

cylinderShape.transformAllPoints(translation, quat);

cylinder.addShape(cylinderShape);

world.addBody(cylinder);

bodies.push(cylinder);

let cylGeo = new THREE.CylinderGeometry(1, 1, 2, 20);

let material = new THREE.MeshPhongMaterial({color: randomColor()});

createMesh(cylGeo, material);

};

var fixedTimeStep = 1.0 / 60; // seconds

var maxSubSteps = 3;

let lastTime;

// Values needed for the shadow calculations

renderer.gammaInput = true;

renderer.gammaOutput = true;

renderer.shadowMapEnabled = true;

renderer.render(scene, camera);

// This function is the physics simulation loop

(function simloop(time){

requestAnimationFrame(simloop);

if(lastTime !== undefined){

var dt = (time - lastTime) / 1000;

world.step(fixedTimeStep, dt, maxSubSteps);

}

// This loop cycles through the arrays of meshes, and updates

// their position and rotation to match their counterparts in the

// physics simulation

for (let i = 0; i < meshes.length; i++) {

meshes[i].position.copy(bodies[i].position);

meshes[i].quaternion.copy(bodies[i].quaternion);

// If a body is below a certain point, it will stop calculating

// physics, and stop rendering in the Three.js scene

if (bodies[i].position.z < -200) {

bodies[i].sleep

meshes[i].visible = false;

}

}

// When a key is pressed, if it is a certain key, perform a certain action.

window.onkeyup = function(event) {

if (event.keyCode === 66) {

createBox();

} else if (event.keyCode === 83) {

createSphere();

} else if (event.keyCode === 67) {

createCylinder();

} else if (event.keyCode === 74) {

// This event applies a force of 50 Newtons in the positive z direction,

// as well as a random force between 0 and 20 Newtons, in either the

// positive or negative x and y directions

for (let i = 0; i < bodies.length; i++) {

bodies[i].applyImpulse(

new CANNON.Vec3(getRandomForce(), getRandomForce(), 50),

new CANNON.Vec3(bodies[i].position.x, bodies[i].position.y,0));

}

}

};

// Updates the scene with new positions/rotations

renderer.render(scene, camera);

lastTime = time;

})();

};

**Physics Simulation HTML Code**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="style.css" />

<script src="three.js"></script>

<script src="cannon.min.js"></script>

<script src = "Physics\_Simulation.js"></script>

<title>Physics!</title>

<body>

<h4>Press 'B' to add a box, 'C' to add a cylinder, and 'S' to add a sphere!</h4>

<h4>Press 'J' to make the objects jump!</h4>

</body>

</head>

</html>