University of Colorado
Boulder

Introduction to Virtual Reality

Intro to WebGL and Three.js

Professor **Dan Szafir**

Computer Science & ATLAS Institute
University of Colorado Boulder

OpenGL

Graphics programming API introduced in the 1990s

THE standard for graphics in the 1990s and 2000s

Lets users better interface 3D models with graphics hardware for interactive applications

WebGL

Making OpenGL more portable

JavaScript API for rendering any interactive 3D or 2D graphics using HTML5

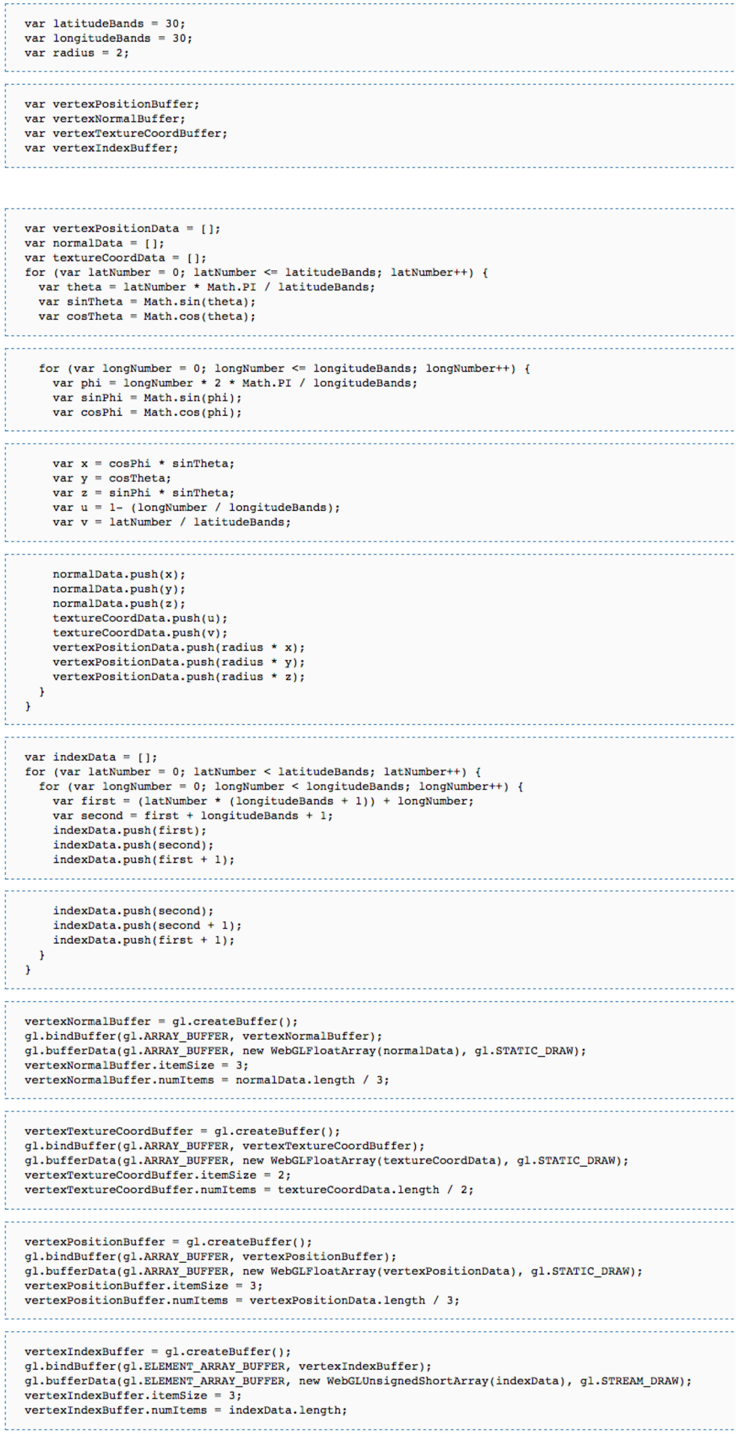
Render complex graphics directly in the browser

