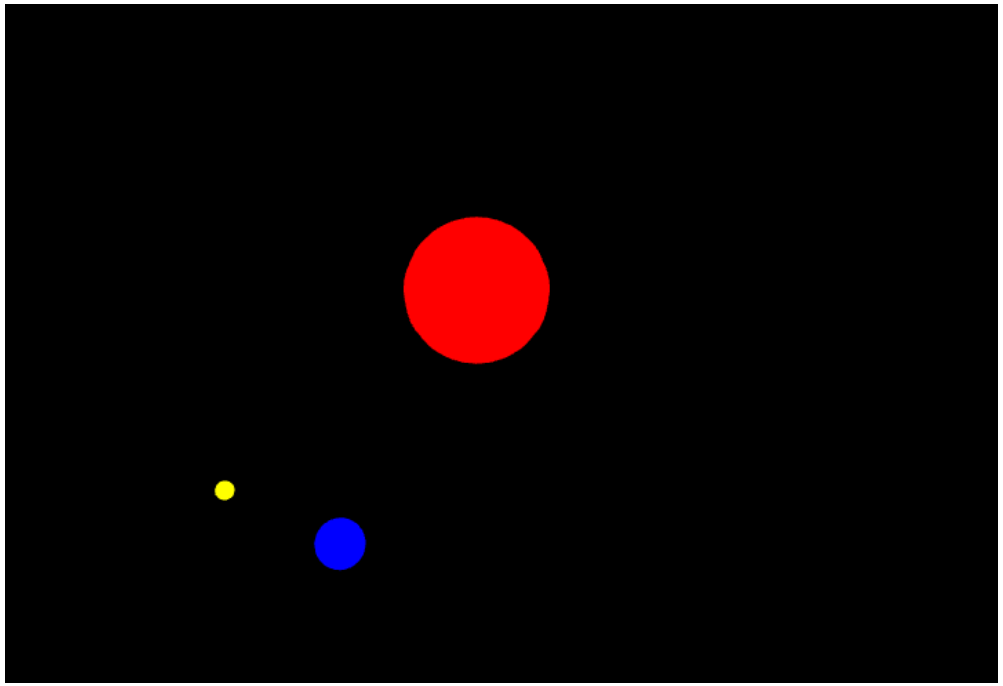
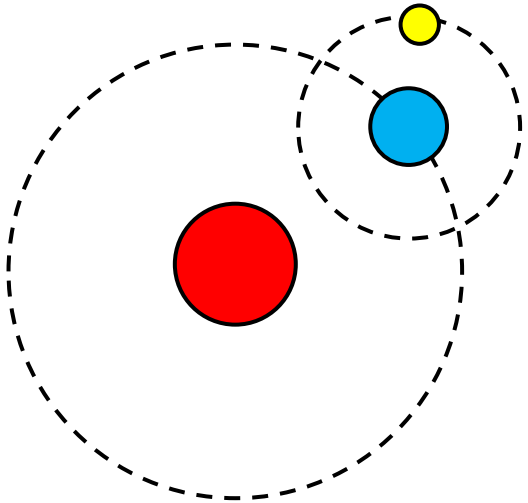
LIGHTING AND TEXTURE IN THREE JS

