Web 3D using A-Frame (Mozilla)

Intended Learning Outcome of Lesson

By the end of the workshop, participants are expected to be able to:

- Create 3D scene, with 3D primitives, using Mozilla A-Frame framework
- Combine A-Frame and THREE.js in creating components

Text eBook

- ***AR and VR using the WebXR API: learn to create immersive content with WebGL, Three.js, and A-Frame***
- by Baruah, Rakesh
- 2021, Apress
- (Chapter 8,9 and 10 for A-Frame)

Introduction

- Developed by **Mozilla**, the team behind Firefox, A-Frame is a framework for creating Three.js applications.
- https://aframe.io/

- A-Frame, as a framework for Three.js, provides a set of rules and conveniences that place the writing of Three.js applications more closely in line with HTML documents.

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>A-Frame app by your name</title>

<head>

  <script src="https://aframe.io/releases/1.7.0/aframe.min.js"></script>


</head>
```

a-scene

- A-Frame is written in **declarative HTML syntax**, like a traditional, 2D webpage (like SwiftUI too!).
- To add a scene object to an A-Frame application, add a scene tag to the body of an HTML document.

```
<body>
  <a-scene>


  </a-scene>
</body>
```

- Remember, A-Frame is an abstraction of Three.js, which is, in turn, an abstraction of WebGL.
- As an abstraction of an abstraction, A-Frame hides a lot of details required to create an XR-enabled Web application
- Contained within the scene entity in A-Frame is a collection of components,
  - **Three.js camera**;
  - a **THREE.Scene** object; and
  - a **THREE.WebGLRenderer** in a component called, conveniently, “renderer”.
- With one tag in A-Frame we’ve instantiated no less than four Three.js objects.

ECS

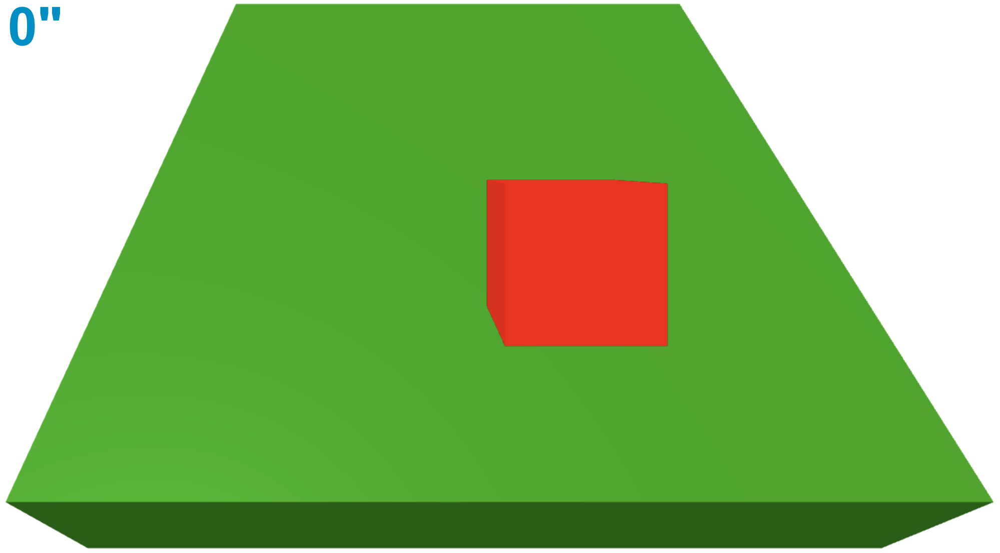
- A software design pattern, particularly popular for game design, is called the **Entity Component System** (ECS)
- It is the premise on which A-Frame has been built.
- Entity-Component-System – A-Frame

Entities

- https://aframe.io/docs/1.7.0/core/entity.html

```
<a-entity geometry="primitive: box" material="color: red"
position="1 1 -4"></a-entity>
```

```
<a-entity geometry="primitive: box" rotation="-45 0 0"
material="color: green" position="0.768 1.04 -10"
scale="10 10 2" ></a-entity>
```



Primitives

- A-Frame wouldn't be all that convenient if every primitive shape we hoped to include in an XR scene we had to create from generic entities.
- Fortunately, the A-Frame library provides an assortment of commonly used primitive objects as premade, ready-to-use entities.

