Three.js : Creator Ricardo Cabello (mrDoob)

Javascript API for WebGL
Results can be high or low level, in canvas or SVG
Compatible with most modern browsers
Hooks into the GPU

CPU can render few calculations very quickly, while GPU does it slowly however with far more parallel processing concurrently

Paints Triangles are drawn and then the pixels therein are colored according to the material shader

github.com/mrdoob/three.js/

Basic Scene

github.com/nickolasnikolic/meetup

index.html

main.js
- Scene
- Camera
- Renderer
  - geometry
  - material shader

THREE variable lots of classes

Scene
- Like a container
- Objects, lights, models, particles
- Rendered at output

Mesh
- Combination
  - geometry
  - material
    - color in hex 0x ffffff = white in RGB

Easiest thing to forget is to forget to add the mesh object to scene. Next, lights. Lastly, camera.

Camera
- point of view
- can have many cameras and swith them programmatically
- Kinds
  - perspective → similar to eye w/ adjustible mm
  - box → isometric

Perspective Camera (positionable too)

└ Main
└ in degrees
└ Aspect ratio adjustible
└ Size → "Size" of image → stretchable to viewport

(Aspect ratio is width/Height)

Renderer takes element to transform into canvas and renders objects

---

Loading Modules triggers CORS restrictions

To load needed additional modules, run a server.

Bundler needed to follow imports to new files
Webpack is dominant

Textures and othe files are needed for Threejs to do its work.

Transformation

- Position
- Scale
- Rotation
- Quarternion

All classes that inheirit from Object3D possess these properties. These classes include PerspectiveCamera and Mesh

These properties are agregated in matrices. All general functions use the matrix.

Animation

Like stop motion technique

- Move
- Take Picture

In order to refresh the screen, you use requestAnimationFrame(). This asks the computer to refresh the area you are using in the viewport. Consequently, requestAnimationFrame triggers during the frame and displays the next frame.

The pattern used is to set a function that again calls itself using requestAnimationFrame(); ticking away at the speed of ~60 fps

☆ Ironically, don't forget to first call the recursive function

Cameras

- Camera: Abstract class used to build other cameras
- Array Camera: Multiple cameras in a grouped set
- StereoCamera: 2 cameras; used in VR
- Cube Camera: 6 cameras; used in environment maps
- OrthographicCamera: no perspective

☆ - PerspectiveCamera: normal 75° **Vertical** is default

Geometries

Composed of verticios and faces. A vertex is the corner/point wheras a face is edge to edge.

Every Vertex has:
- position
- UV (Texture coordinates)
- Normal (Forward face?)

Types:
- Box - Plane - Circle - Cone - Cylinder
- Ring - Torus - TorusKnot - Dodecahedron

Textures

Images stretched or laid on the surface of the geometry (see 3D textures, me)

Materials

Put color on each visible pixel
They also include shaders

Shaders (Advanced Topic)

A set (at least 2) of programs written in GLSL that are sent to the GPU directly. GLSL is a subset of c in c syntax.
The 2 are:

1 - Vertex Shader - Position the verticies differently than the model
2 - Face Shader - Colorize the pixels within a face of the model. Bruno Simon specialized the word pixels and prefers "Fragment"

Shaders get and process a lot of data:

- Vertices Coords | • Mesh Transformation
- Camera Info • Color • Texture • Lights • Fog • etc

Shaders always present in interpreting model and space.

Vertex shaders position every vertex. Commonly, vertices are simply funneled through. But, that need not be the case

Uniforms are the same data shared between every vertex.

Attributes are also data held by a vertex, but may be individual

Fragment (Face) Shaders Colorsvisible fragments of the geometry