Texture Lecture(s)
Part C
Texture in THREE

The steps to texture mapping

1. Define UVs for each triangle
2. Specify image to be used for lookup
3. Specify how to use the color information

and

4. make sure all the parameters are set right

(Basic) Texture Mapping in THREE

- Create objects with UVs
- Load Textures
- Attach as colors to objects

Textures in THREE

- Create objects with UVs
- Load Textures
- Attach as colors to objects

Primitives – THREE

- have predefined UVs

~~**Geometry**~~

- ~~need to define for each face (complex)~~

Buffer Geometry

- it's an attribute UV

Setting Texture Coordinates for a Buffer Geometry

- Create a `BufferAttribute`
- named "uv"
- 2 numbers per vertex

```
let pts = [ 1,2,  3,4,  /* ... */];
let mem = new Float32Array(pts);

let buf = new T.BufferAttribute(mem,2);
geom.setAttribute("uv",buf);
```

Textures in THREE

- Create objects with UVs
- **Load Textures**
- Attach as colors to objects

Loading images may take time

Asynchronous loading

THREE takes care of it!

- texture is blank and fills in later

```
let tl=new T.TextureLoader().load("./THREE/UV_Grid_Sm.jpg");
```

- re-use these if possible (share between objects)
- different kinds of pre-processing happens (it's not just an image)

Textures in THREE

- Create objects with UVs
- Load Textures
- **Attach as colors to objects**

Most Materials take **maps**

Color map is called `map`

```
let t1=new T.TextureLoader().load("./THREE/UV_Grid_Sm.jpg");
let material = new T.MeshStandardMaterial({map:t1,roughness:0.75});
```

A **lot** is happening behind the scenes.

An Entire Example

```
// the square showing the texture
const sqGeom = new T.BufferGeometry();
const sqXYZ = new Float32Array([0,0,0, 1,0,0, 0,1,0, 1,1,0]);
const sqUV  = new Float32Array([0,0,   1,0,   0,1,   1,1  ]);
sqGeom.setAttribute("position", new T.BufferAttribute(sqXYZ,3) );
sqGeom.setAttribute("uv", new T.BufferAttribute(sqUV,2) );
sqGeom.setIndex([0,1,2,3,2,1]);
sqGeom.computeVertexNormals();
const texture = new T.TextureLoader().load("../textures/UV_Grid_Sm.jpg");
const texMat = new T.MeshBasicMaterial({color:"white",map:texture});
const square = new T.Mesh(sqGeom,this.texMat);
```

①
②
— position
↳ position
↳ uv

Some caveats for Textures in THREE

1. Lighting and material affects color too!
    - `MeshBasicMaterial` if you don't want lighting
2. Y is flipped by default

But, in Summary...

THREE makes texture easy! (even for fancy textures)