Advanced Creative Coding with Three.js

Dr. Stavros Didakis (Session 1)

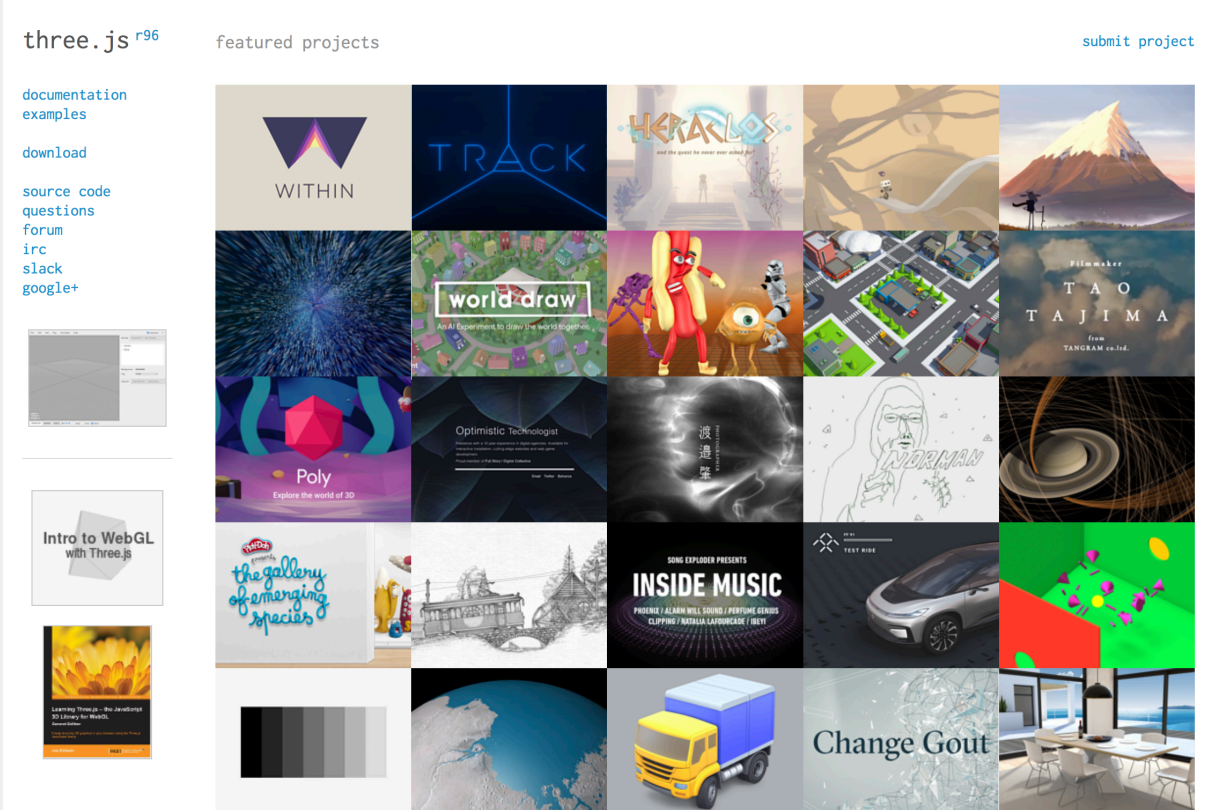

THREEJS
INTRO

Three.js (threejs.org)

Three.js is a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser.

Three.js allows the creation of Graphical Processing Unit (GPU)-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins.

Three.js utilizes the power of WebGL (and related libraries, such as WebAudio, WebVR, etc).



WebGL

WebGL (Web Graphics Library) is a JavaScript API for rendering interactive 2D and 3D graphics within any compatible web browser without the use of plug-ins. WebGL is integrated completely into all the web standards of the browser, allowing GPU-accelerated usage of physics and image processing and effects as part of the web page canvas.

WebGL elements can be mixed with other HTML elements and composited with other parts of the page or page background.[3] WebGL programs consist of control code written in JavaScript and shader code that is written in OpenGL ES Shading Language (GLSL ES), a language similar to C or C++, and is executed on a computer's graphics processing unit (GPU).

WebGL & Three.js

With WebGL, you can directly make use of the processing resources of your graphics card and create high-performance 2D and 3D computer graphics. Programming WebGL directly from JavaScript to create and animate 3D scenes is a very complex and error- prone process. Three.js is a library that makes this a lot easier.

Basically, Three.js runs on any of the modern browsers except older versions of IE.