By Raymond Pang

Planet Orbits Example

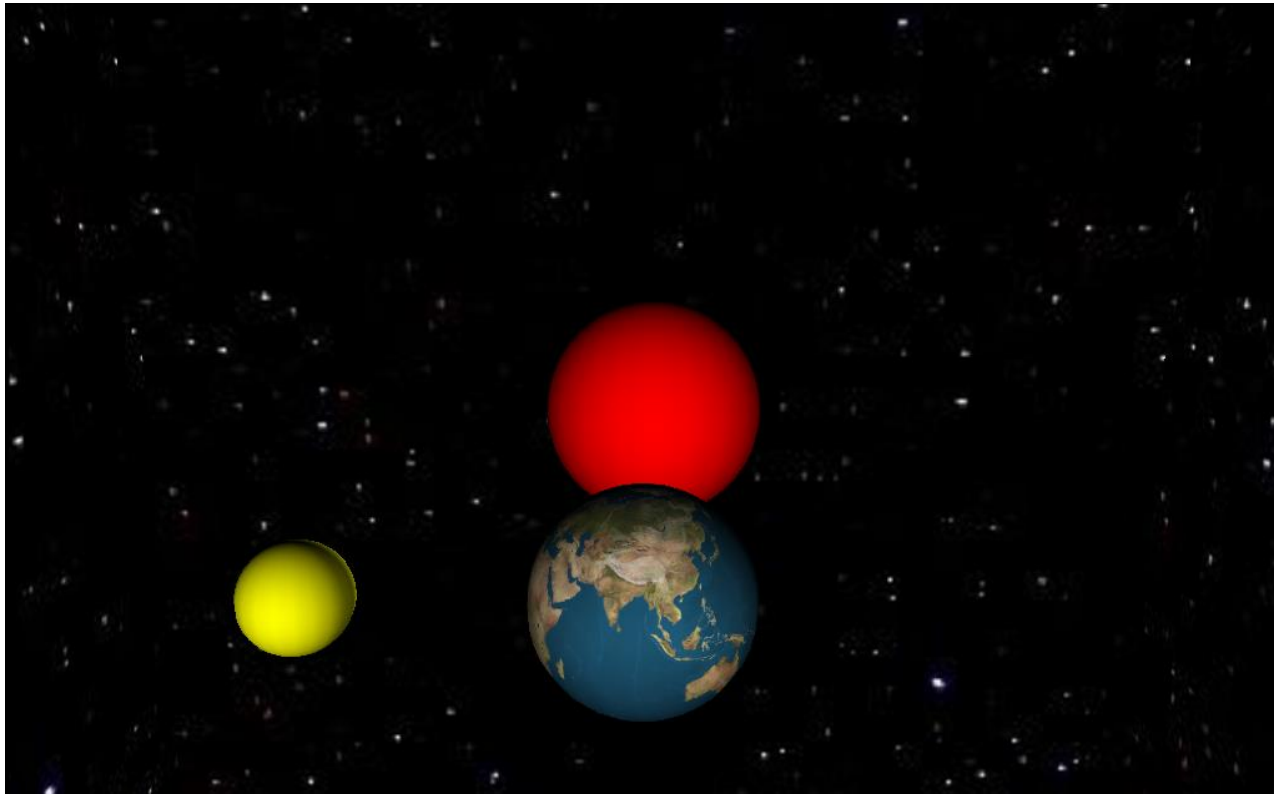
- In last tutorial, we have gone through the "Planet Orbits" sample
- But all the objects are rendered in plain colors





Planet Orbits Example with lighting and texture

- In the coming tutorial, we will add lighting and texture to the planet example, so that the appearance is greatly improved:



Lighting in ThreeJS

Adding Lights in ThreeJS

- As introduced in the lecture, lighting is important for shading and showing shape of objects more clearly
- It is easy to add a simple point light in the scene within ThreeJS
- Use “THREE.PointLight”, and add it to the scene graph

```
var light = new THREE.PointLight(0xffffff, 1, 100);
light.position.set(10,10,10);
scene.add(light);
```

Adding Lights in ThreeJS

- The parameters given when creating point light source:
  - Light Color (in hexadecimal)
  - Intensity (1.0 by default)
  - Distance (0.0 by default)
    - The distance by which the attenuation will linearly drop to zero. No attenuation is considered if it is zero.

```
THREE.PointLight(color_in_hex, intensity, distance)
```

Adding Lights in ThreeJS

- In example below, we create
  - Point light with white color, and the maximum intensity is 1.0, with attenuation distance of 100 unit in space

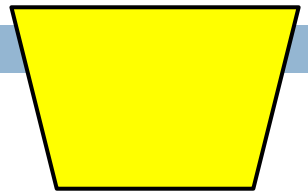
```
var light = new THREE.PointLight(0xffffff, 1, 100);
```

- We can control its location like other 3D objects by

```
light.position.set(10,10,10);
```

- As point light will shine in all directions, it is not very necessary to configure its orientation

Other Lights in ThreeJS



- Area Light
  - Specify Width and Height

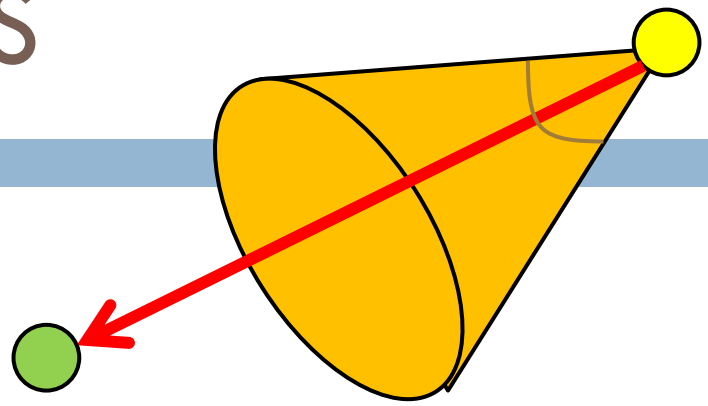
```
var arealight1 = new THREE. AreaLight(0xffffff,10);
areaLight1.width = 10; areaLight1.height = 5;
```