Three.JS

Wrapper over WebGL

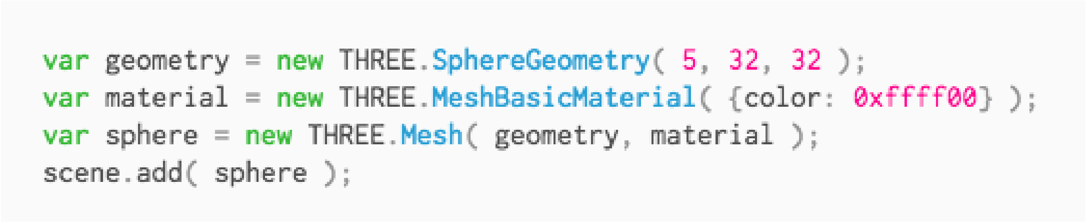
Makes WebGL WAY more intuitive…

Draw a sphere in WebGL

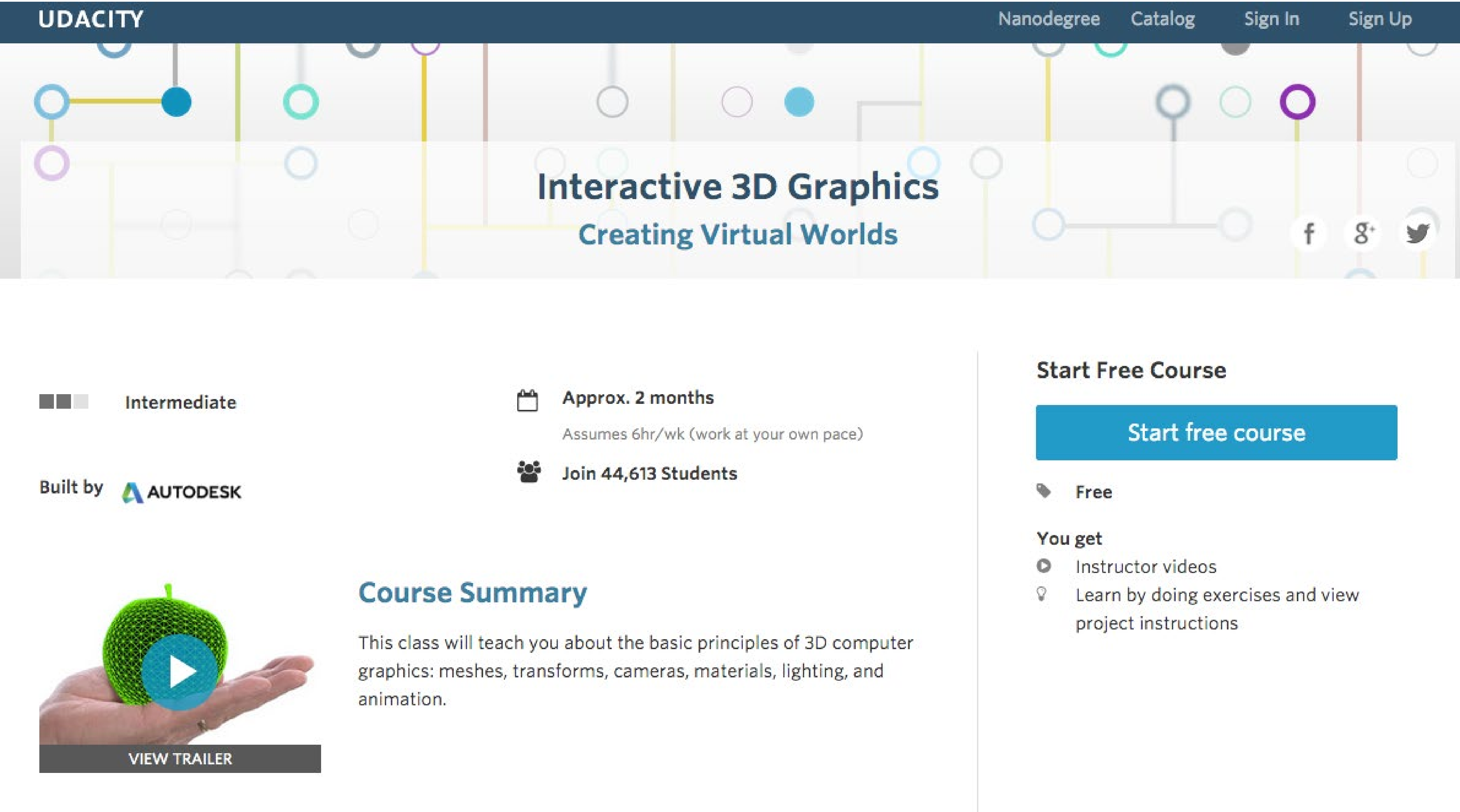


```
var latitudeBands = 30;
var longitudeBands = 30;
var radius = 2;

var vertexPositionBuffer;
var vertexNormalBuffer;
var vertexTextureCoordBuffer;
var vertexIndexBuffer;

var vertexPositionData = [];
var normalData = [];
var textureCoordData = [];
for (var latNumber = 0; latNumber <= latitudeBands; latNumber++) {
  var theta = latNumber * Math.PI / latitudeBands;
  var sinTheta = Math.sin(theta);
  var cosTheta = Math.cos(theta);

  for (var longNumber = 0; longNumber <= longitudeBands; longNumber++) {
    var phi = longNumber * 2 * Math.PI / longitudeBands;
    var sinPhi = Math.sin(phi);
    var cosPhi = Math.cos(phi);

    var x = cosPhi * sinTheta;
    var y = cosTheta;
    var z = sinPhi * sinTheta;
    var u = 1- (longNumber / longitudeBands);
    var v = latNumber / latitudeBands;

    normalData.push(x);
    normalData.push(y);
    normalData.push(z);
    textureCoordData.push(u);
    textureCoordData.push(v);
    vertexPositionData.push(radius * x);
    vertexPositionData.push(radius * y);
    vertexPositionData.push(radius * z);
  }
}

var indexData = [];
for (var latNumber = 0; latNumber < latitudeBands; latNumber++) {
  for (var longNumber = 0; longNumber < longitudeBands; longNumber++) {
    var first = (latNumber * (longitudeBands + 1)) + longNumber;
    var second = first + longitudeBands + 1;
    indexData.push(first);
    indexData.push(second);
    indexData.push(first + 1);

    indexData.push(second);
    indexData.push(second + 1);
    indexData.push(first + 1);
  }
}

vertexNormalBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexNormalBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new WebGLFloatArray(normalData), gl.STATIC_DRAW);
vertexNormalBuffer.itemSize = 3;
vertexNormalBuffer.numItems = normalData.length / 3;

vertexTextureCoordBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexTextureCoordBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new WebGLFloatArray(textureCoordData), gl.STATIC_DRAW);
vertexTextureCoordBuffer.itemSize = 2;
vertexTextureCoordBuffer.numItems = textureCoordData.length / 2;

vertexPositionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexPositionBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new WebGLFloatArray(vertexPositionData), gl.STATIC_DRAW);
vertexPositionBuffer.itemSize = 3;
vertexPositionBuffer.numItems = vertexPositionData.length / 3;

vertexIndexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, vertexIndexBuffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new WebGLUnsignedShortArray(indexData), gl.STREAM_DRAW);
vertexIndexBuffer.itemSize = 3;
vertexIndexBuffer.numItems = indexData.length;
```

Draw a sphere in Three.JS



```
var geometry = new THREE.SphereGeometry( 5, 32, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var sphere = new THREE.Mesh( geometry, material );
scene.add( sphere );
```



Interactive 3D Graphics

Creating Virtual Worlds

Intermediate

Built by AUTODESK

Approx. 2 months

Assumes 6hr/wk (work at your own pace)

Join 44,613 Students

Course Summary

This class will teach you about the basic principles of 3D computer graphics: meshes, transforms, cameras, materials, lighting, and animation.

Start Free Course

Free

You get

- Instructor videos
- Learn by doing exercises and view project instructions

Today: Three.JS

Components of an Application

Setting up an Image

Creating an Image

Time-Permitting: Animation

Today: Three.JS

Components of an Application

Setting up an Image

Creating an Image

Time-Permitting: Animation

What pieces do we need to build a graphics app?

- Scene
  - Holds the image
- Renderer
  - Translates the image
- Camera
  - Viewpoint in 3D space
- Objects
  - Geometries in the image
- Lighting
  - Illuminates the scene

Today: Three.JS

Components of an Application

Setting up an Image

Creating an Image

Time-Permitting: Animation

Components of a Scene in Three.JS

Container for the image

Links the content of the image to a place in the webpage

REMEMBER: Everything you build in a scene needs to be explicitly added to a scene!

Specify the scene

WebGL Exercises on Moodle

- Exercise 1: Rendering a Sphere #1-4

Coding exercise inspired by Aerotwist:
https://aerotwist.com/tutorials/getting-started-with-three-js/

Camera Attributes

Aspect ratio

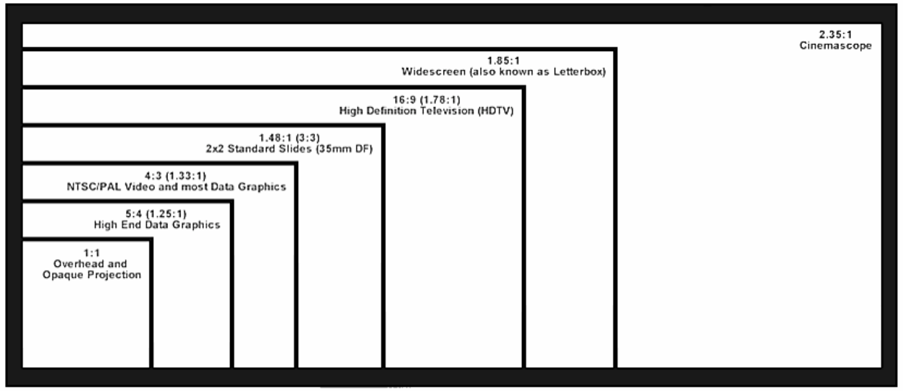
Clipping Planes

Projection

Position, Target, & Angle

Aspect Ratio

What is a good aspect ratio?