Browser	Support
Mozilla Firefox	This browser has supported WebGL since version 4.0.
Google Chrome	This browser has supported WebGL since version 9.
Safari	Safari Version 5.1 and newer installed on Mac OS X Mountain Lion, Lion, or Snow Leopard supports WebGL. Make sure you enable WebGL in Safari. You can do this by going to **Preferences | Advanced** and checking **Show develop menu in menu bar**. After that, go to **Develop | Enable WebGL**.
Opera	This browser has supported WebGL since version 12.00. You still have to enable this by opening **opera:config** and setting the values of **WebGL** and **Enable Hardware Acceleration** to 1. After that, restart the browser.
Internet Explorer	Internet Explorer was for a long time the only major player that didn't support WebGL. Starting with IE11, Microsoft has added WebGL support.

WebGL & Three.js

It is also possible to run Three.js on mobile devices.

Device	Support
Android	The native browser for Android doesn't have WebGL support and is generally also lacking in support for modern HTML5 features. If you want to use WebGL on Android, you can use the latest Chrome, Firefox, or Opera mobile versions.
IOS	With IOS 8, there is also support for WebGL on IOS devices. IOS Safari version 8 has great WebGL support.
Windows mobile	Windows mobile supports WebGL since version 8.1.

Development Tools

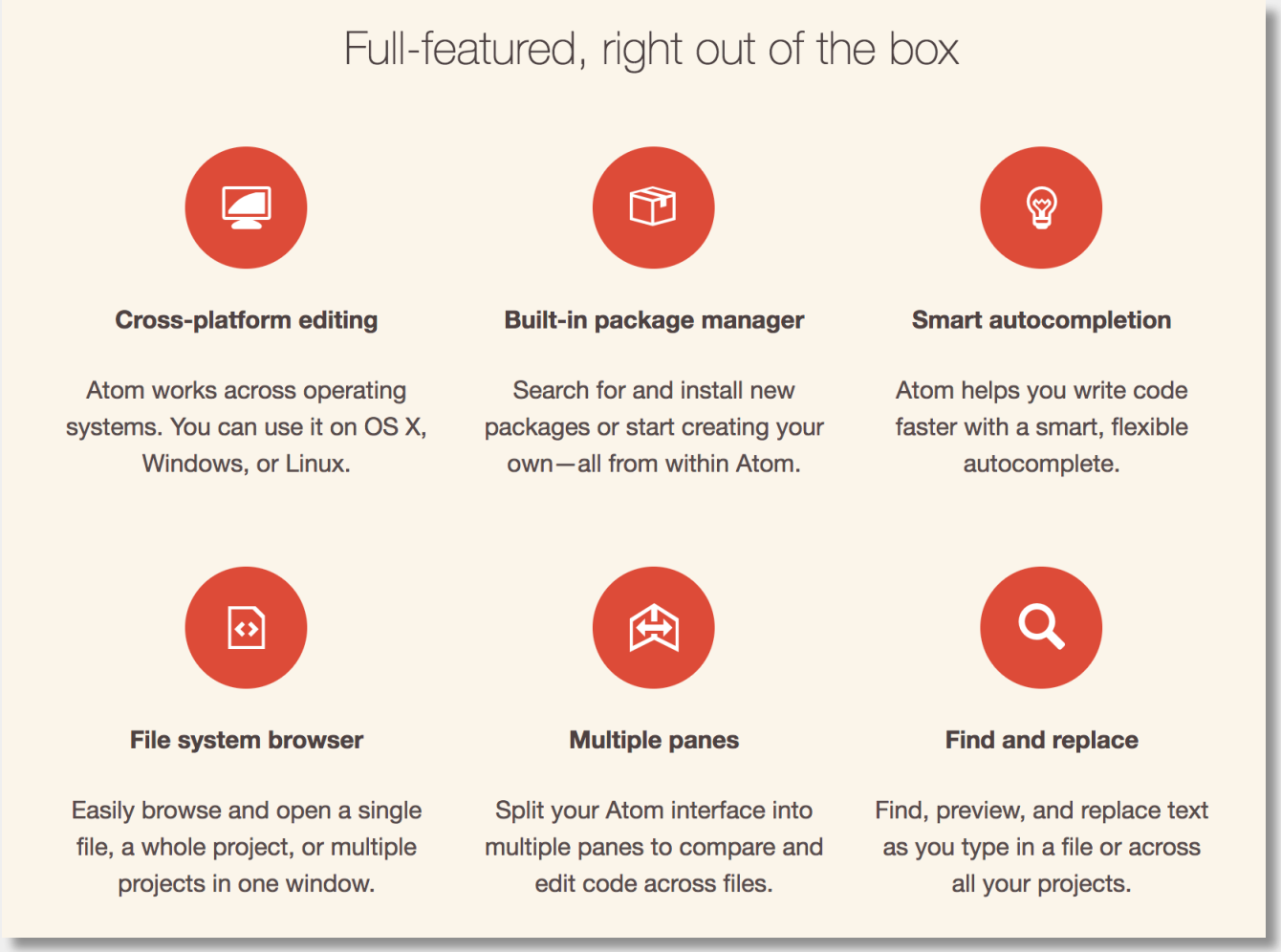
To start developing a project for **Three.js**, it is necessary to have a code editor for managing the project. There are mainly two archetypes: IDE (Integrated Development Environment), and lightweight editors such as:

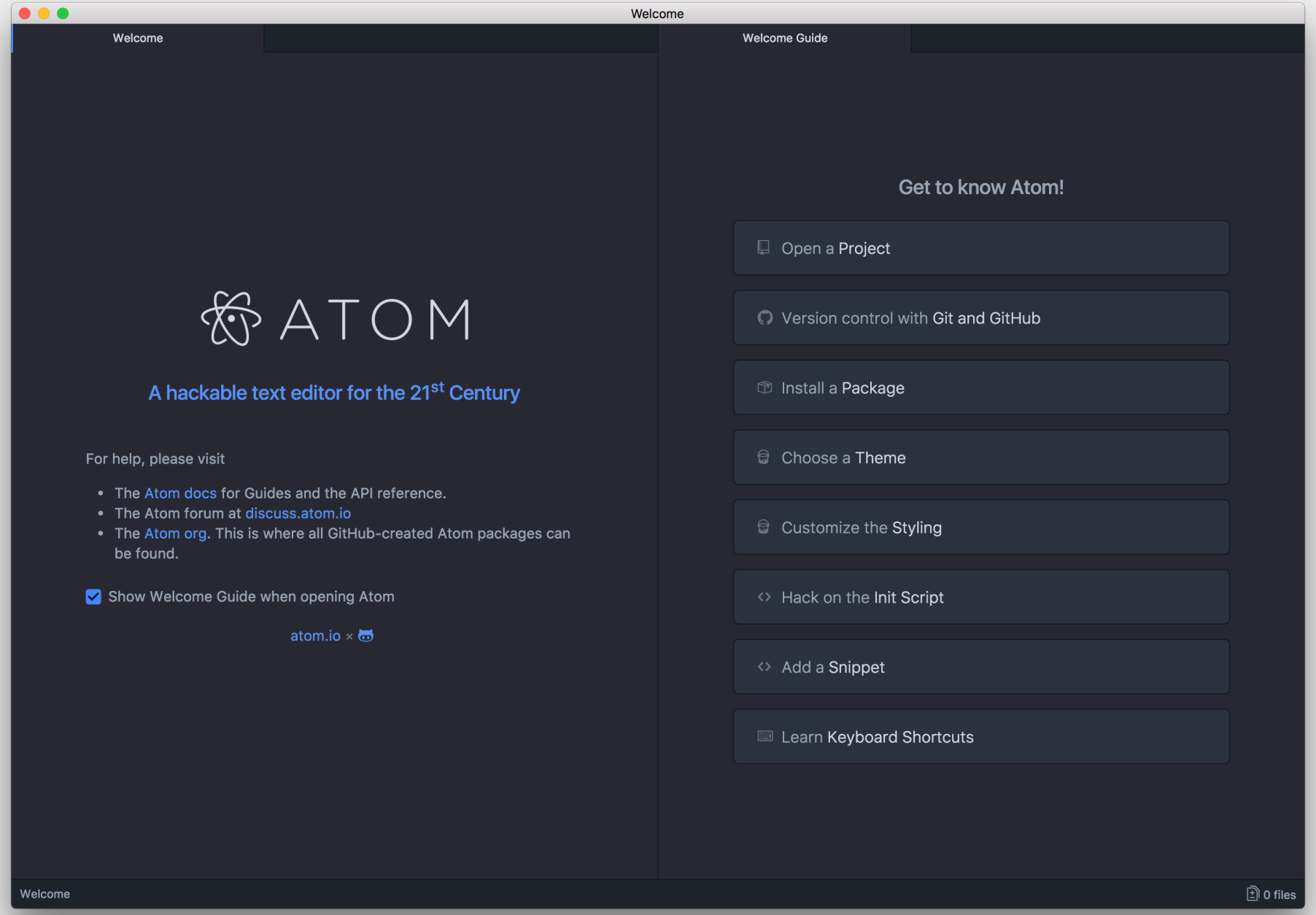
- Sublime
- Atom
- Brackets

In this workshop we will be using **Atom**, as it seems the most efficient editor with the best **Git** and **GitHub** integration, and **Chrome** as the preferable web browser.

