3D OBJECT LOADING AND MOUSE INTERACTIONS IN THREE JS

By Raymond Pang

ThreeJS Examples Online

- There are lots of resources on using ThreeJS
- Two better choices are:
  - http://threejs.org/examples/
  - http://stemkoski.github.io/Three.js/
- You can learn many advanced features in ThreeJS from the above site
- In the following of this tutorial, we will go through two major features:
  - 3D Object Loading
  - Mouse Interaction

Loading of OBJ files

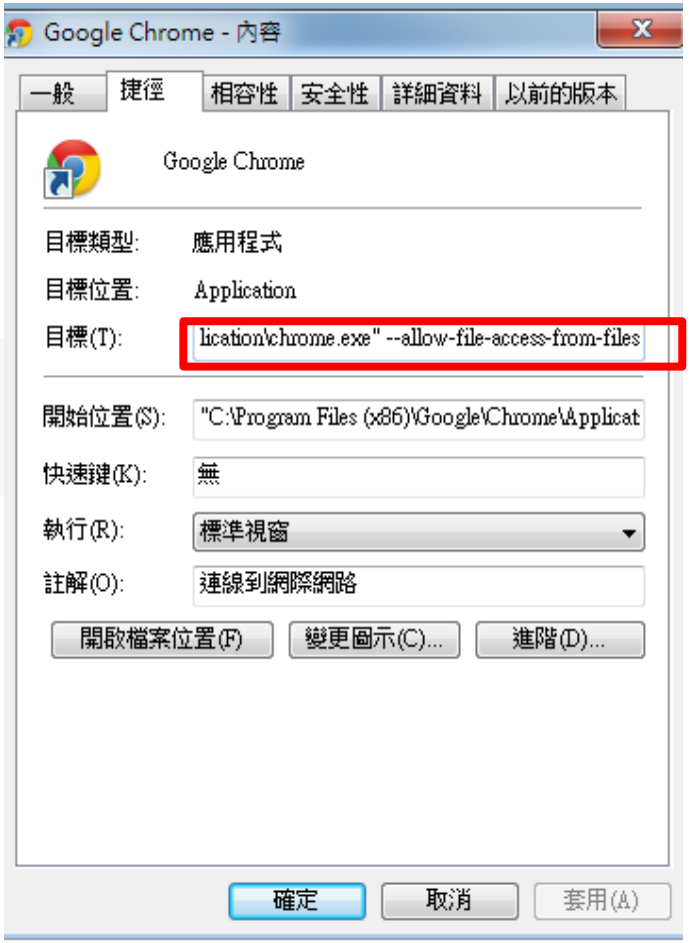
- As introduced in the lecture, 3D models are stored as vertices and triangles in files
- There are various formats for storing 3D models
  - OBJ, PLY, JSON, STL, COLLADA, and etc.
  - All these well-known formats are supported by ThreeJS
- Here, we use OBJ as our standard format, and see how to load 3D models to ThreeJS
- First, we need to enable access to files from local storage (as in last tutorial)

Local File Access

- 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



A Simplified OBJLoader Example

- We are based on OBJLoader example on THREEJS website
- But, we skip texture loading and mapping
- Use the "OBJLoader" Object
- Create it with following code:

```
var loader = new THREE.OBJLoader();
```

webgl_loader_obj_simplified.html

A Simplified OBJLoader Example

- Use the "load(…)" method of OBJLoader
- Two parameters: 1. file to load, 2. callback function
- After OBJ file is loaded, add it to the scene

```
var loader = new THREE.OBJLoader();
loader.load( 'male02.obj',
function ( object ) {
    object.position.y = - 80;
    scene.add( object );
}
);
```

OBJ file to load

Call back function
Invoked when loading of
OBJ file is completed

Add model to the scene

THREEJS OBJLoader Example

- The original example also has
  - Use of LoadingManager
  - Texture loading & mapping