- Directional Light
  - Specify its shining direction (in vector), with the position property

```
var dirLight = new THREE.DirectionalLight(0xffffff, 10 );
dirLight.position.set( 0, 1, 0 );
```

Y-axis (upward direction)

Other Lights in ThreeJS



- Spot Light
- 2 major parameters have to set
  - Cutoff angle (Default — Math.PI/3)
    - in radians, from its direction. Should be no more than Math.PI/2.

```
THREE.SpotLight(color_hex, intensity, distance, angle, exponent)
Or light.angle = angleinRadian
```

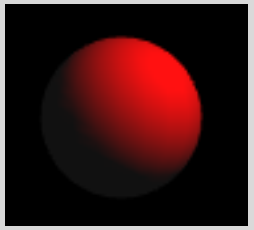
  - Direction
    - Set using spotlight position and target position

```
light.position.set(x,y,z);
light.target.positiion.set(x,y,z);
```

Other Lights in ThreeJS

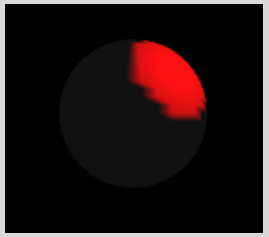
- Spot Light Example
  - A red spot light shining in 45 angle from top right

```
var splight = THREE.SpotLight(0xff0000);
splight.position.set(1,1,1);
splight.position.set(0,0,0);
```



  - Another example with smaller cutoff angle

```
var splight = THREE.SpotLight(0xff0000);
splight.angle = Math.PI / 40.0;
splight.position.set(1,1,1);
splight.position.set(0,0,0);
```



Other Lights in ThreeJS

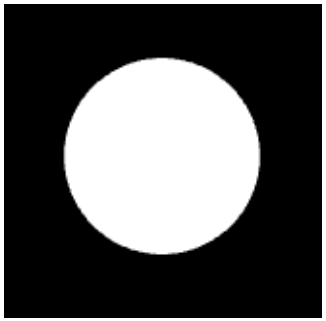
- Ambient Light
  - A separate light to represent all other indirect lightings
  - Also known as background light
  - No need to set its position

```
var alight = THREE.AmbientLight(color_hex);
```

  - The color will affect the base color/brightness of object

0x111111

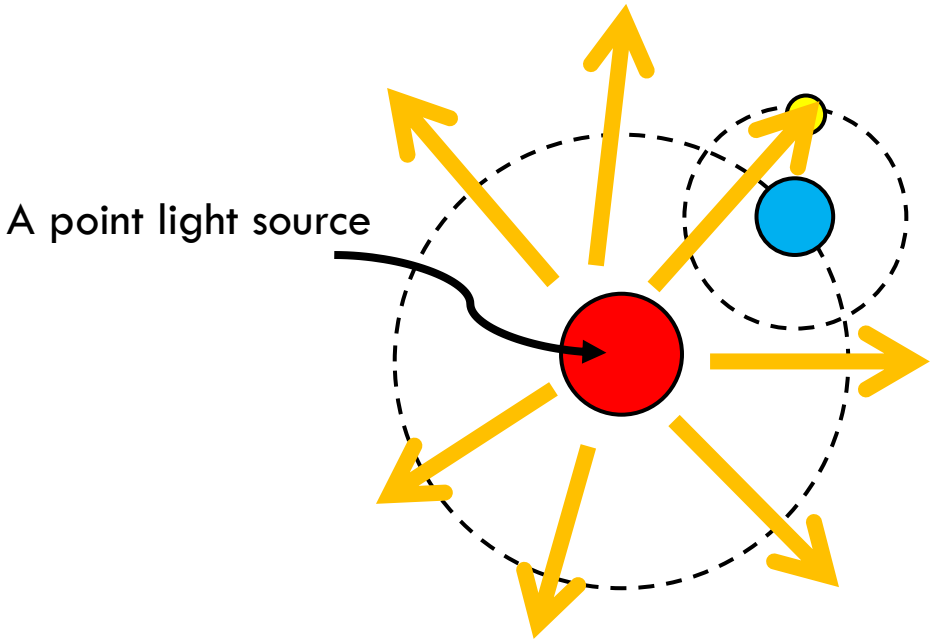
0x33333



0xffffff

Our Planet Sample

- Obviously we can model light emitted from the Sun as a point light source
- A point light shines in all directions equally