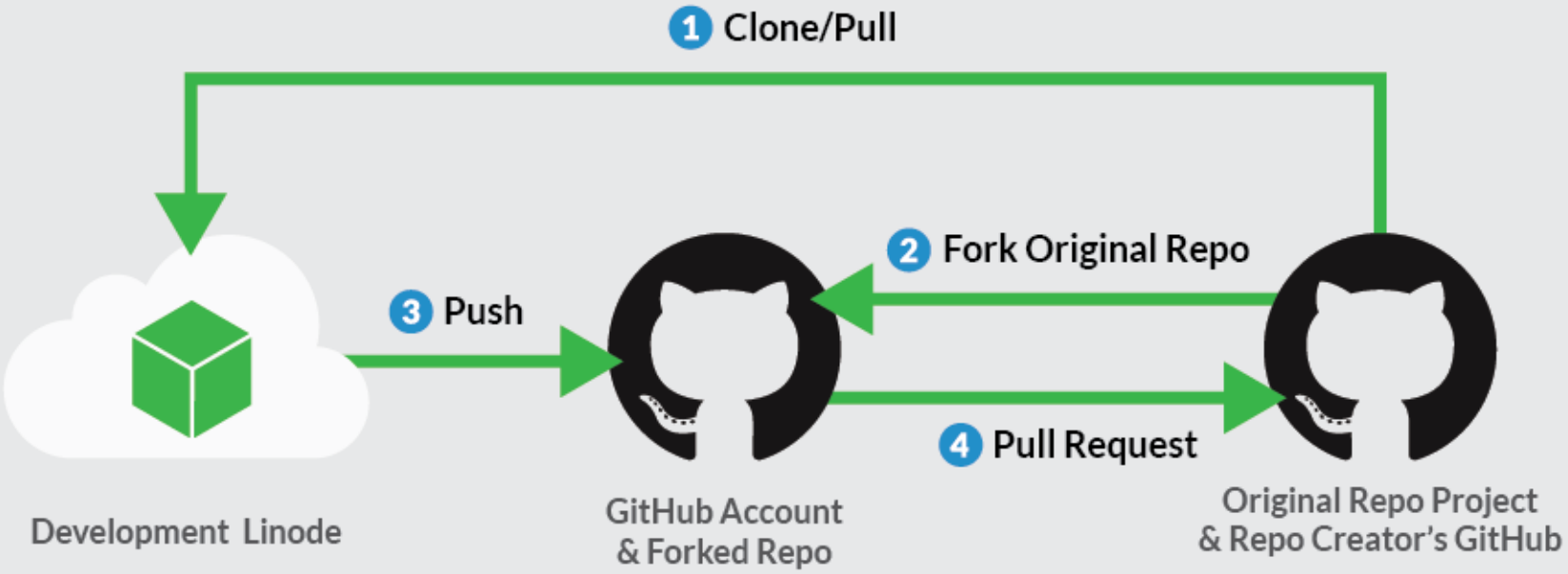
Atom (https://atom.io)

Atom is a cross-platform text editor with a large number of features such as a built-in package manager and highly effective customization.



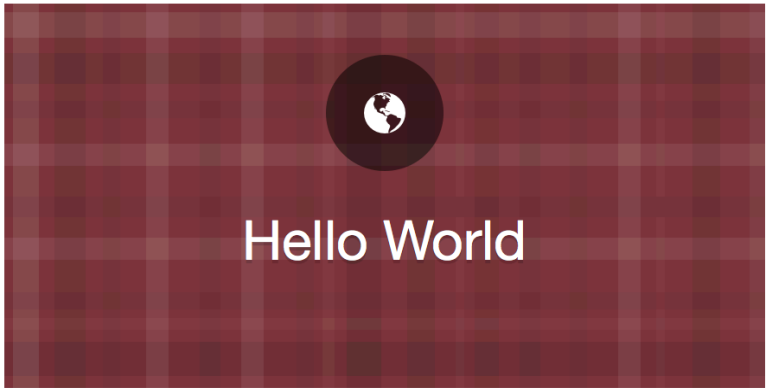

Welcome
Welcome
Welcome Guide
ATOM
A hackable text editor for the 21st Century
For help, please visit
The Atom docs for Guides and the API reference.
The Atom forum at discuss.atom.io
The Atom org. This is where all GitHub-created Atom packages can be found.
Show Welcome Guide when opening Atom
atom.io ×
Get to know Atom!
Open a Project
Version control with Git and GitHub
Install a Package
Choose a Theme
Customize the Styling
Hack on the Init Script
Add a Snippet
Learn Keyboard Shortcuts
Welcome
0 files