2.35:1
Cinemascope
1.85:1
Widescreen (also known as Letterbox)
16:9 (1.78:1)
High Definition Television (HDTV)
1.48:1 (3:3)
2x2 Standard Slides (35mm DF)
4:3 (1.33:1)
NTSC/PAL Video and most Data Graphics
5:4 (1.25:1)
High End Data Graphics
1:1
Overhead and
Opaque Projection

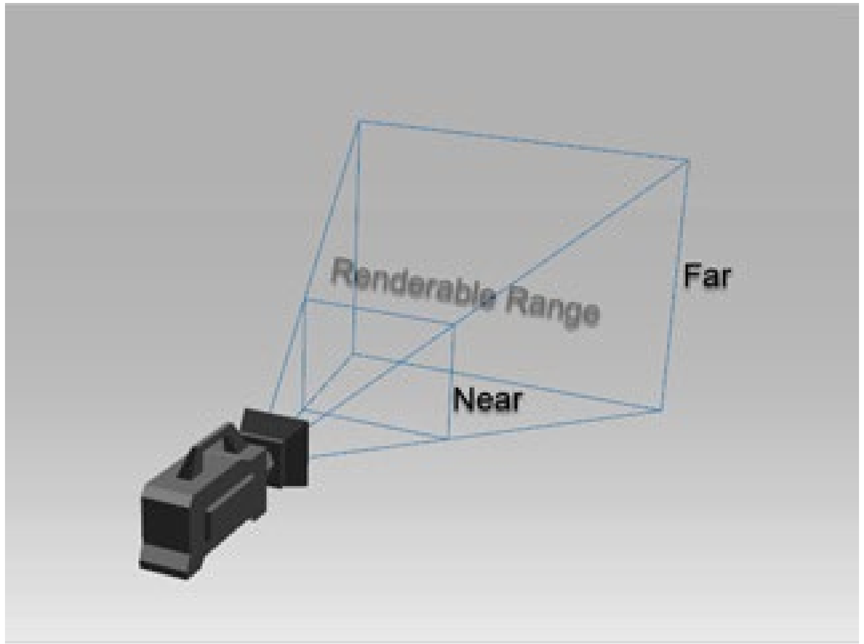
Width / Height (1:1)

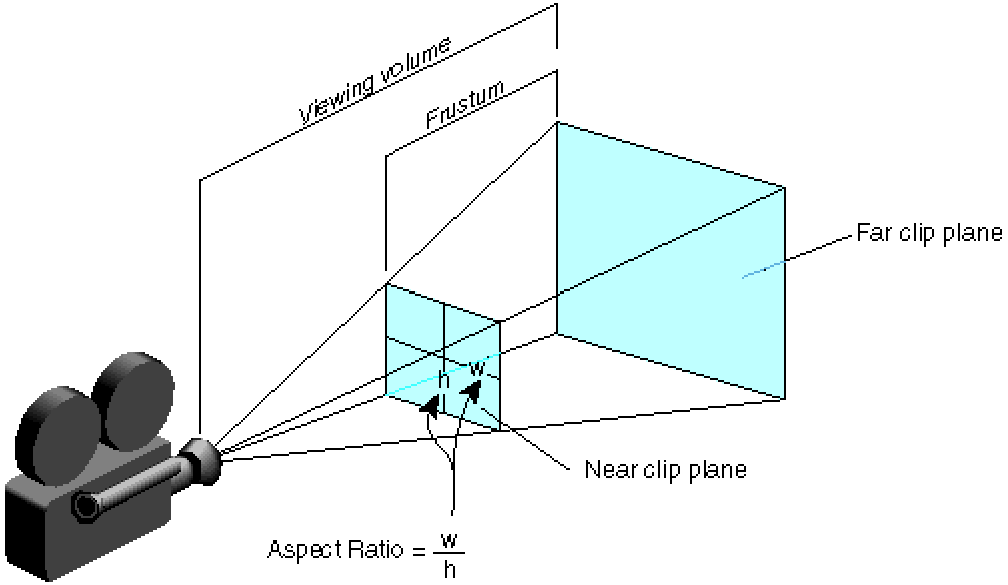
Otherwise, will squash or stretch the image accordingly

Clipping Planes

To reduce computation, cut out things that are too close and too far

Objects between planes are projected on to the near clipping plane





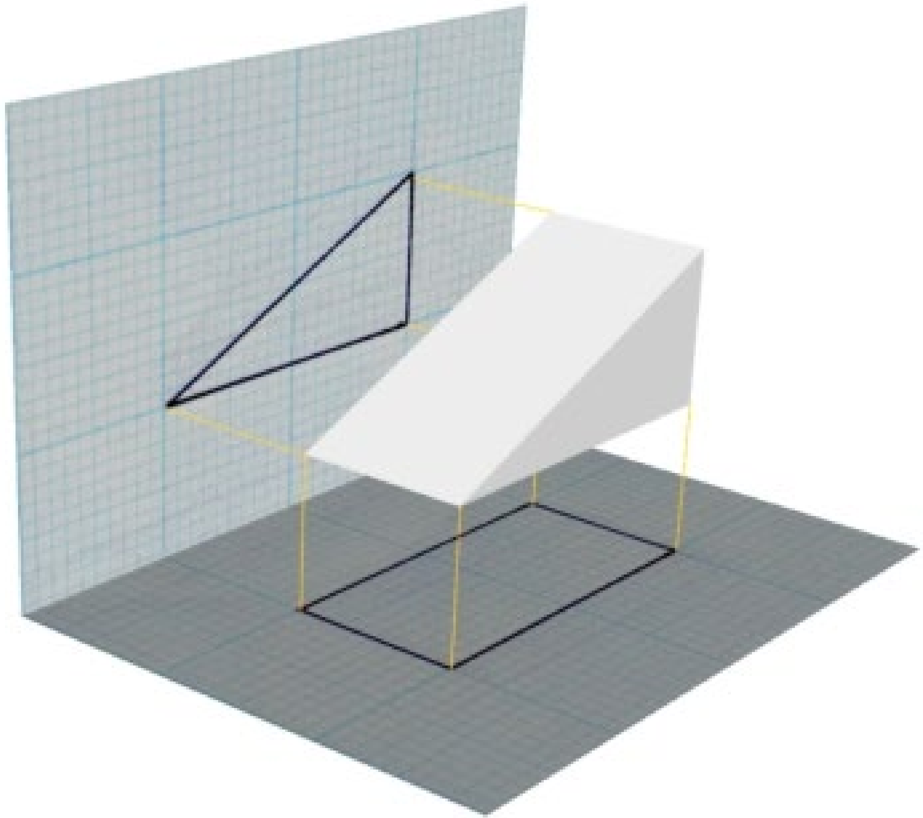
How we go from 3D to a 2D monitor?