Placing Point Light in Our Sample

- As our Sun is placed at the center (origin) of the world coordinate
- We place a point light source at origin, and finally add it to our scene graph

```
var plight = new THREE.PointLight(0xffffff);
plight.position.set(0,0,0);
scene.add(plight);
```

Materials in ThreeJS

- Unfortunately, even we add lights in the scene, nothing seems to happen
- As mentioned in the lecture, proper material properties have to be set on the object surface when using lighting
- In the last lecture, we use only the Basic Material for our planets
- Only plain color is presented

Materials in ThreeJS

- Basic Material
  - A material for drawing geometries in a simple shaded (flat or wireframe) way
  - Only plain colors

```
THREE.MeshBasicMaterial( parameters );
```

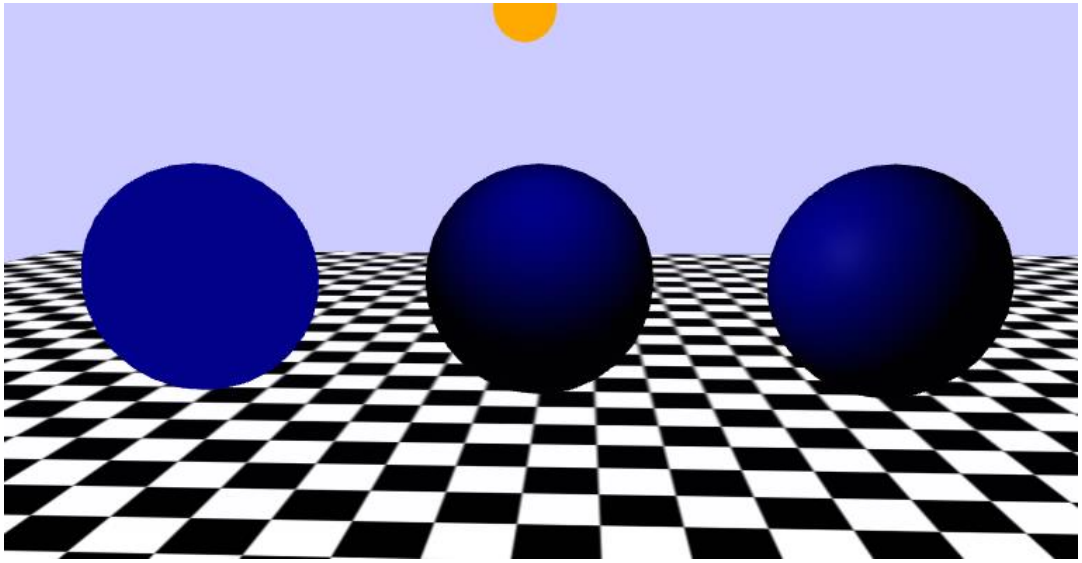
- Lambertian Material
  - A material for non-shiny (Lambertian) surfaces
  - Count only diffuse component

```
THREE.MeshLambertMaterial( parameters );
```

Materials in ThreeJS

- Phong Material
  - Based on the full Phong lighting model
  - Have diffuse and specular components, suitable for shiny surface

```
THREE.MeshPhongMaterial( parameters );
```



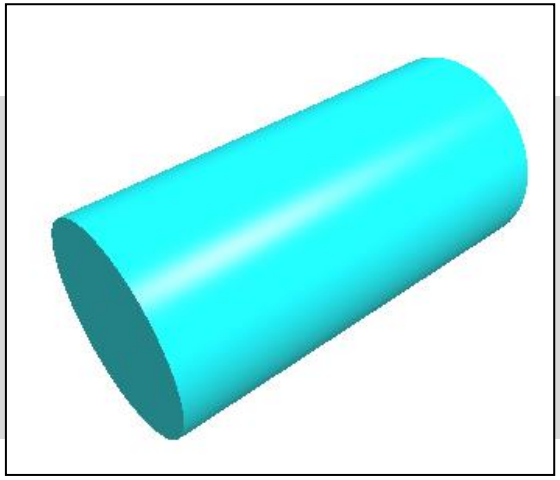
Parameters of Materials

- Large number of parameters in setup of materials, here lists a few major ones
  - Color
    - Diffused color for the material
  - Ambient
    - Ambient color of the material, multiplied by the color of the AmbientLight
  - Specular
    - Color for specular component
  - Shininess
    - How shiny the specular highlight is; higher value gives sharper highlight
  - Shading
    - Possible Values: NoShading, FlatShading, SmoothShading