Git and GitHub



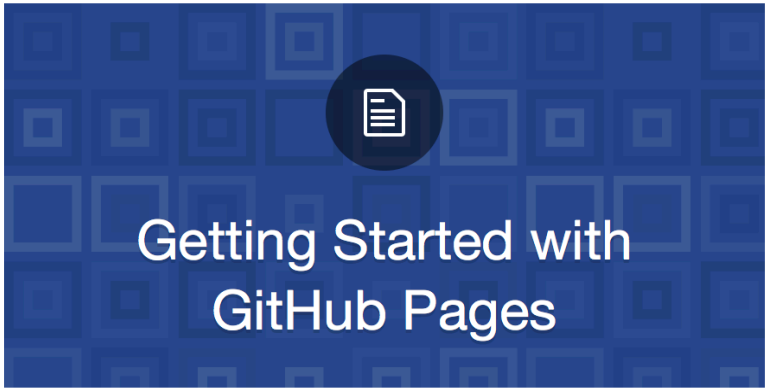
GitHub Flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly. This guide explains how and why GitHub Flow works.

5 minute read



The easiest way to get started with GitHub. In this guide you'll complete a time honored "Hello World" exercise, and learn GitHub essentials.

10 minute read



GitHub Pages are a great way to showcase some open source projects, host a blog, or even share your résumé. This guide will help get you started on creating your next website.

10 minute read

Learn about version control—in particular, Git, and how it works with GitHub.

10 minute read

Atom & GitHub

Here is a short video introduction of Atom:
https://www.youtube.com/watch?v=U5POoGSrtG

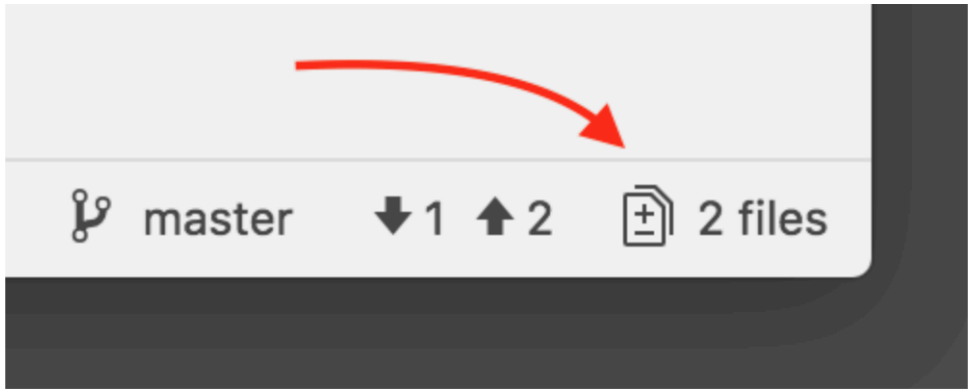
A link to the official documentation:
https://atom.io/docs

And more info on setting up Git and Github:
http://flight-manual.atom.io/using-atom/sections/github-package/

Most of the functionality lives in the Git and GitHub panel. There are different ways to access them, probably the most common way is through their keybindings:

- Open the **Git** panel: `Ctrl+9`
- Open the **GitHub** panel: `Ctrl+8`

Or you can also toggle the Git panel from the Status Bar by clicking on the changed files icon:



To access the GitHub for this particular class, please follow the link below:

https://github.com/stavrosdidakis/DAT505-Fall2018

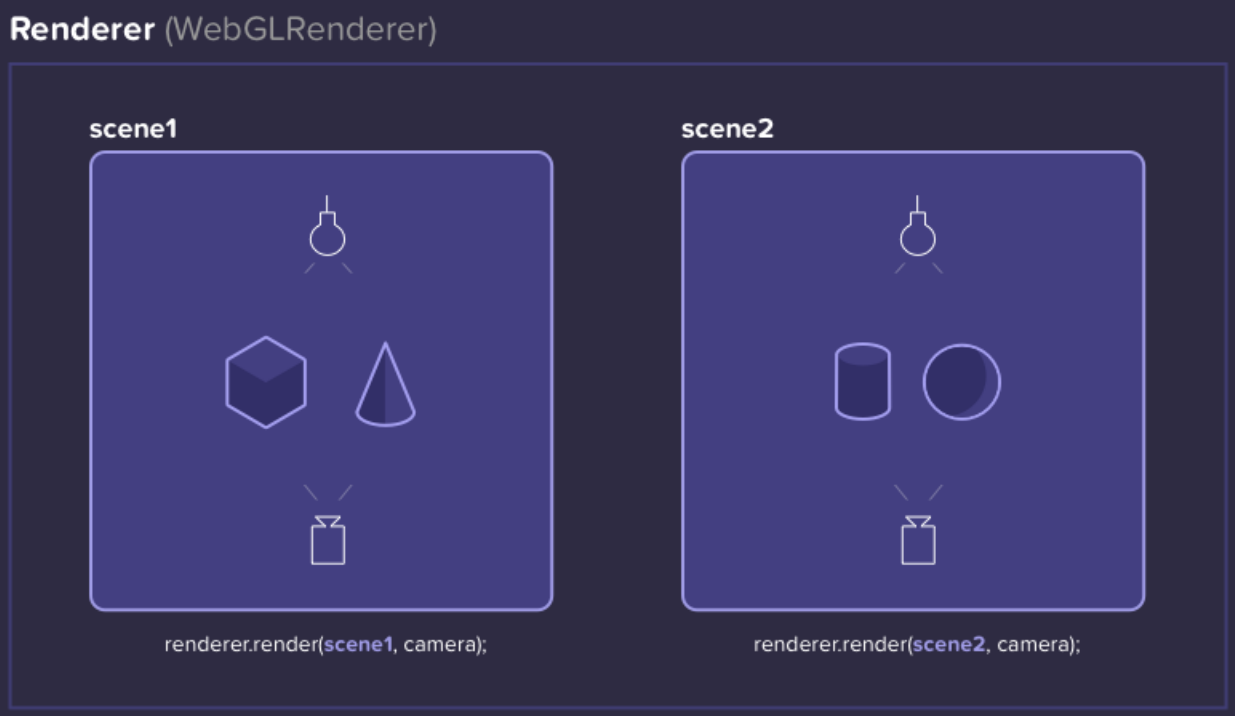

THREEJS
BASICS

Rendering a Basic Scene

This is the first common pattern we are going to see in every Three.js app:

- Create a Renderer
- Create a Scene
- Create a Camera

The renderer is the place where we are going to put the result of our scene.
In Three.js we can have multiple scenes and each one can have different objects.