Projection: Function that transforms points from 3D to 2D space

2 types: orthographic or perspective

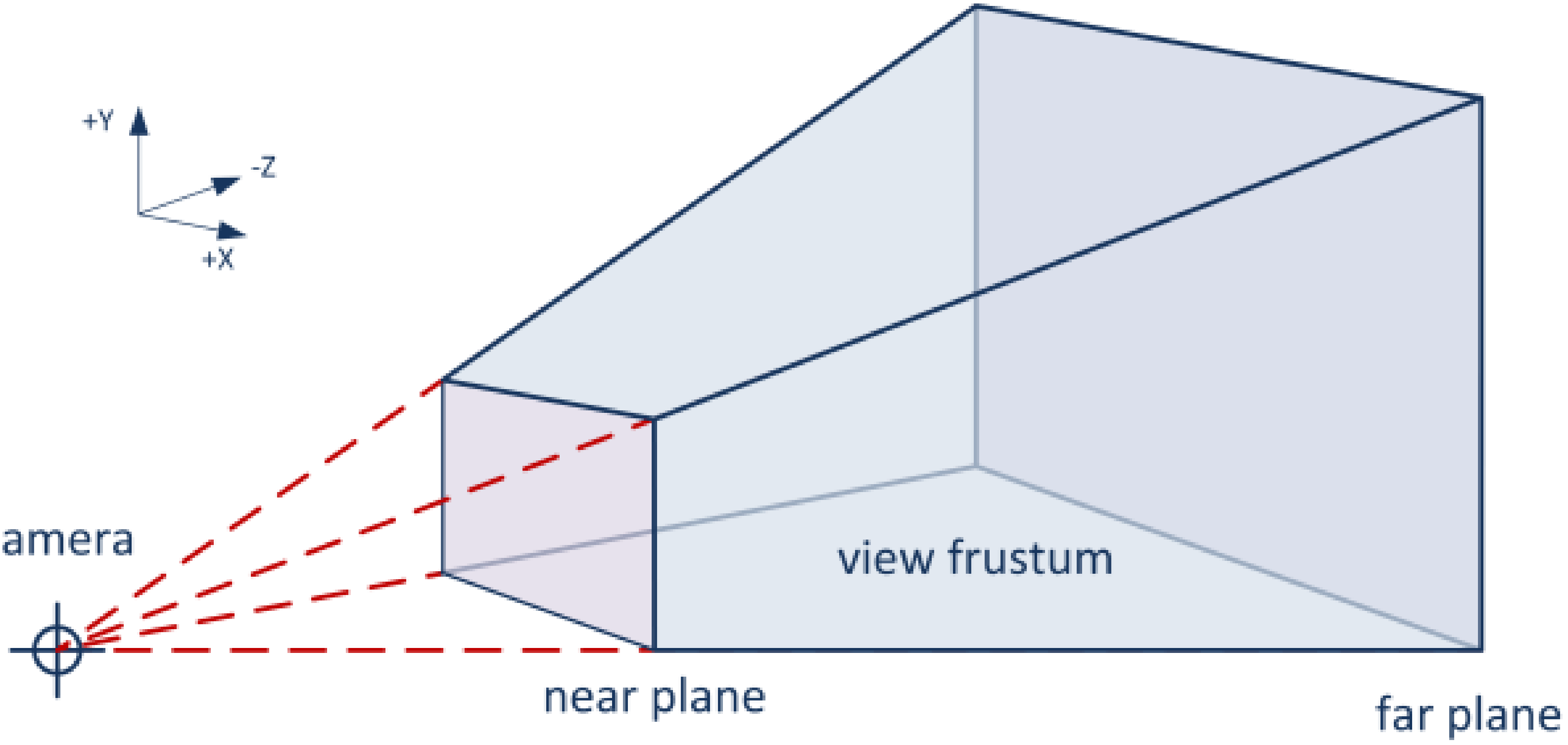
Orthographic Projection

Objects project to a 2D plane regardless of how far they are from the camera

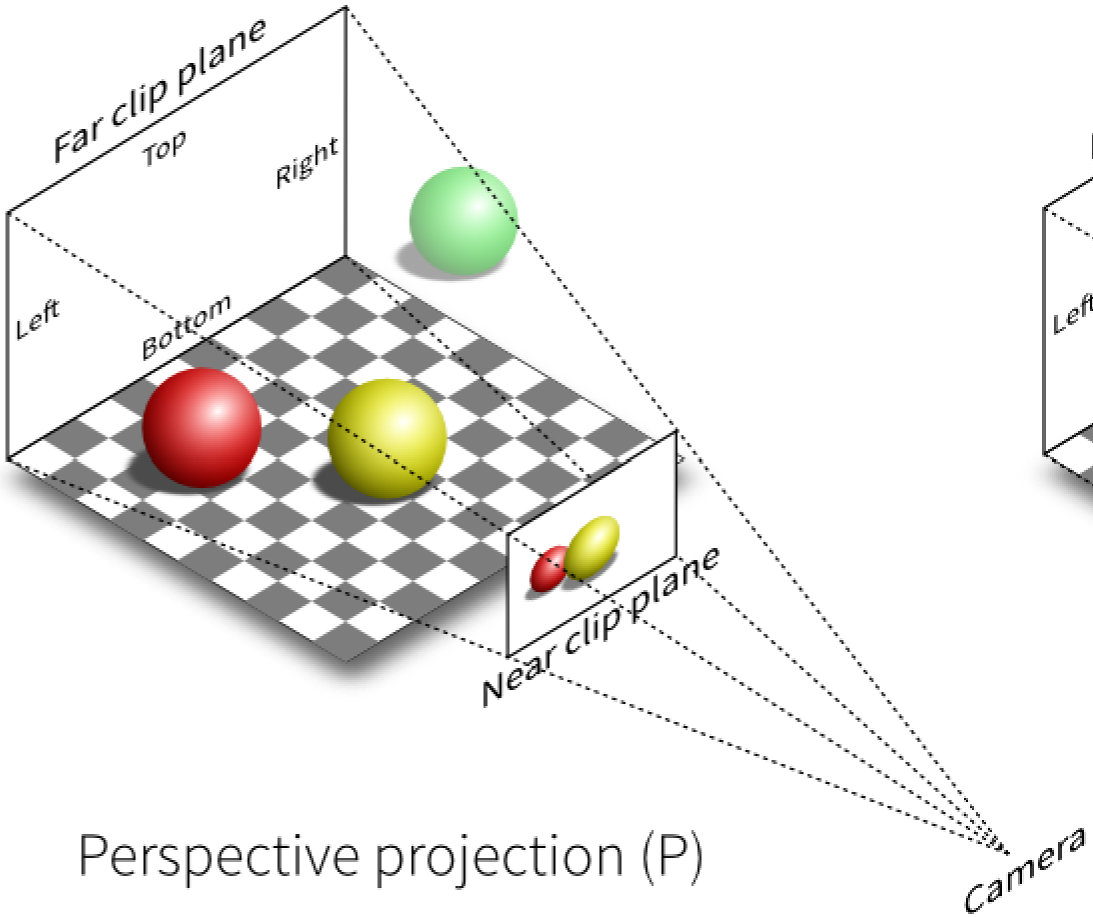


$$P = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix}$$

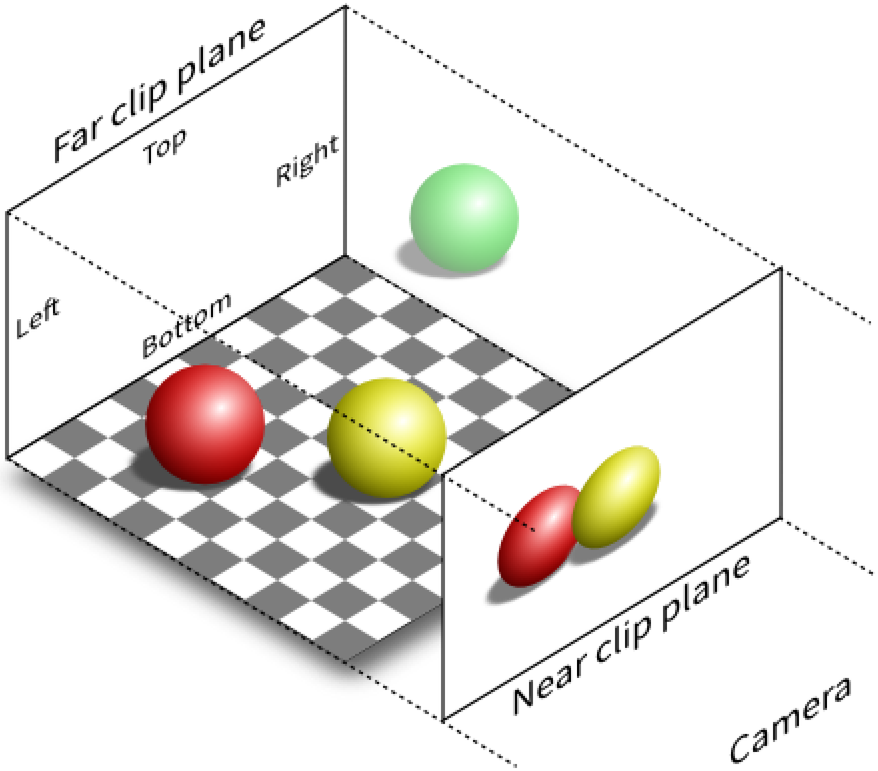
Perspective Projection



$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



Perspective projection (P)



Orthographic projection (O)

When might I use orthographic projection?