```
<a-box color = "tomato" height = "0.5" width = "0.1" position = "1 1 -
2"> </a-box>
```

The Component

- Components, in A-Frame, are objects that define the character of an entity.
- Components built into the A-Frame library include:
  - an animation component,
  - a background component,
  - a camera component,
  - a 3D-model component,
  - a touch-control component,
  - …

Asset

- In addition to the more obvious attributes a primitive shape may have, like its colour and dimensions, A-Frame primitives offer convenient access to more complex properties like image textures and materials.
- To add an image file to our scene as a material, we can make use of A-Frame's **Asset Management System**.
- A **system** provides global scope, services, and management to classes of components. The Asset Manager is one type of system provided by A-Frame

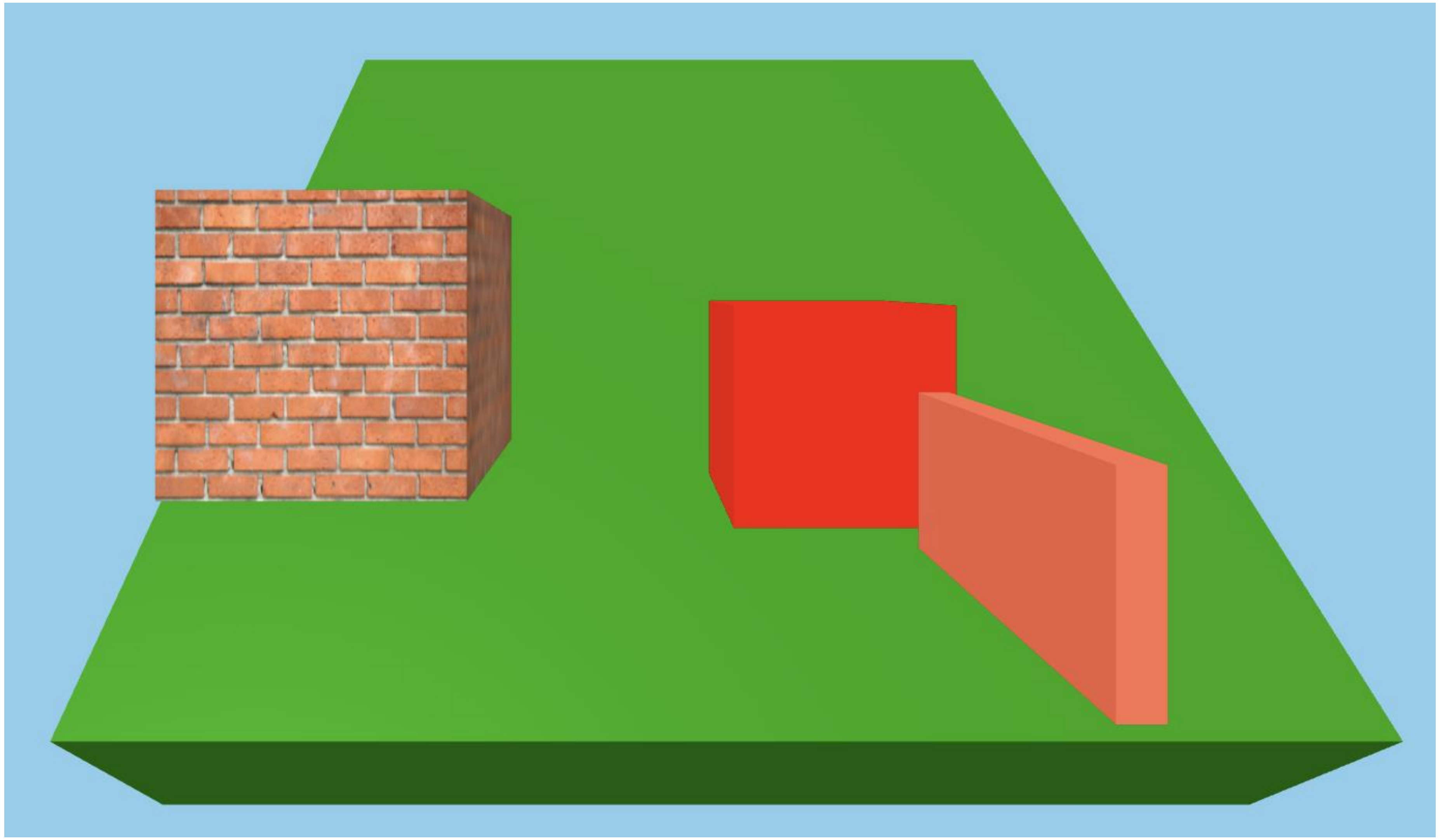
Asset & Material Component

- <a-scene>

```
<a-assets>
<img id="brick" src="img/brick_mat.jpg" />
</a-assets>
```

- ...

```
<a-box position="-0.5 1.5 -1.5" material="src: #brick; roughness: 100;"
depth="0.5" height="0.5" width="0.5"></a-box>
 <a-sky color="#87CEEB"></a-sky>
```



Interaction using THREE.js

- Three.js provides properties through its texture and material objects that allow us to fine-tune the settings applied to assets.
- Extending A-Frame through Three.js will set us on the road toward creating A-Frame components of our own.
- Add a box (#cubrick) in the scene.

```
<a-box id="cubrick" position="-0.3 3.0 -3.8" depth="0.5" height="0.5"
width="1.9"></a-box>
```

- Beneath closing </a-scene> tag, add the following JavaScript

```
<script>
        //5 seconds delay
        setTimeout((e) => {
            const texture = new window.THREE.TextureLoader().load('img/bricks01.jpg');

            const material_tex = new THREE.MeshBasicMaterial({ map: texture });
            const box = document.querySelector('#cubrick');
            let mesh = box.getObject3D('mesh');
            console.log(mesh)
            mesh.material = material_tex;
        }, 5000)
    </script>
```