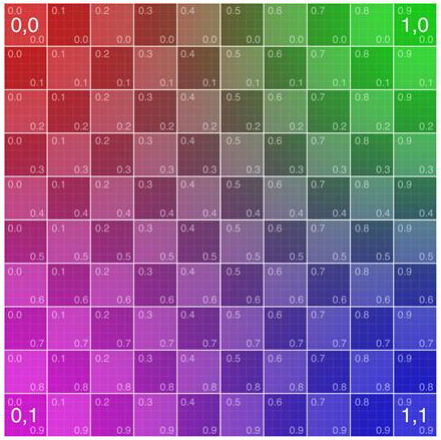
```
var manager = new
THREE.LoadingManager();

manager.onProgress = function
( item, loaded, total ) {
   console.log( item, loaded,
total );
};
```

```
var texture = new THREE.Texture();
var loader = new
THREE.ImageLoader( manager );
loader.load( 'UV_Grid_Sm.jpg', function
( image ) {
    texture.image = image;
    texture.needsUpdate = true;
} );
```

THREEJS OBJLoader Example

```
loader.load( 'male02.obj', function ( object ) {
    object.traverse( function ( child ) {
        if ( child instanceof THREE.Mesh ) {
            child.material.map = texture; }        } );
 ..... } );
```

Apply texture when model is loaded



UV_Grid_Sm.jpg

webgl_loader_obj.html

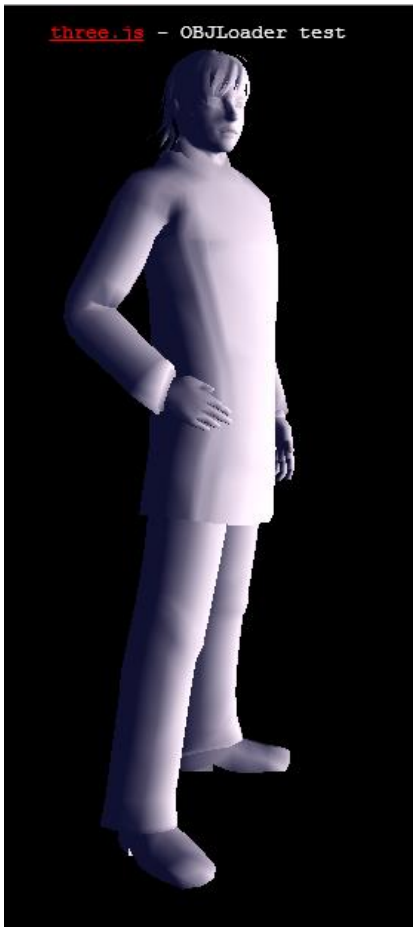
Loading together with MTL file

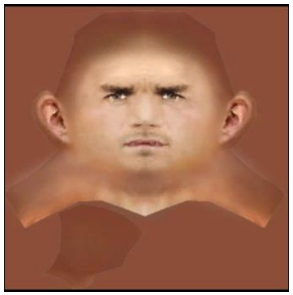
- OBJ files are commonly accompanied with a material file (MTL)
- In ThreeJS, we use the OBJMTLLoader class
  - 2nd parameter is the material file

```
var loader = new THREE.OBJMTLLoader();
loader.load( 'male02.obj', 'male02_dds.mtl',
function ( object ) {
        object.position.y = - 80;
        scene.add( object );
} );
```

OBJMTLLoader

- Inside the MTL file, it defined some materials with textures





orig_02_-_Defaul1noCulling.dds

01_-_Default1noCulling.dds

male-02-1noCulling.dds

Mouse Interactions

Dragging 3D Object with Mouse

- We are based on Draggable Cubes example on THREEJS website
- Steps for Dragging

1. Detect mouse click
2. Check if mouse clicked on a cube
3. Cube move with mouse
4. Until mouse release

Dragging 3D Object with Mouse

- Mechanism of capturing mouse events is similar to what is in Javascript
- Use “addEventListener”, and add callback functions