When might I use perspective projection?

Camera in Three.js

WebGL Exercises on Moodle

- Exercise 1: Rendering a Sphere #5-7

Renderers

Translate Three.js code to an image

Three types:

- WebGLRenderer
- CanvasRenderer
- SVGRenderer (we’ll go over SVGs later)

How do we use renderers in Three.js?

WebGL Exercises on Moodle

- Exercise 1: Rendering a Sphere #8-10

Today: Three.JS

Components of an Application

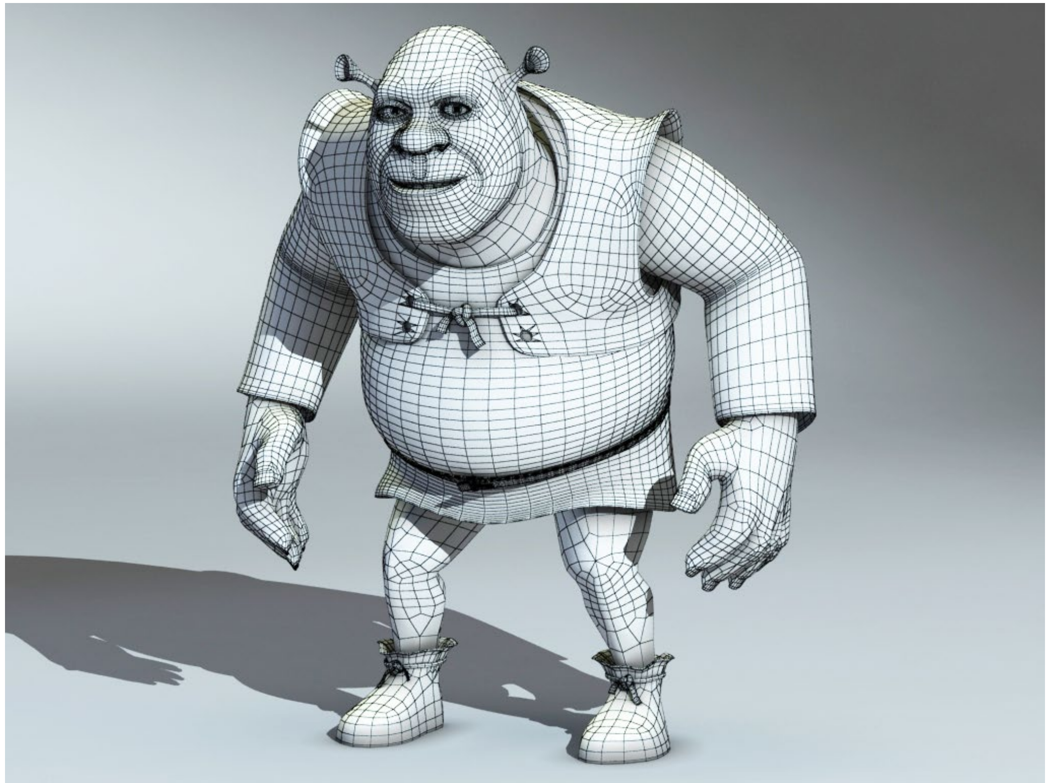
Setting up an Image

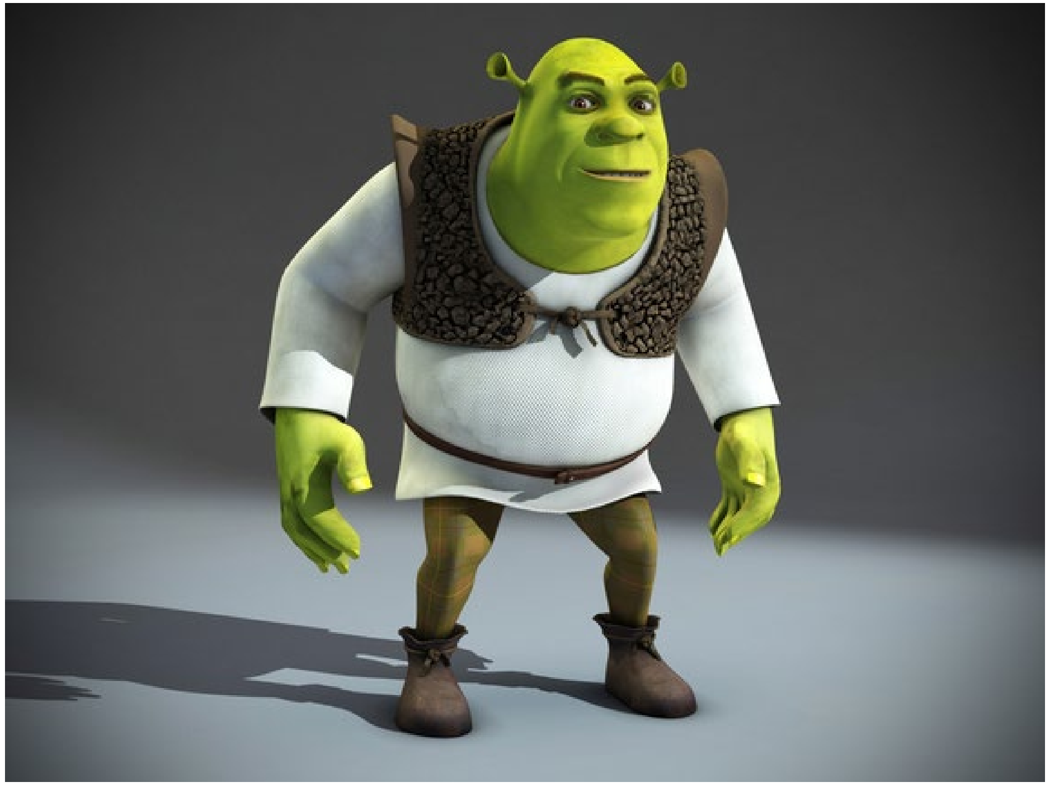
Creating an Image

Time-Permitting: Animation

Building things!

Things have a structure (mesh) and an appearance (material)





