Algorithmic Art

Three.js Overview and Worked Example

Carl Bateman
WebGL Workshop

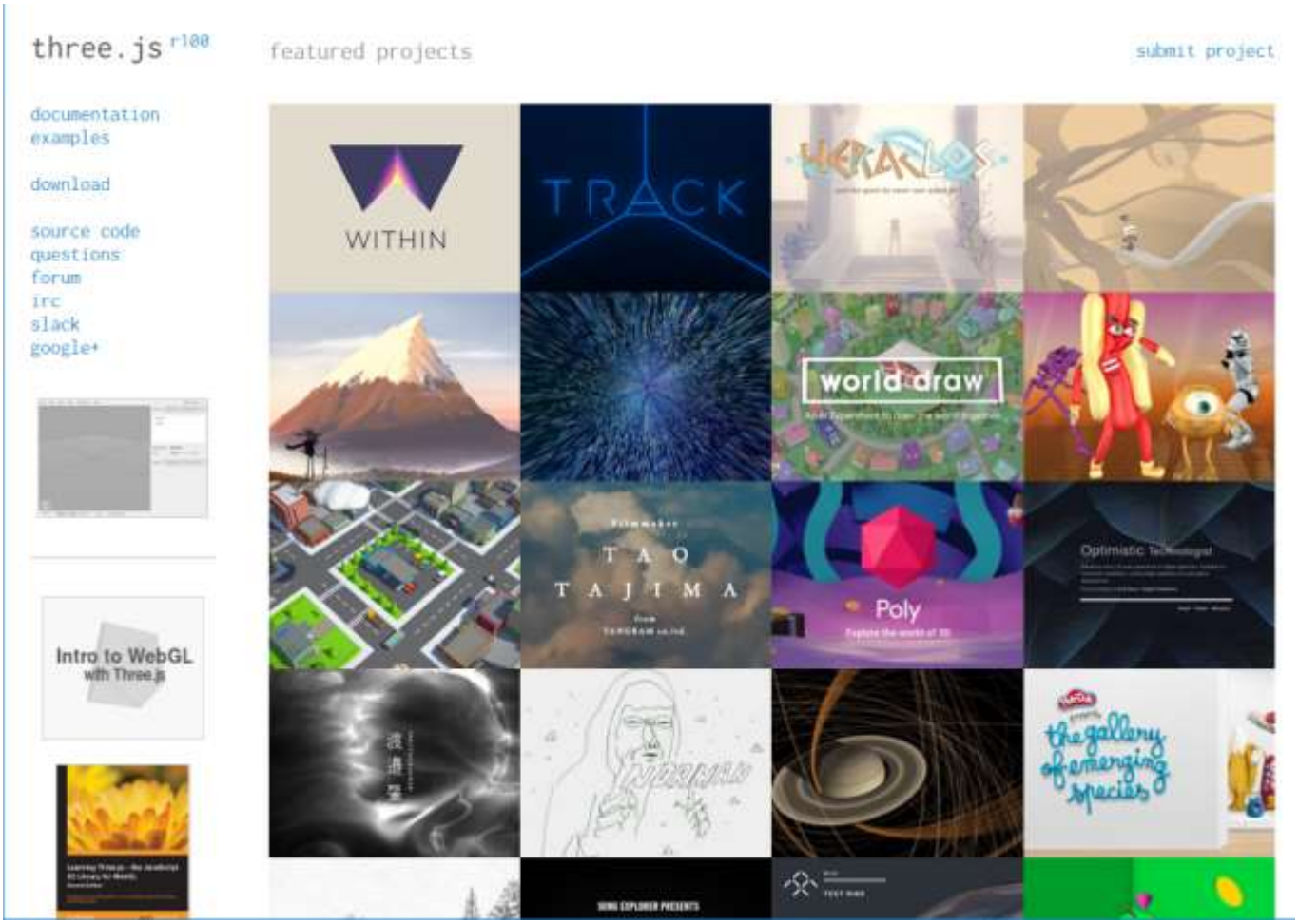
Table of Contents

Part 1. Agenda

Well, take a look at the table of contents.

