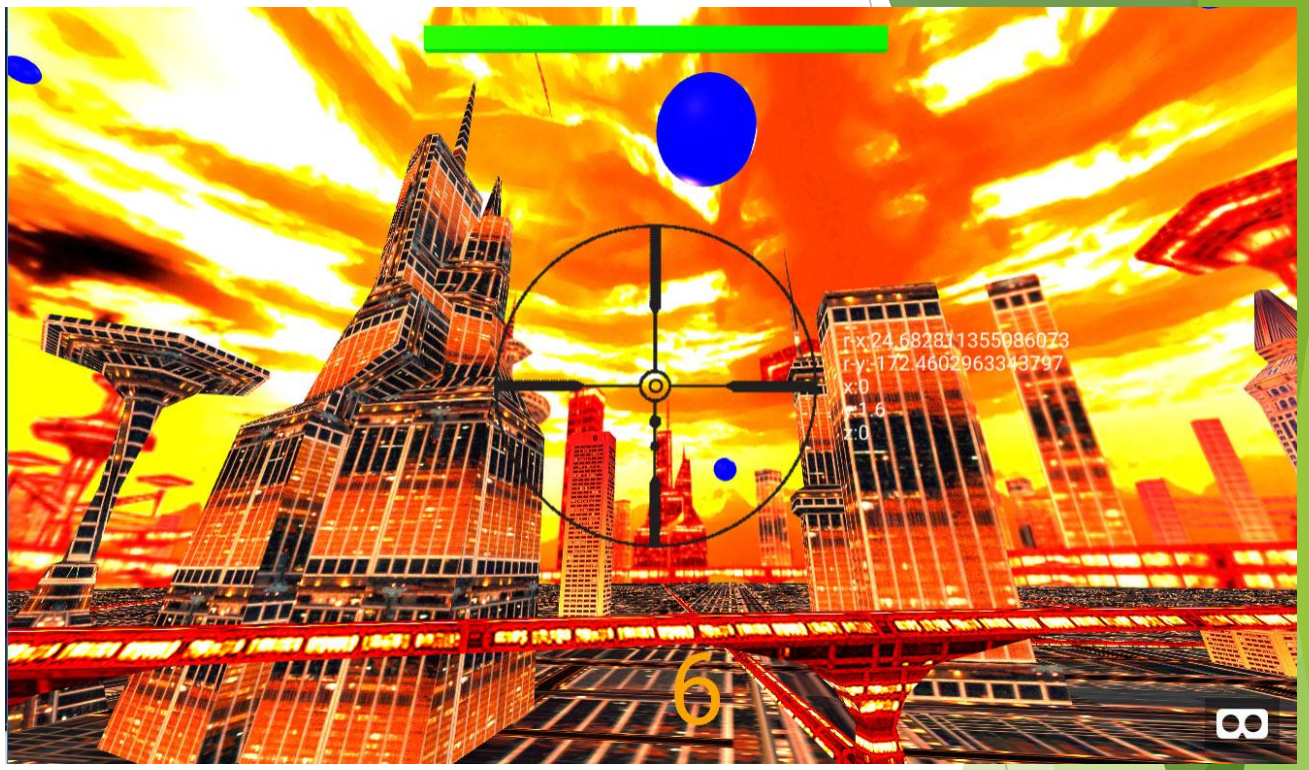
WebVR with Aframe

Interactive 3D websites with HTML and TypeScript

Michael Kappel

Please Download



http://michaelkappel.com/webvr/

http://michaelkappel.com/webvr/WebVR.pptx

http://michaelkappel.com/webvr/webvr.zip

What is A-Frame?

- A-Frame is an open-source web framework for building virtual reality (VR) experiences.
  - HTML Support
  - Simplified API
  - https://aframe.io/
  - https://en.wikipedia.org/wiki/A-Frame_(virtual_reality_framework)
- A-Frame is built on top of three.js
  - three.js is a JavaScript Library
  - https://threejs.org/
  - https://en.wikipedia.org/wiki/Three.js
- three.js is built on top of WebGL
  - WebGL is a cross-platform, royalty-free web standard for a low-level 3D graphics API based on OpenGL ES
  - https://www.khronos.org/webgl/
  - https://en.wikipedia.org/wiki/WebGL

index.html – simple example

```
<!DOCTYPE html>
<html>
<head>
    <title>My A-Frame Game</title>
    <script src="https://aframe.io/releases/0.6.0/aframe.min.js"></script>
</head>
<body>
    <a-scene>
        <a-sky src="https://c1.staticflickr.com/5/4210/35046006562_6f3dcffd85_o.jpg"></a-sky>
     </a-scene>
</body>
</html>
```