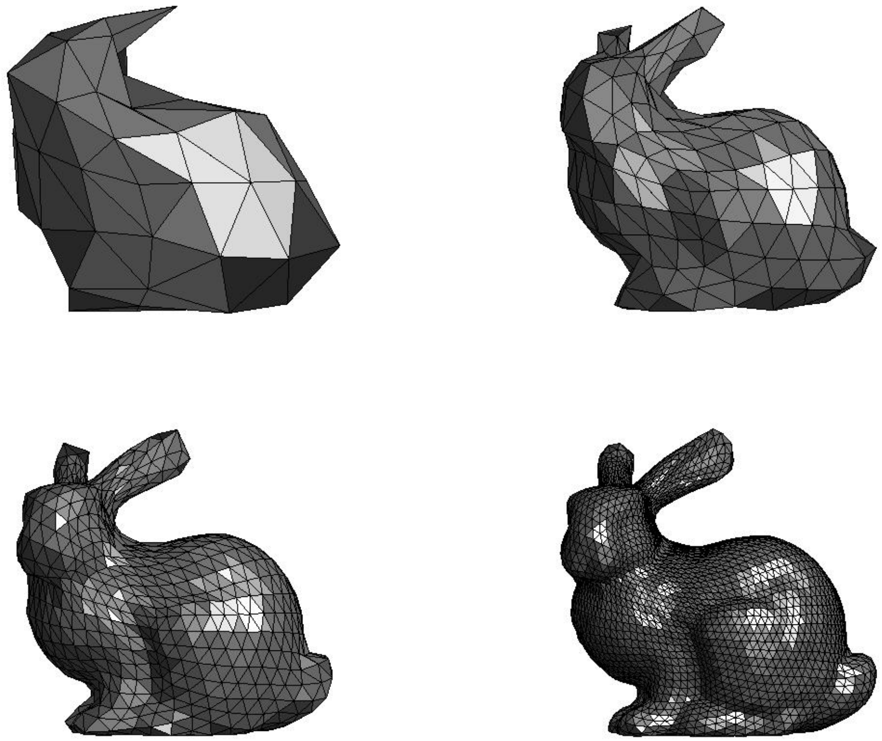
Meshes

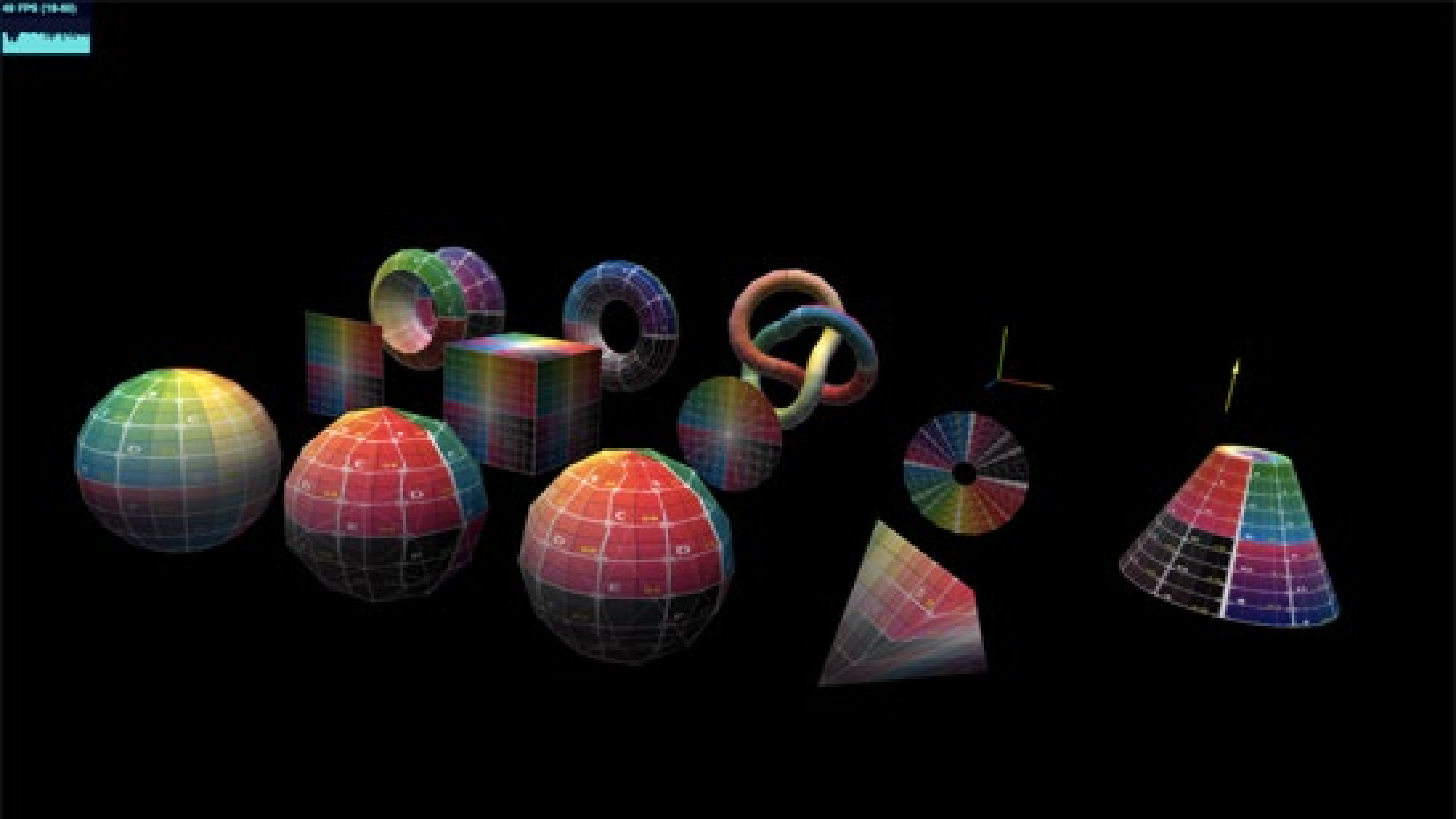
Define the shape of an object

Generally triangles

Vertices: Points in a mesh

Edges: Lines connecting those points

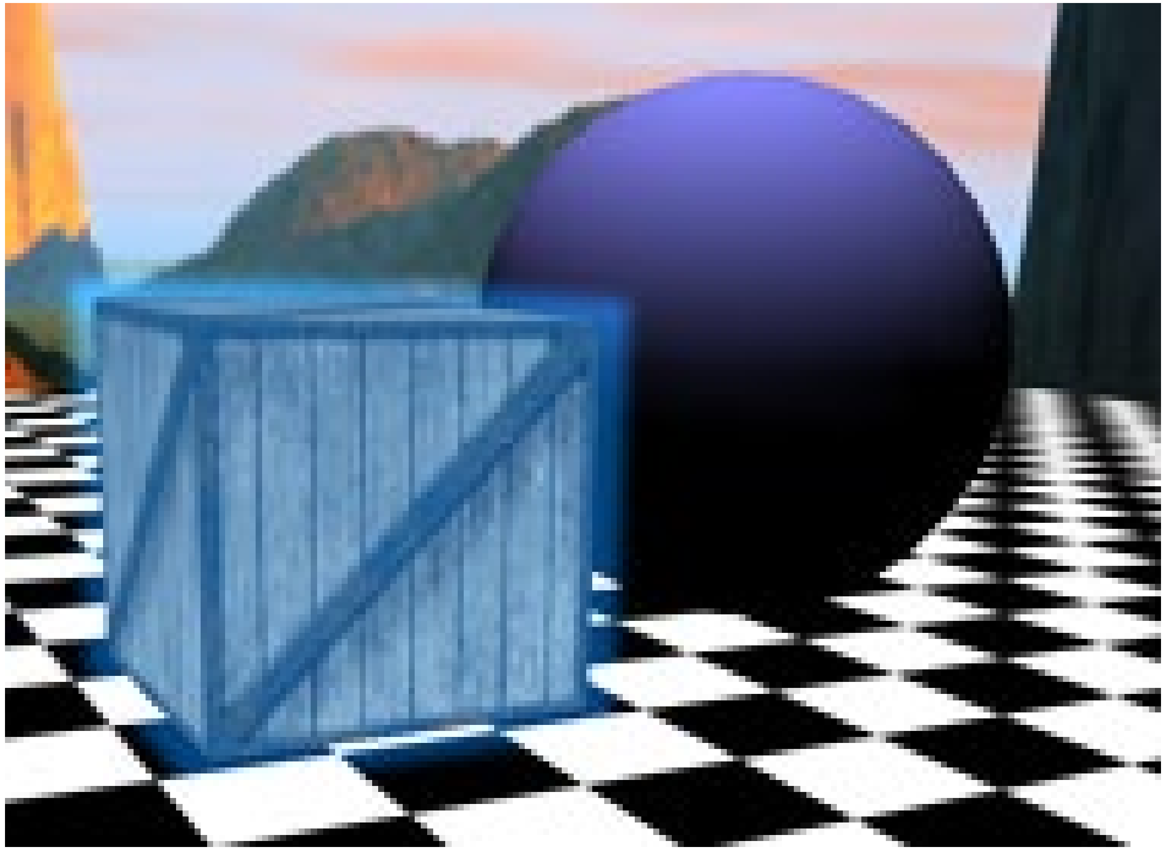



48 FPS (19-60)

Materials

“Skin” applied to a mesh

Gives the mesh color, texture, and reflective properties



Add a sphere

WebGL Exercises on Moodle

- Exercise 1: Rendering a Sphere #11-15

Can add more complex models from other programs…