- Intro to three.js:
  - look at the three.js website
  - look at some of resources available
- Review the basic sample app from the three.js web site
- Using Glitch.com for editing
- Make a few minor changes:
  - material/colour
  - control
- Properties to change:
  - position
  - rotation
  - scale
  - colour
- Factors/functions to derive values to use to control the above
  - step
  - cos, sin
  - sqrt

Part 2. threejs.org



Some of my favourites

Cube Slam

https://www.cubeslam.com

Moments of Happiness

https://moments.epic.net/

http://hands.wtf/

Part 3. Preparation

https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene

The three.js site has the following sample HTML/JavaScript to create a simple scene, a rotating green cube.

```
<html>
  <head>
    <title>My first three.js app</title>
    <style>
      body { margin: 0; }
      canvas { width: 100%; height: 100% }
    </style>
  </head>
  <body>
    <script src="js/three.js"></script>
    <script>
      var scene = new THREE.Scene();
      var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.innerHeight, 0.1, 1000 );

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

      var geometry = new THREE.BoxGeometry( 1, 1, 1 );

      var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );

      var cube = new THREE.Mesh( geometry, material );
      scene.add( cube );

      camera.position.z = 5;

      var animate = function () {
        requestAnimationFrame( animate );

        cube.rotation.x += 0.01;
        cube.rotation.y += 0.01;

        renderer.render( scene, camera );
      };

      animate();
    </script>
  </body>
</html>
```

Glitch.com

We'll be using Glitch.com as it provides an online editor and lets us share our work. You don't need to create an account, but it would probably help.

There's a basic project based on the code above set up at **https://glitch.com/~algoart-feb19-a**

The code is split into three parts (index.html, style.css and script.js), because this the way glitch.com organises things. A couple of additional changes have been made.

index.html

The script source tag in index.html has been changed to work.

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <title>My first three.js app</title>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- import the webpage's stylesheet -->
    <link rel="stylesheet" href="/style.css">

    <!-- import three.js -->
    <script src="https://threejs.org/build/three.js"></script>

    <!-- import the webpage's javascript file -->
    <script src="/script.js" defer></script>
  </head>
  <body>

  </body>
</html>
```

style.css

style.css has been changed so things look better.

```
body { margin: 0; overflow: hidden; }
canvas { width: 100%; height: 100% }
```

script.js

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.innerHeight, 0.1, 1000 );

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

var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );

camera.position.z = 5;

var animate = function () {
  requestAnimationFrame( animate );

  cube.rotation.x += 0.01;
  cube.rotation.y += 0.01;

  renderer.render( scene, camera );
};

animate();
```

Part 4. It's Editing time!

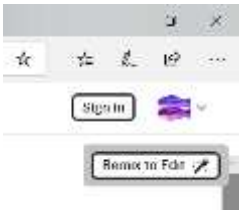
Moar Preparation (Sorry)

We still need to make a few more changes before we're ready to get stuck in. You can skip to the next section if this looks like too much like busy work.

Remix!

Reminder: The basic project is at **https://glitch.com/~algoart-feb19-a**

Click on the Remix to Edit button in the top right.



This will create your own version of the project to work on.

Better Colour

Change

```
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
```

to

```
var material = new THREE.MeshNormalMaterial();
```

The cube should actually look like a cube.

Take (Interactive) Control!

To the **HTML** add

```
<script src="https://threejs.org/examples/js/controls/OrbitControls.js"></script>
```

To the **JavaScript** add

```
var controls = new THREE.OrbitControls( camera );
```

Delete or comment out

```
//cube.rotation.x += 0.01;
//cube.rotation.y += 0.01;
```

Add

```
control.update();
```

Now we can control the position and orientation of the box with the mouse.

Add a Pile of Boxes

We need something to work with. Let's add a "pile" of boxes!

Replace

```
var material = new THREE. MeshNormalMaterial(  );
```

```
var cube = new THREE.Mesh( geometry, material );
```

with

```
var boxes = [];
var numBoxes = 9;

for (let x = -numBoxes; x <= numBoxes; x++) {
  for (let y = -numBoxes; y <= numBoxes; y++) {
    var material = new THREE.MeshNormalMaterial( );
    var cube = new THREE.Mesh( geometry, material );
    cube.position.x = x * 1.1;
    cube.position.y = y * 1.1;
    boxes.push(cube);
    scene.add(cube);
  }
}
```

Add a function to update the boxes, with a step parameter to control the speed of any changes we make. We need a way to make changes over time. Three.js provides a clock function, but a simple counter/step will suffice. This is just a skeleton function and doesn't anything just yet.

```
let step = 0;
function updateBoxes(step) {
  for (let i = 0; i < boxes.length; i++) {

  }
}

var animate = function () {
  requestAnimationFrame( animate );

  controls.update();

  step += 0.05;
  updateBoxes (step);

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

To produce interesting effects, maths functions are very useful, particularly: cos and sin. Since we'll be using them a lot let's alias the Math functions, we'll be using.