Parameters of Materials

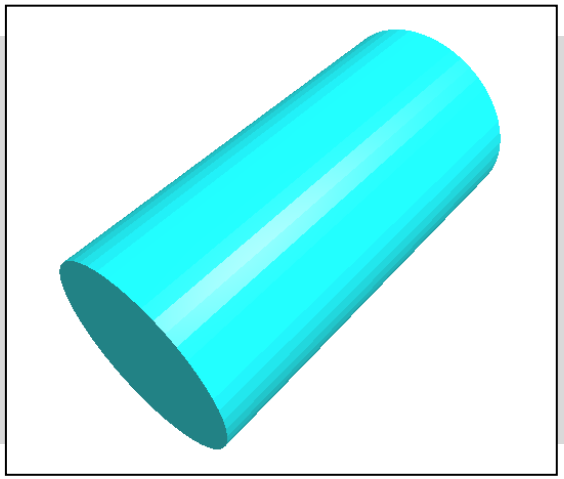
- Example

```
new THREE.MeshPhongMaterial( {
        specular: '#a9fcff',
        color: '#00abb1',
        emissive: '#006063',
        shininess: 100 } )
```



- If we use FlatShading instead:

```
new THREE.MeshPhongMaterial( {
        specular: '#a9fcff',
        color: '#00abb1',
        emissive: '#006063',
        shininess: 100
        shading: THREE.FlatShading } )
```



Our Planet Sample

- Back to our planet sample, we modify our planets sample with Lambertian material
- We take the Sun as an example, other two planets are similar

```
var sunmaterial =  new
THREE.MeshLambertMaterial( { color: 0xff0000 });

sunMesh = new THREE.Mesh(new
THREE.SphereGeometry(2, 20, 20), sunmaterial);
```

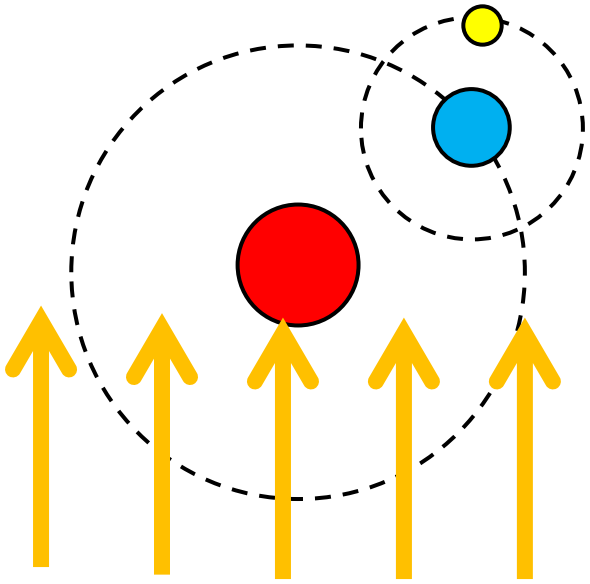
- However, as light is from center of the Sun, it can not shine surface of the Sun properly

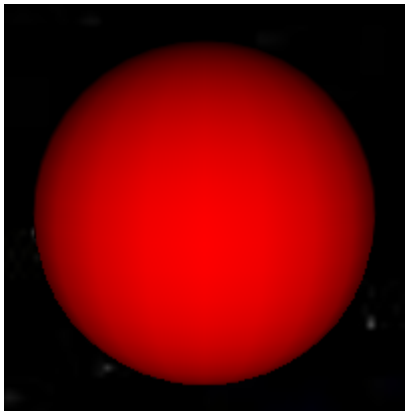
Our Planet Sample

- So, we add one more light as follow

```
var dlight = new THREE.DirectionalLight(0xffffff);
dlight.position.set(0,0,1); //+ve Z
scene.add(dlight);
```