*https://www.flickr.com/groups/360degrees

Animations, Audio, and Assets

```
<a-scene>
        <a-assets timeout="1000">
            <audio id="loadsoundasset" src="loadsound.mp3"></audio>
            <img id="backgroundasset" src="sky.jpg" />
            <img id="phoneasset" src="phone.jpg" />
        </a-assets>
        <a-sky src="#backgroundasset"></a-sky>

        <a-box id="phone" material="src:#phoneasset“ sound="src: #loadsoundasset; on: click">
            <a-animation attribute="rotation"
                        dur="10000"
                        fill="forwards"
                        to="0 360 0"
                        repeat="indefinite"></a-animation>
        </a-box>
        <a-camera>
            <a-cursor ></a-cursor>
        </a-camera>
    </a-scene>
```

*https://freesound.org/

.gltf - royalty-free compressed models

```
<a-scene>
    <a-assets timeout="1000">
        <a-asset-item id="cityModel" src="https://cdn.aframe.io/test-models/models/virtualcity/VC.gltf"></a-asset-item>
    </a-assets>
    <a-entity gltf-model="#cityModel" play-all-model-animations=""></a-entity>
    <a-camera>
        <a-cursor></a-cursor>
    </a-camera>
</a-scene>
```

**USE 1.0 "glTF-MaterialsCommon"*
https://github.com/KhronosGroup/glTF-Sample-Models
http://cesiumjs.org/convertmodel.html

.glft in the Web.Config

```
<system.webServer>
    <staticContent>
        <remove fileExtension=".bin" />
        <mimeMap fileExtension=".bin" mimeType="application/octet-stream" />
        <mimeMap fileExtension=".gltf" mimeType="application/octet-stream" />
        <mimeMap fileExtension=".glb" mimeType="application/octet-stream" />
    </staticContent>
<system.webServer>
```

onclick events

```
<!DOCTYPE html>
<html>
<head>
    <title>My A-Frame Game</title>
    <script src="aframe.min.js"></script>
    <script type="text/javascript">
        function kill(elem) {
            var scene = document.querySelector('a-scene');
            scene.removeChild(elem);
        }
    </script>
</head>
<body>
        <a-scene>
            <a-assets timeout="1000">
                <a-asset-item id="duckasset" src="/duck/Duck.gltf"></a-asset-item>
            </a-assets>
            <a-entity  onclick="kill(this)">
                <a-entity id="duck" gltf-model="#duckasset" play-all-model-animations width="1" position="0 2 -20">
                </a-entity>
            </a-entity>
            <a-box id="box" onclick="kill(this)" color="blue" position="0 1 -2"></a-box>
            <a-camera>
                <a-cursor></a-cursor>
            </a-camera>
        </a-scene>
</body>
</html>
```

Get the JavaScript out of the page!
* Log all entity event example

```
<a-scene>
        <a-entity log-detail>
        </a-entity>
</a-scene>
```

aframe.patch.js

```
var VrWrapper = (function () {
    me = this;
    function RegisterComponentWithAFrame(title, obj)
    {
        var aFrameObj = {};
        if (obj.dependencies){aFrameObj.dependencies = obj.dependencies;}

        if (obj.schema){ aFrameObj.schema = obj.schema;}

        if (obj.init) { aFrameObj.init = function () { var elem = this; obj.init(elem);}}

        if (obj.update){ aFrameObj.update =  function () {var elem = this;obj.update(elem);}}

        if (obj.tick){ aFrameObj.tick =  function () {var elem = this; obj.tick(elem);}}

        if (obj.remove){ aFrameObj.remove =  function () {var elem = this;obj.remove(elem);}}

        if (obj.pause){ aFrameObj.pause =  function () {var elem = this;obj.pause(elem);}}

        if (obj.play){ aFrameObj.play =  function () {var elem = this;obj.play(elem);}}

        if (obj.updateSchema) { aFrameObj.updateSchema = function () {var elem = this; obj.updateSchema(elem);}}

        AFRAME.registerComponent(title, aFrameObj);
    }

    return {
        RegisterComponent: function (title, obj) {
            return RegisterComponentWithAFrame(title, obj);
        }
    }
})();
```