```
var cos = Math.cos;
var sin = Math.sin;
var sqrt = Math.sqrt;
var abs = Math.abs;
var PI = Math.PI;
```

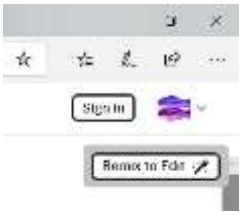
Part 5. Get Those Boxes Moving

OK! We're finally ready to get something interesting going.

Remix Reminder!

There's a starter project setup at **https://glitch.com/~algoart-feb19-d**

Click on the Remix to Edit button in the top right.



Fun Stuff

Animation

Messing about with values.

To get something interesting to happen on screen there are a number of properties we can influence:

Position, Scale, Rotation, Color*

* Please forgive the Americanism, I find it avoids confusion later on.

Position

```
function setBoxPosZ(box, step) {
  let x = box.position.x;
  let y = box.position.y;
  let md = abs(x) + abs(y);
  let d = sqrt(x*x + y*y);

  box.position.z = x;
  box.position.z = cos(x/2) + cos(y/2);
  box.position.z = cos(d);
  box.position.z = cos(d + step);
  box.position.z = d;
}
```

Scale

```
function setBoxScaleZ(box, step) {
  let x = box.position.x;
  let y = box.position.y;
  let md = abs(x) + abs(y);
  let d = sqrt(x*x + y*y);

  box.scale.z = (d);
  //position.z = d;//cos(abs(x) - abs(y) + step);
}
```

Rotation

```
function setBoxRotZ(box, step) {
  let x = box.position.x;
  let y = box.position.y;
```

```
  let md = abs(x) + abs(y);
  let d = sqrt(x*x + y*y);

  box.rotation.z = ((step*5+d)*Math.PI/45);
  //position.z = d;//cos(abs(x) - abs(y) + step);
}
```

Color

Normal material may look nice, but it doesn't support colour, but rather than use the basic material we'll use standard material which also supports/requires lighting, so we still get a feeling of perspective for the cubes.

Note: This a bit involved so you can just skip to **https://glitch.com/~algoart-feb19-e** to save time.

Change the material to StandardMaterial (everything's now black).

```
var material = new THREE.MeshStandardMaterial( );
```

Add a directional light

```
var directionalLight = new THREE.DirectionalLight( 0xffffff, 1 );
directionalLight.position.x = 1;
```

The shading is a bit dark, and an ambient light to fill them in.

```
scene.add( directionalLight );
```

Play time

https://threejsplaygnd.brangerbriz.com/gui/ is a great interactive resource for experimenting. Choose something from the drop downs and the relevant code is displayed, copy 'n' paste, simples!

Change shapes (sphere, torus, cylinder, cone)

We now have control of an array of attributes to create interesting effect.

We can now combine the different property changes, and tweak the functions for different effects.

Try changing to a positional spotlight, control its position.

Standard material has some nice additional attributes, such as shininess. Try changing them.

Part 6. More advanced stuff

More complex movement/orbits can be achieved by grouping/nesting objects. Objects can be grouped or nested together with each other by adding one to another (as well as to the scene) or by using a Group object or an Object3D.

This can make orbital motion a lot easier, doing away with having to keep track of sines and cosines.

https://glitch.com/~algoart-feb19-g

```
for (let x = -numBoxes; x <= numBoxes; x++) {
  var anchor = new THREE.Group();
  var material = new THREE.MeshStandardMaterial( );
  var box = new THREE.Mesh( geometry, material );
  box.position.x = 5;
  boxes.push(anchor);
  anchor.add(box);
  scene.add(anchor);
  anchor.rotation.z = x * PI / (numBoxes);
}


function setOrbitZ(box, step) {
  let x = box.position.x;
  let y = box.position.y;
  let md = abs(x) + abs(y);
  let d = sqrt(x*x + y*y);

  box.rotation.z += 0.01;

  box.children[0].position.x = 10*cos(d + step/10);
  box.children[0].rotation.x = 10*cos(d + step/10);
  box.children[0].material.color.r = abs(box.children[0].getWorldPosition().x)/2;
  box.children[0].material.color.g = abs(box.children[0].getWorldPosition().y)/2;
  box.children[0].material.color.b = abs(box.children[0].getWorldPosition().z)/2;
}
```

Part 7. Something A Bit Different, SDF

Rather than playing around with whole objects, we can just as easily manipulate the individual vertices that make up an object.

https://glitch.com/~algoart-feb19-k