- It is a directional light shining in inward to the scene

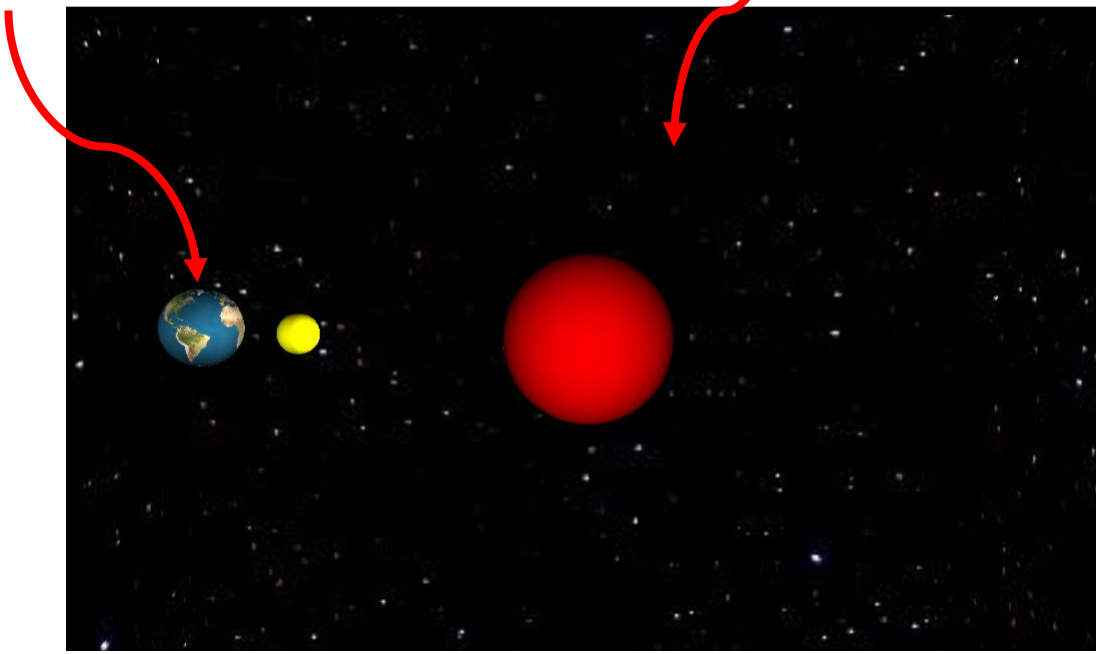




Texture in ThreeJS

Applying Texture in JOGL

- In the planet sample program, we will apply texture mapping in following places:
  - Creating the star field sky box
  - Texturing the Earth



Texture Mapping in ThreeJS

- An attribute in material : **.map**
  - Set the color texture for texture mapping
- But we need to load in a texture from an image file
  - Using ImageUtils.loadTexture(…)

```
ImageUtils.loadTexture(url, mapping, onLoad, onError)
```

- url: where the texture locate
- Mapping: Type of mapping, values can be Three.UVMapping,THREE.CubeReflectionMapping, THREE.SphericalReflectionMapping orTHREE.SphericalRefractionMapping
- onLoad: callback function
- onError: callback funtction

Texture Mapping in ThreeJS

- Take the earth texture mapping as example
  - First, create a white color Lambert material
  - Then, assign and load texture map to the material

```
var earthmaterial = new
THREE.MeshLambertMaterial( { color: 0xffffff });

earthmaterial.map  =
THREE.ImageUtils.loadTexture('earthmap1k.jpg');
```