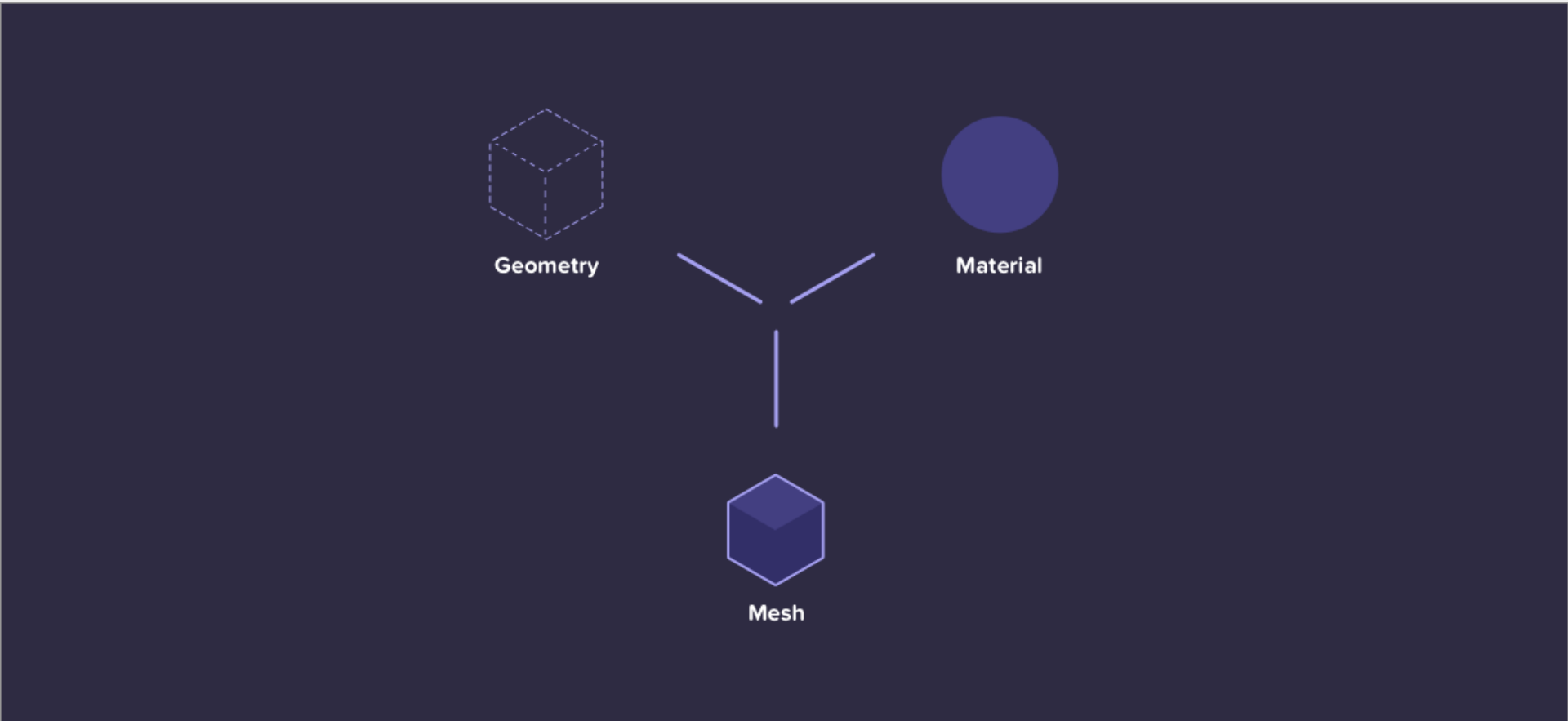


Rendering a Basic Scene

The second common pattern is adding objects to the scene:

- Create a Geometry
- Create a Material
- Create a Mesh.
- Add mesh to scene.

In Three.js a Mesh is the composition of a Geometry with a Material.



00_BasicStructure Example

Opening the example 00_BasicStructure, you will see the fundamental files needed for a project.

The files include the main **index.html** (show below), which has a standard HTML page formatting. To include the Three.js library, there are two ways: (1) either you load it locally – as in the example below, or (2) you use the CDN link (uncommented in this case).

In addition, the HTML code includes the **index.js** file, which is the file that will include our JavaScript code.

00_BasicStructure Example

Here we create an empty scene where we are going to add our objects (i.e. mesh), and also setup the camera, passing as parameters the FOV, the Aspect Ratio, near plane and far plane, and finally to add the WebGLRenderer, and pass to it the size of the window as a parameter and then append it to the DOM.

```
// Create an empty scene ----------------------------
var scene = new THREE.Scene();
// --------------------------------------------------

// Create a basic perspective camera ----------------
camera = new THREE.PerspectiveCamera(35, window.innerWidth /
  window.innerHeight, 300, 10000 );
// --------------------------------------------------

// Create a renderer with Antialiasing --------------
var renderer = new THREE.WebGLRenderer({antialias:true});

// Configure renderer clear color
renderer.setClearColor("#000000");

// Configure renderer size
renderer.setSize( window.innerWidth, window.innerHeight );

// Append Renderer to DOM
document.body.appendChild( renderer.domElement );
// --------------------------------------------------
```

00_BasicStructure Example

The next part demonstrates how we create an object (box in this case), create a material, and apply these two to the mesh object that is finally added to the scene. In addition, the function render takes care of the animation of the object, using the scene and camera settings.

```
// Create a Cube Mesh with basic material ----------
var geometry = new THREE.BoxGeometry(100, 100, 100);
var material = new THREE.MeshBasicMaterial( { color: "#433F81" } );
var mesh = new THREE.Mesh( geometry, material );
mesh.position.z = -1000;
// ------------------------------------------------

// Add mesh to scene
scene.add( mesh );

// Render Loop
var render = function () {
  requestAnimationFrame( render );

  mesh.rotation.x += 0.01; //Continuously rotate the mesh
  mesh.rotation.y += 0.01;

  // Render the scene
  renderer.render(scene, camera);
};

render(); //Run the function render
```

00_BasicStructure Example

To run this example, open the **index.html** in your Chrome browser.

Geometry, Material and Mesh

A Geometry is the mathematical formula of an object, providing for us the vertices of the object we want to add to the scene.

If you take time to look through the code for Three.js you'll see a lot of objects "inherit" from Object3D. This is a base object which contains some very useful properties, such as the **position**, **rotation** and **scale** information. In particular our Sphere is a Mesh which inherits from Object3D, to which it adds its own properties: **geometry** and **materials**.