```
function wave(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = geometry.vertices[i].x;
    let y = geometry.vertices[i].y;
    let r1 = Math.sqrt(x*x+y*y)*3;
    let r2 = (Math.abs(x)+Math.abs(y))*2;
    let r3 = Math.abs(Math.abs(x)-Math.abs(y));
    let rad = Math.atan2(y, x)
    //geometry.vertices[i].z = (Math.cos(r1*r2+off)/30);
    geometry.vertices[i].z = (Math.cos(r3));
  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

function sqwave(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = geometry.vertices[i].x;
    let y = geometry.vertices[i].y;
    let r1 = Math.sqrt(x*x+y*y)*3;
    let r2 = (Math.abs(x)+Math.abs(y));
    let r3 = Math.abs(Math.abs(x)-Math.abs(y));
    let rad = Math.atan2(y, x)
    let b = Math.abs(x)<Math.abs(y)?Math.abs(x):Math.abs(y);
    //geometry.vertices[i].z = (Math.cos(r1*r2+off)/30);
    //geometry.vertices[i].z = Math.ceil(Math.cos(r1+off/20));
    geometry.vertices[i].z = Math.ceil(Math.cos(Math.ceil(r2)+off/2));
    //geometry.vertices[i].z = Math.cos(b*1+off);
    //geometry.vertices[i].z = Math.cos(r2+b+off)/2;

  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

function hemi(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = geometry.vertices[i].x;
    let y = geometry.vertices[i].y;
    let r = Math.sqrt(x*x+y*y);//Math.abs(x) + Math.abs(y);//
    // r += Math.sqrt(x*x+y*y);
    x=1-(r+1)%2;
     r = Math.sqrt(1-x*x);
    geometry.vertices[i].z = r;//Math.cos(r*2)/5;
    //geometry.vertices[i].z += Math.sin(r*5)/5;
  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

function manhat(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = geometry.vertices[i].x;
    let y = geometry.vertices[i].y;
    let r1 = (Math.abs(x)+Math.abs(y));
    let r2 = ( Math.abs(x) + Math.abs(y) );
    let r3 = Math.abs(x)>Math.abs(y)?Math.abs(x):Math.abs(y);
```

```
    let r = Math.sqrt(x*x+y*y);
    if(x<0 && y<0)
    geometry.vertices[i].z = Math.cos(r+off);
    else
    geometry.vertices[i].z = Math.cos(r3+off);
    //geometry.vertices[i].z = Math.cos(r3+off);
  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

function mix(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = geometry.vertices[i].x;
    let y = geometry.vertices[i].y;
    let r1 = (Math.abs(x)+Math.abs(y));
    let r2 = x+y;
    let r3 = Math.max(Math.abs(x),Math.abs(y));
    let r = Math.sqrt(x*x+y*y);

    geometry.vertices[i].z = Math.cos(-r1+r3-r+off);
  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

function heart(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = Math.abs(geometry.vertices[i].x);
    let y = Math.abs(geometry.vertices[i].y);
    let r1 = Math.pow((x*x + y*y - 1), 3) - x*x*x*y*y*y;
    let r2 = (x*x*x*x+y*y*y*y) - 2*x*x*y;
    let r3 = Math.max(Math.abs(x),Math.abs(y));
    let r = Math.sqrt(Math.abs(r2));

    let d = Math.pow((x*x+ y*y), 2) + 4*Math.abs(x)*(x*x+y*y)-4*y*y;


    geometry.vertices[i].z = Math.cos(r)/2;
  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

function flo(geometry, off){
  for (var i = 0; i < geometry.vertices.length; i++) {
    let x = geometry.vertices[i].x;
    let y = geometry.vertices[i].y;
    let r = Math.sqrt(x*x+y*y);
    let m = (Math.abs(x)+Math.abs(y));
    let b = Math.max(Math.abs(x),Math.abs(y))*.707;


    geometry.vertices[i].z = Math.cos(2.5*(m+b-r+off/2))/2;
  }
  geometry.verticesNeedUpdate = true;
  geometry.computeVertexNormals();
  geometry.computeFaceNormals();
}

var geometry = new THREE.PlaneGeometry( 10, 10, 100, 100 );

var material = new THREE.MeshNormalMaterial({side:THREE.DoubleSide});

var cube = new THREE.Mesh( geometry, material );
```

```
cube.rotation.x = -1;
cube.rotation.z = 1;
scene.add( cube );

camera.position.z = 10;

let off = 0;

var animate = function () {
  off-= 0.05;
  flo(cube.geometry, off);

  requestAnimationFrame( animate );

  controls.update();

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