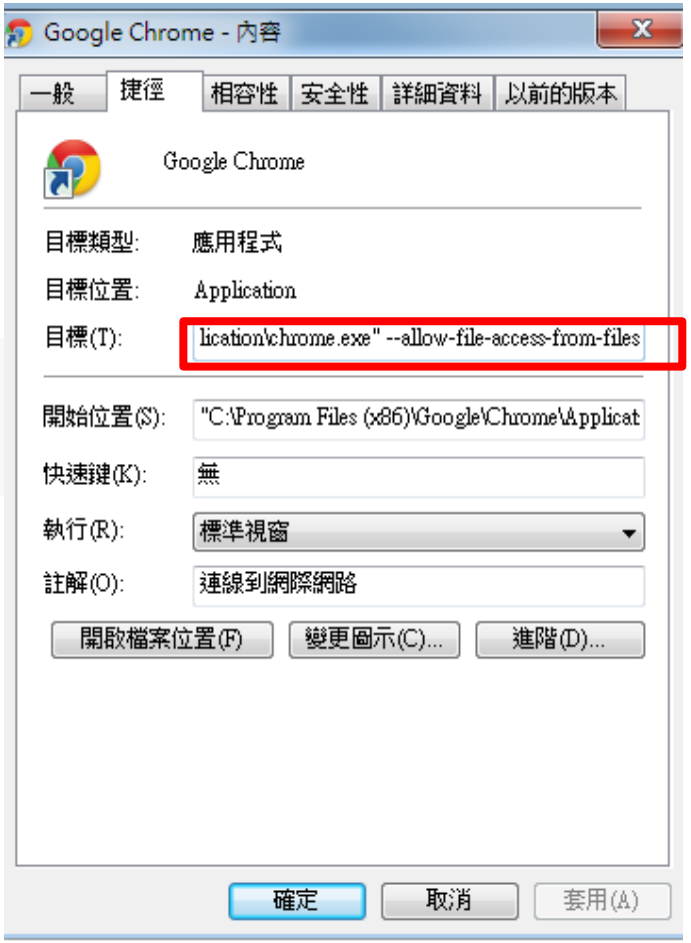
  - By default, UVMapping is used

Loading of Textures

- Notice that we cannot load texture locally due to the browser's restriction
- Solutions
  - Configure the browser to allow it
  - Run a local server
- Details refer to article "How to run things locally"
  - https://github.com/mrdoob/three.js/wiki/How-to-run-things-locally

- Assume you are using Chrome on Windows
  - Add the command line argument "--allow-file-access-from-files" when starting Chrome

```
chrome --allow-file-access-from-files
```



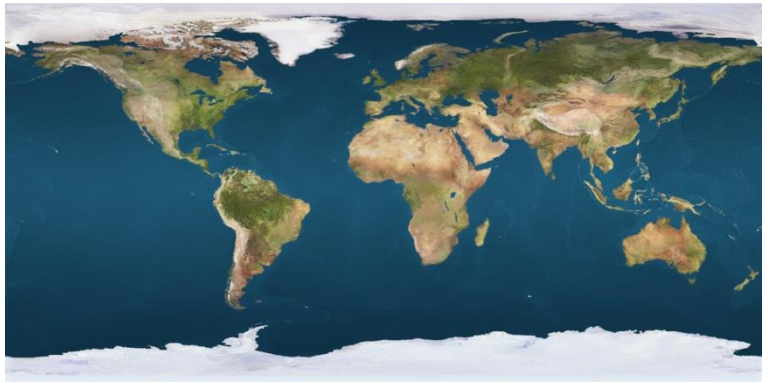
  - You can set it by right-click the Chrome icon, and select "Settings"
  - Add the argument the Target field

Creating the Skybox

- After doing all these steps, we can load in the earth's texture map
- Finally apply it to the earth's mesh

```
earthMesh = new THREE.Mesh(new
THREE.SphereGeometry(1, 20, 20), earthmaterial );
```

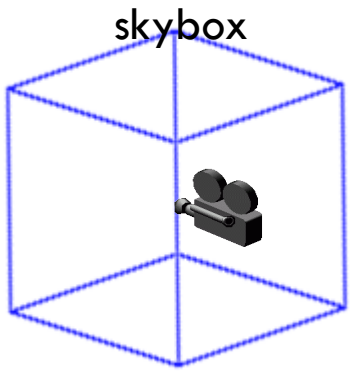
Texture mapped result



earthmap1k.jpg

Creating the Skybox

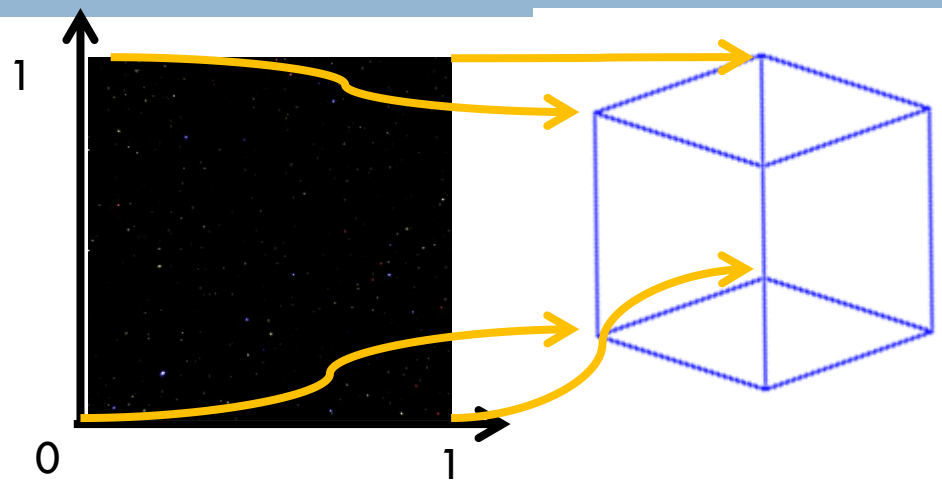
- A skybox is a cube which surrounds the viewing camera
- Usually, we will texture map the interior of the box
  - In our case, we texture map with a star field texture as shown below:



star.jpg

Creating the Skybox

- In plain OpenGL, we create 6 planes and set the texture coordinate for the4 corners of a plane




```
gl.glTexCoord2f(0, 0); gl.glVertex3f( 0.5f, -0.5f, -0.5f );
gl.glTexCoord2f(1, 0); gl.glVertex3f( -0.5f, -0.5f, -0.5f );
gl.glTexCoord2f(1, 1); gl.glVertex3f( -0.5f, 0.5f, -0.5f );
gl.glTexCoord2f(0, 1); gl.glVertex3f( 0.5f, 0.5f, -0.5f );
```