BaseComponents.ts

```
namespace VR {
    export interface ComponentDefinition {
        dependencies?: string[];
        el?: AFrame.Entity;
        id?: string;
        multiple?: boolean;
        schema?: AFrame.Schema;

        init?(arg: AFrame.Component): void;
        update?(oldData: any, arg: AFrame.Component): void;
        remove?(arg: AFrame.Component): void;
        tick?(time: number, timeDelta: number, arg: AFrame.Component): void;
        play?(arg: AFrame.Component): void;
        pause?(arg: AFrame.Component): void;
        updateSchema?(arg: AFrame.Component): void;
        remove?(arg: AFrame.Component): void;

        [key: string]: any;
    }

    export module Base {
        export abstract class Component {
            public ComponentConstructed: AFrame.ComponentConstructor;
            public abstract GetDefinition(): VR.ComponentDefinition;

            constructor(propertyName: string) {
                var me = this;
                me.ComponentConstructed = VrWrapper.RegisterComponent(propertyName, me.GetDefinition());
            }
        }
    }
}
```

LogDetailComponent.ts

```
/// <reference path="BaseComponents.ts" />
namespace VR.Components {
    export class LogDetailComponent extends VR.Base.Component {
        constructor() {
            super('log-detail');
            var me = this;
        }

        public GetDefinition(): VR.ComponentDefinition {
            var me = this;
            return <VR.ComponentDefinition>{
                schema: {
                    type: 'string',
                },
                init: (component: AFrame.Component) => {
                    me.ComponentInit(component);
                },
                update: (oldData: any, component: AFrame.Component) => {
                    me.ComponentUpdate(oldData, component);
                },
                ComponentTick: (time: number, timeDelta: number, component: AFrame.Component) => {
                    me.ComponentTick(time, timeDelta, component);
                },
                remove: (component: AFrame.Component) => {
                    me.ComponentRemove(component);
                },
                pause: (component: AFrame.Component) => {
                    me.ComponentPause(component);
                },
                play: (component: AFrame.Component) => {
                    me.ComponentPlay(component);
                }
            };
        }


        public LogProperties(title: string, arg: Object) {
            console.log('************ ' + title + ' ************');
            for (var prop in arg) {
                console.log(prop + '=' + arg[prop]);
            }
        }

    }
}
```

```
public ComponentInit(component: AFrame.Component): void {
        var me = this;
        var el = component.el;
        var schema = component.schema;
        var data = component.data;
        me.LogProperties('component', component);
        me.LogProperties('el', el);
        me.LogProperties('schema', schema);
        me.LogProperties('data', data);
    }

    public ComponentUpdate(oldData: any, component: AFrame.Component): void {
        var me = this;
        var el = component.el;
        var schema = component.schema;
        var data = component.data;
        me.LogProperties('component', component);
        me.LogProperties('el', el);
        me.LogProperties('schema', schema);
        me.LogProperties('data', data);
    }

    public ComponentTick(time: number, timeDelta: number, component: AFrame.Component): void {
        var me = this;
        var el = component.el;
        var schema = component.schema;
        var data = component.data;
        me.LogProperties('component', component);
        me.LogProperties('el', el);
        me.LogProperties('schema', schema);
        me.LogProperties('data', data);
    }

    public ComponentRemove(component: AFrame.Component): void {
        var me = this;
        var el = component.el;
        var schema = component.schema;
        var data = component.data;
        me.LogProperties('component', component);
        me.LogProperties('el', el);
        me.LogProperties('schema', schema);
        me.LogProperties('data', data);
    }

    public ComponentPause(component: AFrame.Component): void {
        var me = this;
        var el = component.el;
        var schema = component.schema;
        var data = component.data;
        me.LogProperties('component', component);
        me.LogProperties('el', el);
        me.LogProperties('schema', schema);
        me.LogProperties('data', data);
    }

    public ComponentPlay(component: AFrame.Component): void {
        var me = this;
        var el = component.el;
        var schema = component.schema;
        var data = component.data;
        me.LogProperties('component', component);
        me.LogProperties('el', el);
        me.LogProperties('schema', schema);
        me.LogProperties('data', data);
    }
```

Index.ts

```
namespace VR {
    export module Pages {
        export class Index {
            public LogDetail: VR.Components.LogDetailComponent;
            public constructor() {
                var me = this;

                me.LogDetail = new VR.Components.LogDetailComponent();
            }
            public Loaded() {
                alert('page loaded');
            }
        }
    }
}
```

Index.html

```
<head>
    <title>My A-Frame Game</title>
    <script type="application/javascript" src="aframe.min.js"></script>
    <script type="application/javascript" src="aframe.patch.js"></script>
    <script type="application/javascript" src="BaseComponents.js"></script>
    <script type="application/javascript" src="LogDetailComponent.js"></script>
    <script type="application/javascript" src="index.js"></script>
    <script type="application/javascript">
        var index = VR.Pages.Index()
    </script>
</head>
<body>
    <a-scene>
        <a-entity log-detail>
        </a-entity>
    </a-scene>

    <script type="application/javascript">
        index.loaded();
    </script>
</body>
```