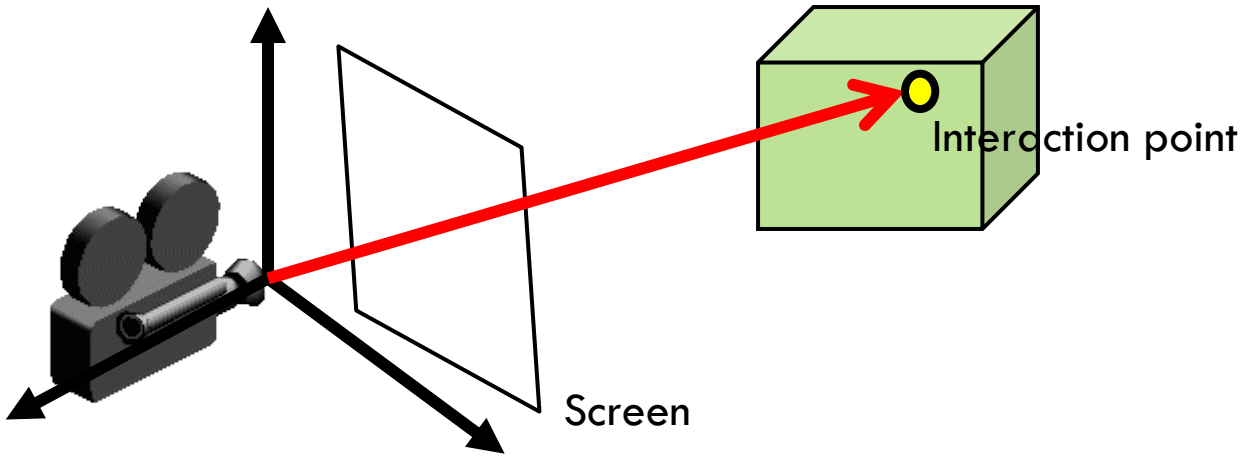
```
renderer.domElement.addEventListener( 'mousemove',
onDocumentMouseMove, false );
renderer.domElement.addEventListener( 'mousedown',
onDocumentMouseDown, false );
renderer.domElement.addEventListener( 'mouseup',
onDocumentMouseUp, false );
```

Callback functions for different events

Cube Selection with Raycaster

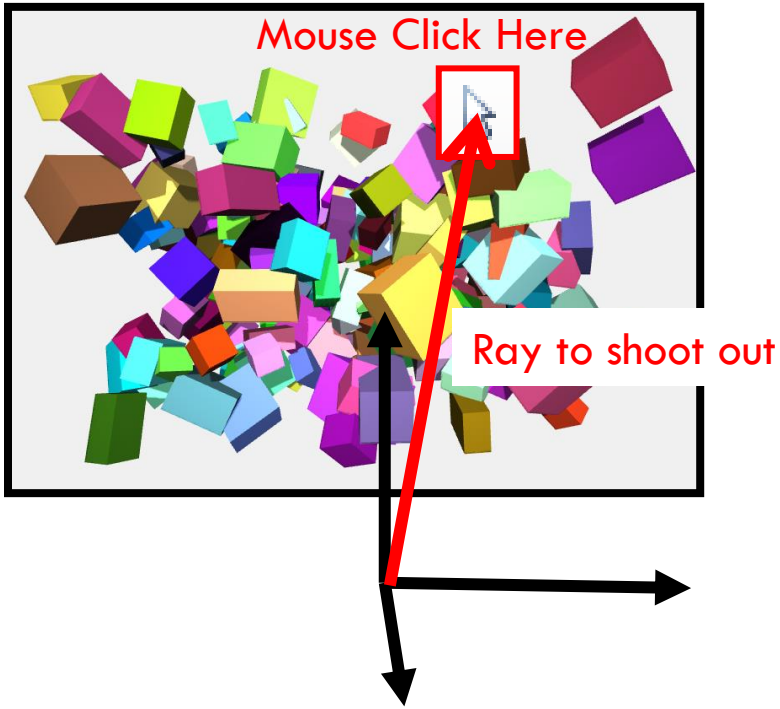
- Casting ray from the screen
  - A straight line from camera to see if intersect any geometries
- We can use "Raycaster" class in THREEJS
  - Given camera's origin and ray direction

```
THREE.Raycaster( origin, direction, near, far )
```



Cube Selection with Raycaster

- Origin : fixed at the current camera's center
- Direction: compose a ray from original to where the mouse click happens