Creating the Skybox in ThreeJS

- Similar but simpler, in ThreeJS, we create a cube geometry that is large enough

```
var skyGeometry = new THREE.CubeGeometry( 40, 40, 40 );
```

- Create the material with texture

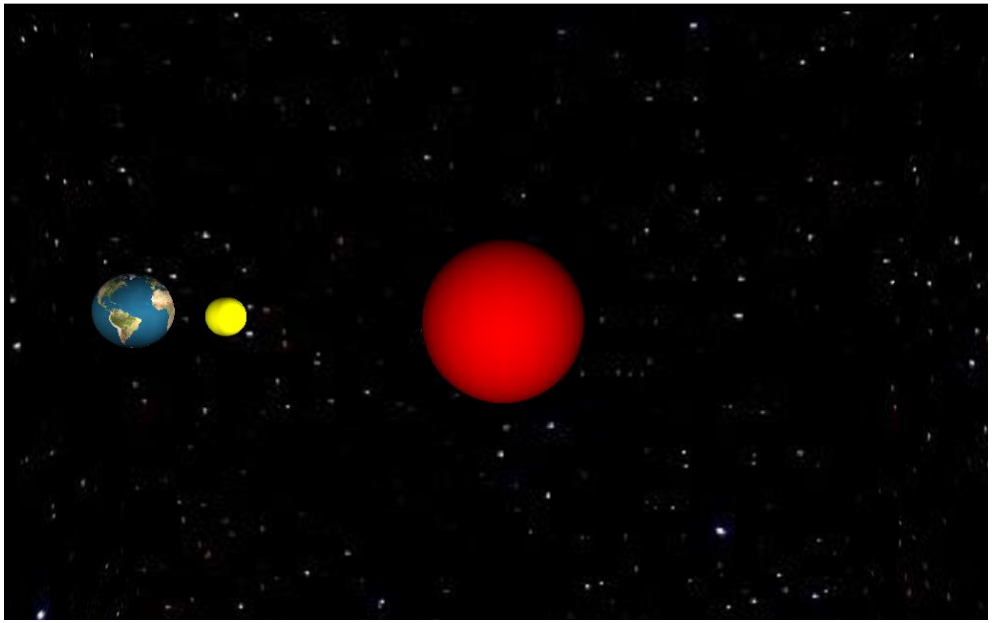
```
var skyMaterial = new THREE.MeshBasicMaterial({
map: THREE.ImageUtils.loadTexture( "star.jpg" ),
side: THREE.BackSide
});
```

Render only back side

Creating the Skybox in ThreeJS

- Finally, create the mesh of skybox, and add it to the scene

```
var skyBox = new THREE.Mesh( skyGeometry, skyMaterial );
scene.add( skyBox );
```



Other Texture Functionalities in ThreeJS

- .warpS and .warpT
- The arrangement of texture coordinate when the out of range [0,1]
- Possible values
  - THREE.ClampToEdgeWrapping (default)
    - i.e. non-repeating
  - THREE.RepeatWrapping, and
  - THREE.MirroredRepeatWrapping
    - i.e. wrap around and repeating

Other Texture Functionalities in ThreeJS

- MIPMapping is a built-in feature in OpenGL and ThreeJS
- **.minFilter**
  - How the texture is sampled when a texel covers less than one pixel.
  - Possible Values
    - THREE.LinearMipMapLinearFilter (default),
      - uses mipmapping and trilinear filter on mipmap application and creation
    - THREE.NearestFilter,
    - THREE.NearestMipMapNearestFilter,
    - THREE.NearestMipMapLinearFilter,
    - THREE.LinearFilter, and
    - THREE.LinearMipMapNearestFilter.
      - Pick nearest texel or linearly interopolate on nearest four texels

Summary

- Based on the Planet Sample we studied important command and concept in ThreeJS for lighting and texturing
  - Setup of different lighting and parameters
  - Setting of material and parameters
  - Shade mode
  - Texture loading and mapping

- Other nice examples can refer to:

  http://threejs.org/examples/

  http://stemkoski.github.io/Three.js/