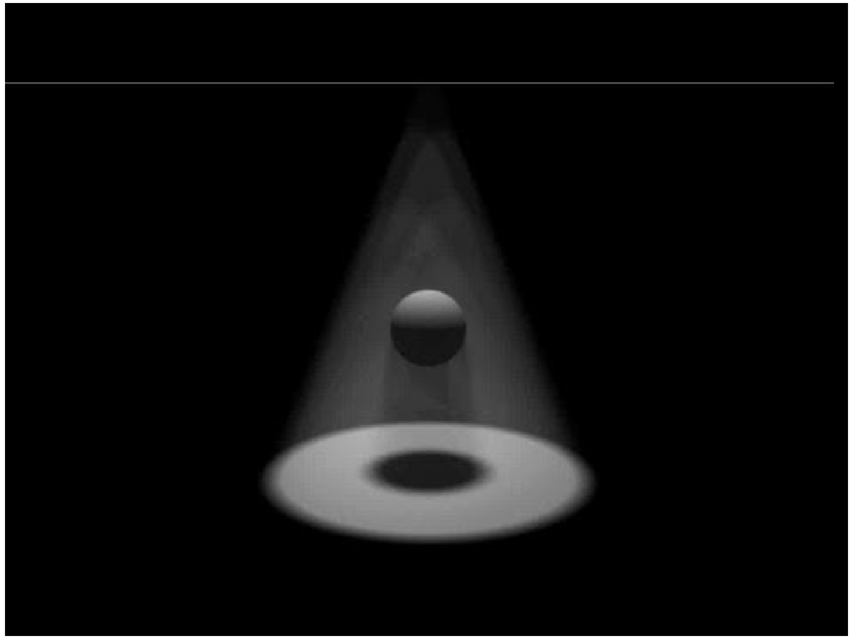
http://threejs.org/examples/webgl_loader_obj.html

We will go over this later, once we have created our own models

Lighting

Ambient light: Lighting from everywhere

Point light: light from a single point



Adding a point light

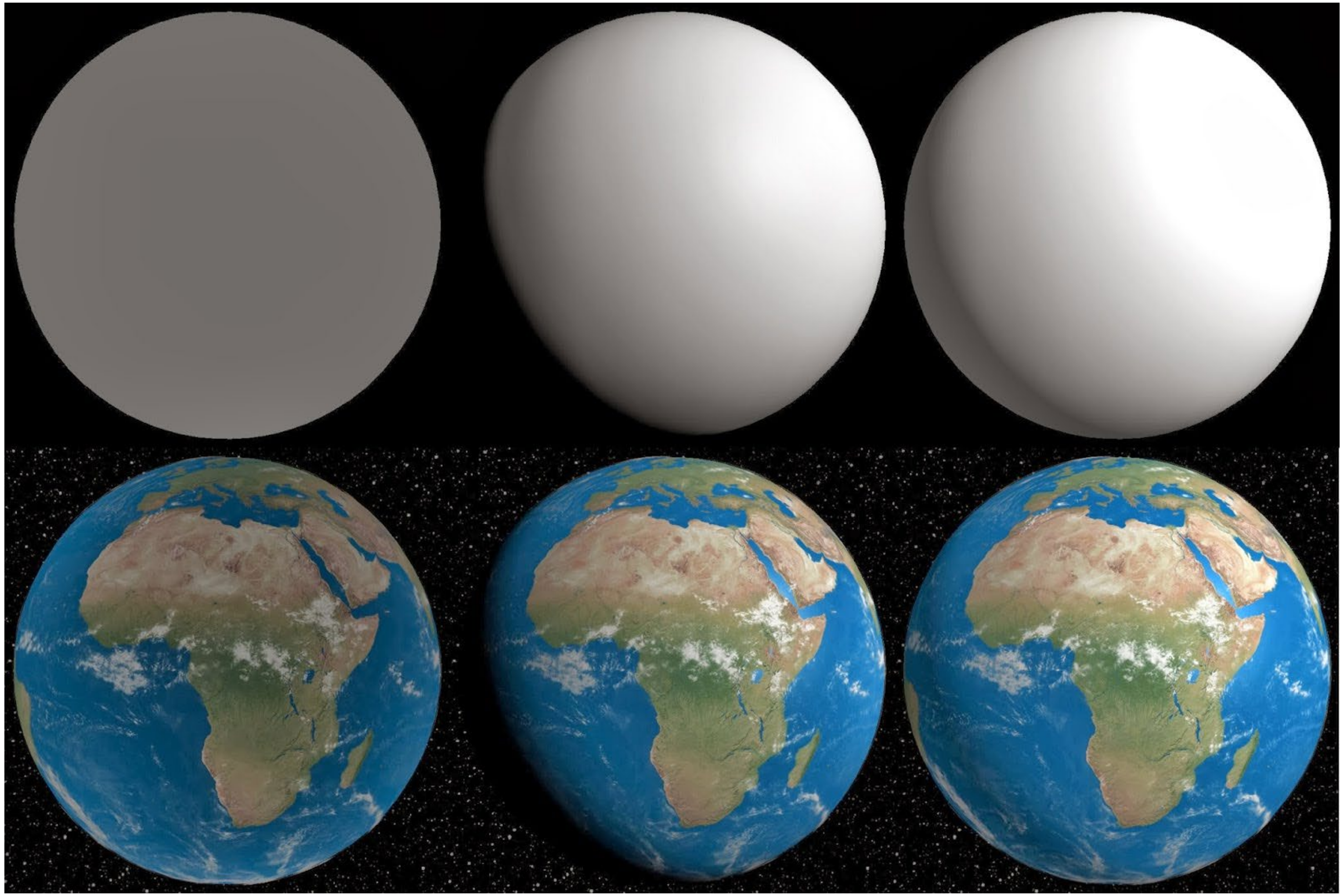
http://threejs.org/examples/#webgl_lights_pointlights

WebGL Exercises on Moodle

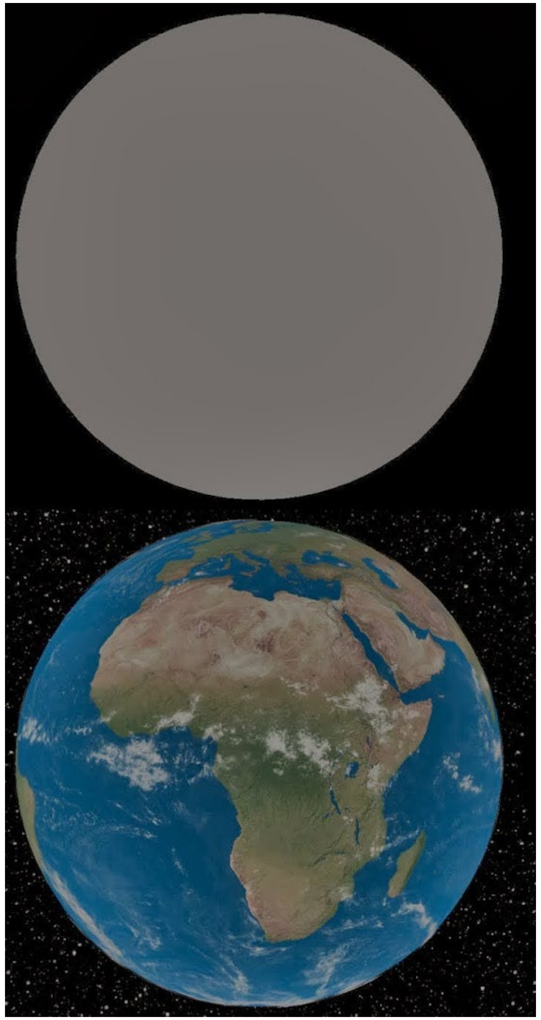
Exercise 1: Rendering a Sphere #16-18

What part of a scene responds to lighting?