Cube Selection with Raycaster

- Code to compute where mouse click happens in world space

```
mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;

var vector = new THREE.Vector3( mouse.x, mouse.y, 0.5 );

projector.unprojectVector( vector, camera );
```

From Viewpoint to Screen space

Screen at z=0.5

Inverse transform from screen space to world space

- Code to compute Ray for casting

```
vector.sub( camera.position ).normalize()
```

substraction

Form the ray from camera center to the mouse click point

Cube Selection with Raycaster

- Create an "Raycaster" object

onDocumentMouseDown

```
var raycaster = new THREE.Raycaster( camera.position,
vector.sub( camera.position ).normalize() );
```

- Invoke "IntersectObjects" method

Array of THREE.Mesh objects

```
var intersects = raycaster.intersectObjects( objects );
if ( intersects.length > 0 ) {
        controls.enabled = false;
        SELECTED = intersects[ 0 ].object;
        …..
        container.style.cursor = 'move';
}
```

Hits any object if intersects.length > 0

Disable trackball rotation (discuss later)

The first object hit

Change mouse cursor

Cube Movement

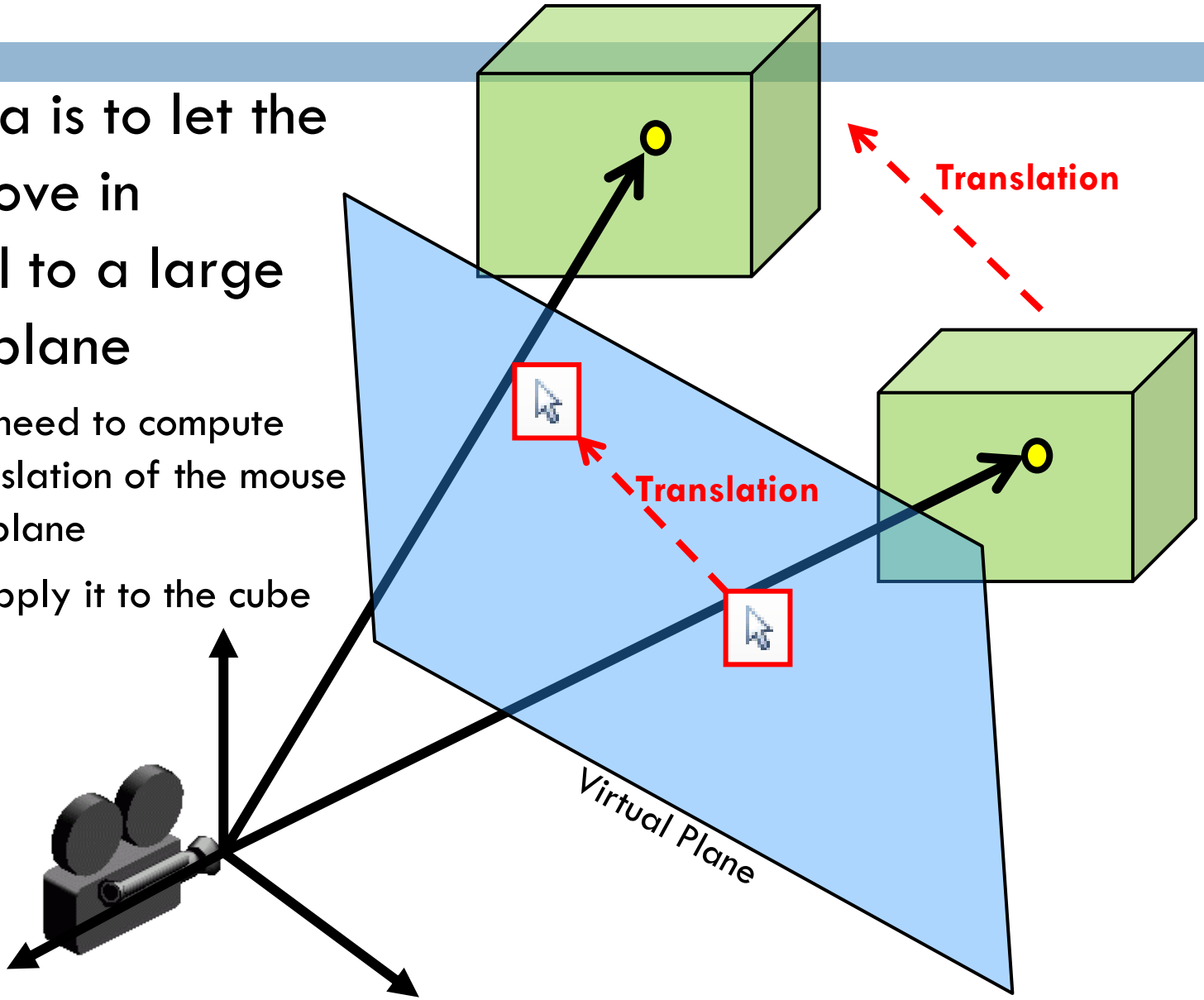
- In mousedown, variable "SELECTED" is assigned with the 3D object being clicked
- So, in mousemove callback, we see codes for handling movement of selected object
  - But how should we move the cube?