SphereGeometry

BoxGeometry

ConeGeometry

CylinderGeometry

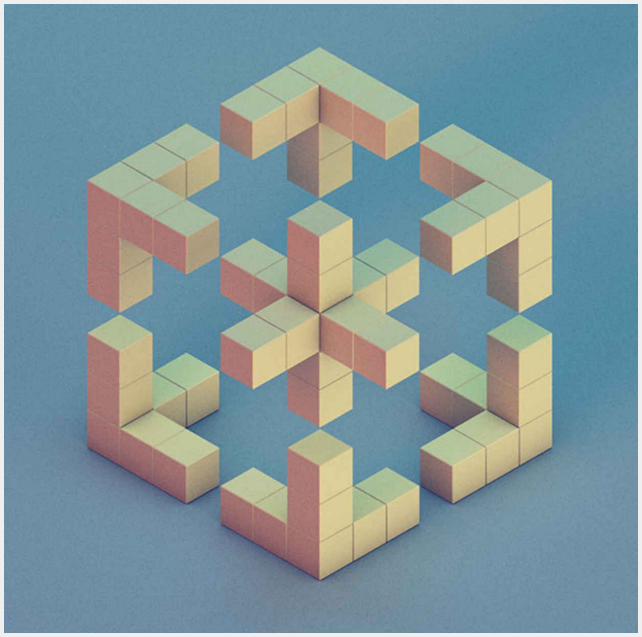
Exercise: 02_Exercise-GeometriesAndMaterials

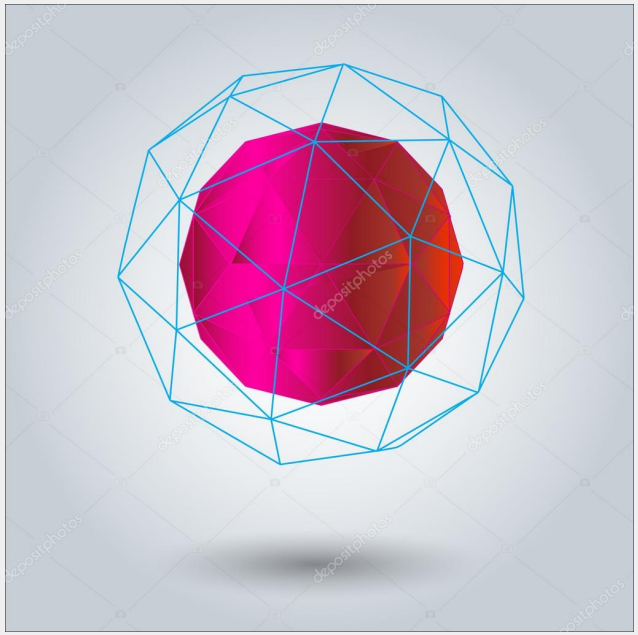
In this exercise, you are asked to utilize the content discussed today and extending it to accommodate the following:

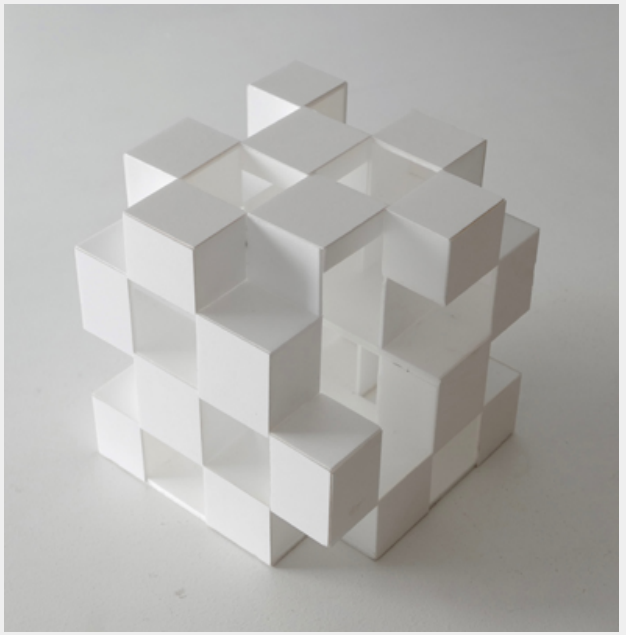
- Create a composition that consists of multiple geometric objects
- Each object (or groups of objects) have different material
- Each object (or groups of objects) have a different position, rotation, motion

This exercise is part of your work, and it is needed to be demonstrated (GitHub).

Evidence: creativity, aesthetic design skills, and experimentation.







Resources (books)