The material responds to the light

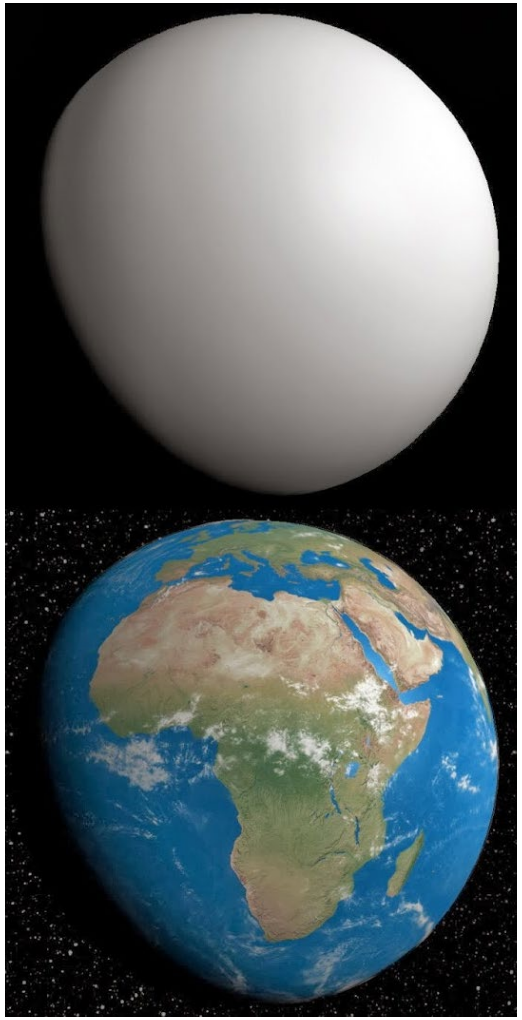


MeshBasicMaterial



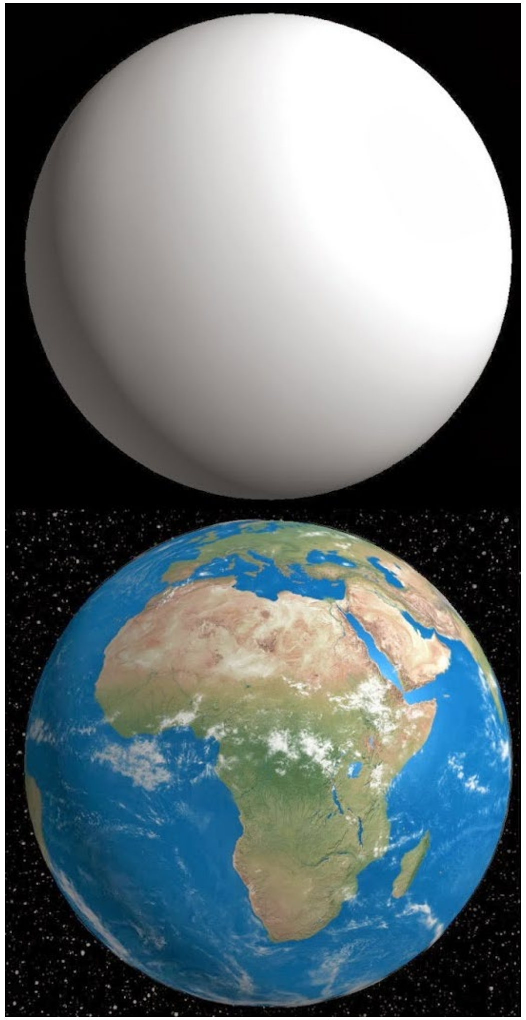
Reflects light equally in all directions

MeshLambertMaterial



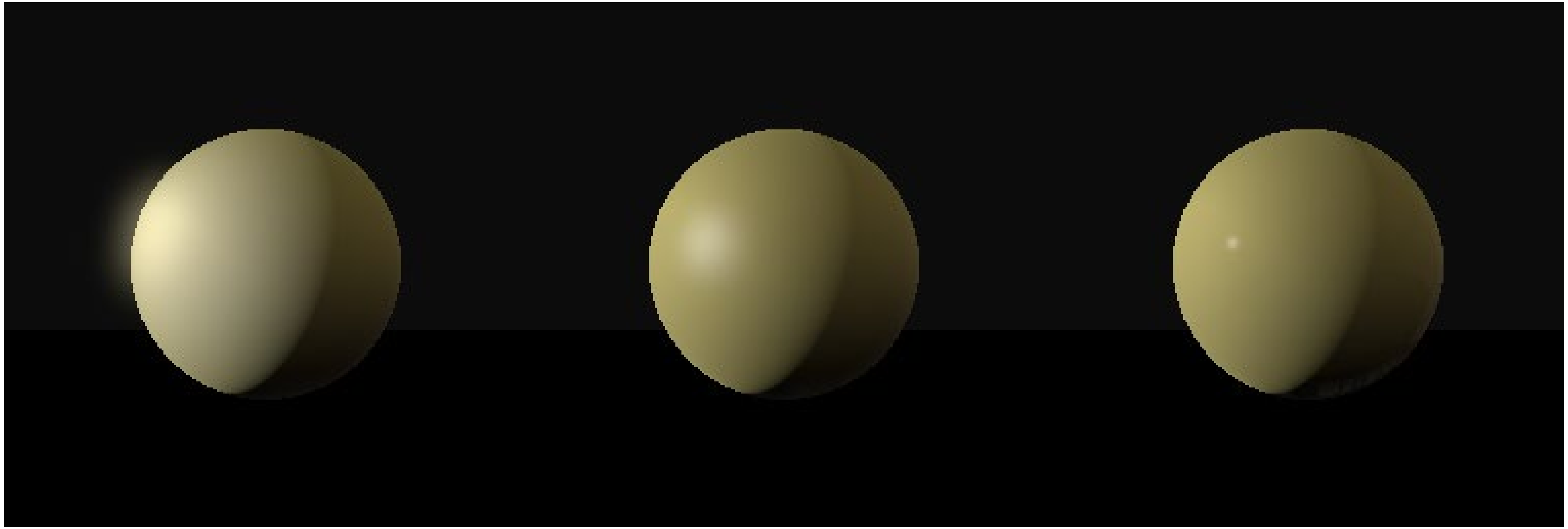
Reflects light diffusely from the surface

MeshPhongMaterial



Reflects light with a highlight

Tighter highlights appear shinier



Render!

WebGL Exercises on Moodle

- Exercise 1: Rendering a Sphere #19

Today: Three.JS

Components of an Application

Setting up an Image

Creating an Image

Time-Permitting: Animation

Animation

Can change on user inputs or on different timesteps:

- WebGL Exercises on Moodle
  - Exercise 2: Animating a Cube

Great examples plus source code:

http://threejs.org/examples/

Next Class

Graphics Pipeline: How we get from the code to the screen

THANKS!

Professor **Dan Szafir**

Computer Science & ATLAS Institute
University of Colorado Boulder