onDocumentMouseMove

```
....
if ( SELECTED ) {
   var intersects = raycaster.intersectObject( plane );
   SELECTED.position.copy( intersects[ 0 ].point.sub( offset ) );
    return;        }
....
```

Cube Movement

- The idea is to let the cube move in parallel to a large virtual plane
  - So, we need to compute the translation of the mouse on this plane
  - Then, apply it to the cube



Cube Movement

- The first line of code, we try to intersect the ray with "plane", which is defined in the program as

```
plane = new THREE.Mesh( new THREE.PlaneGeometry( 2000, 2000, 8, 8 )
```

- It is a very large invisible plane at the center
- We use it to compute the point the cube should move to

onDocumentMouseMove

```
....
var intersects = raycaster.intersectObject( plane );
....
```

Cube Movement

- The second code is simply assigning the cube to new position

onDocumentMouseMove

```
....
SELECTED.position.copy( intersects[ 0 ].point.sub( offset ) );
....
```

- Notice that there is a subtraction between the plane-ray interaction to the "offset", which is computed when mousedown:

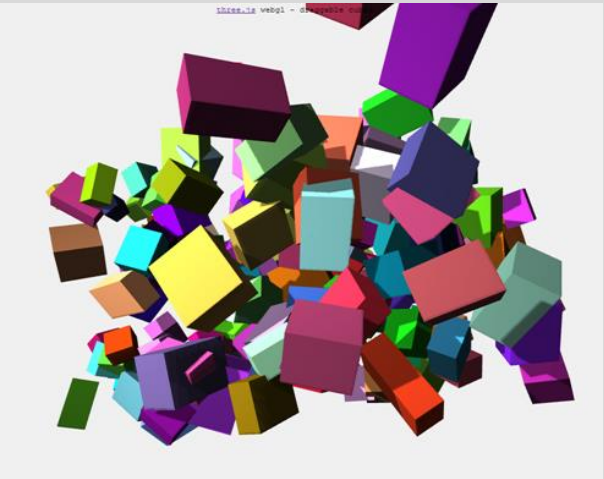
```
...
offset.copy( intersects[ 0 ].point ).sub( plane.position );
```

- This is pretty similar to the DnD example in tutorial 1 (page 43)

Trackball Controls

- The same example can rotate view angle to the scene by mouse drag
- Use "TrackballControls"

```
controls = new THREE.TrackballControls( camera );
controls.rotateSpeed = 1.0;
controls.zoomSpeed = 1.2;
controls.panSpeed = 0.8;
controls.noZoom = false;
controls.noPan = false;
controls.staticMoving = true;
controls.dynamicDampingFactor = 0.3;
```



Summary

- How to load OBJ models to THREEJS environment
- How to use mouse to select objects and move them
- Predefined control objects like Trackball control, easy the task of moving view points
- Try to explore more from:
  - http://threejs.org/examples/
  - http://stemkoski.github.io/Three.js/
- You can refer to some classes and methods
  - http://threejs.org/docs