Music-created Digital Environments with A-Frame

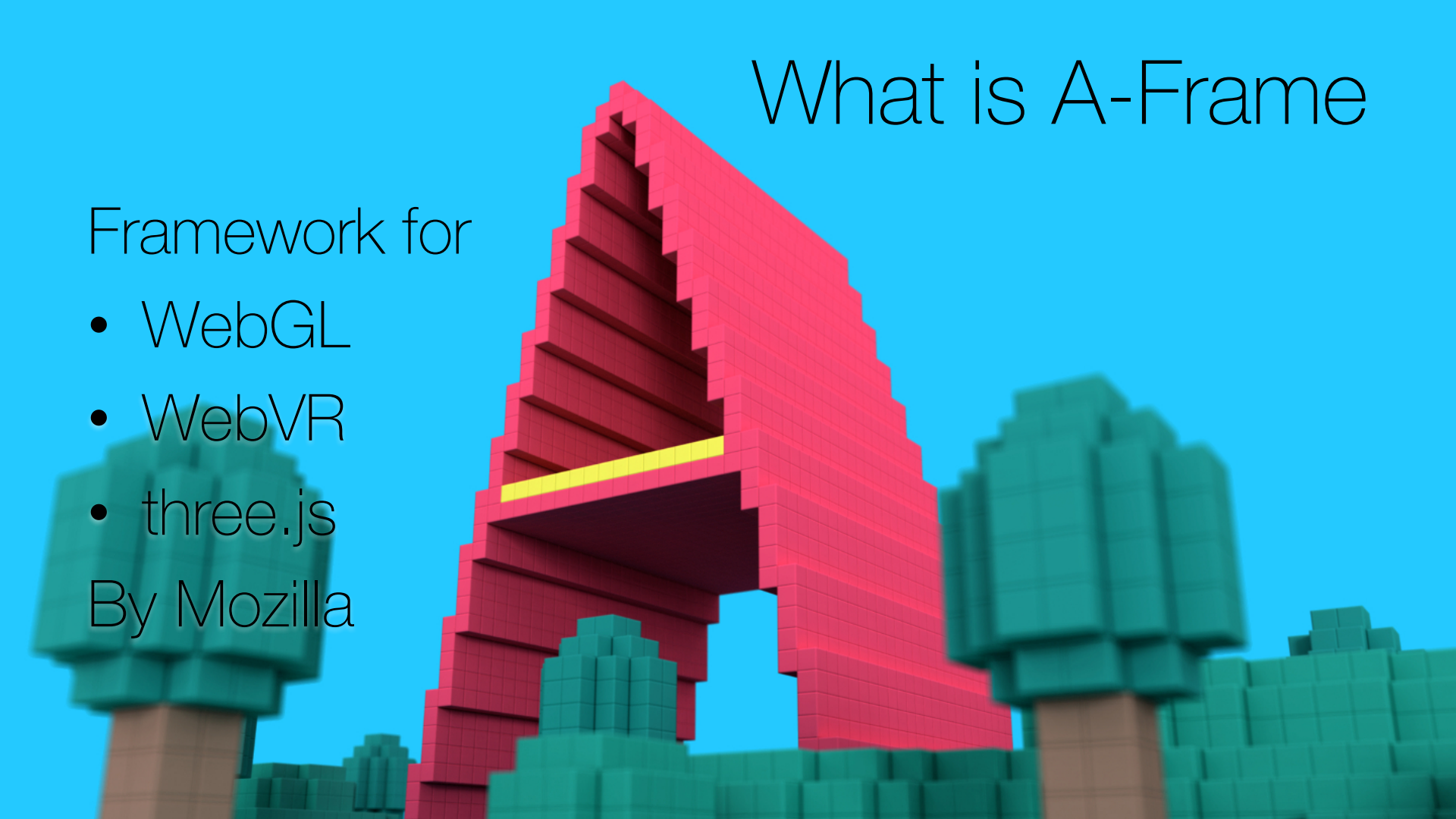
Gray Area Festival 2017

Yagiz Mungan
yagizmungan@gmail.com

Overview

- Introduction to A-Frame
- Fundamentals of A-Frame
- Creating *a city* from music with A-Frame

What is A-Frame

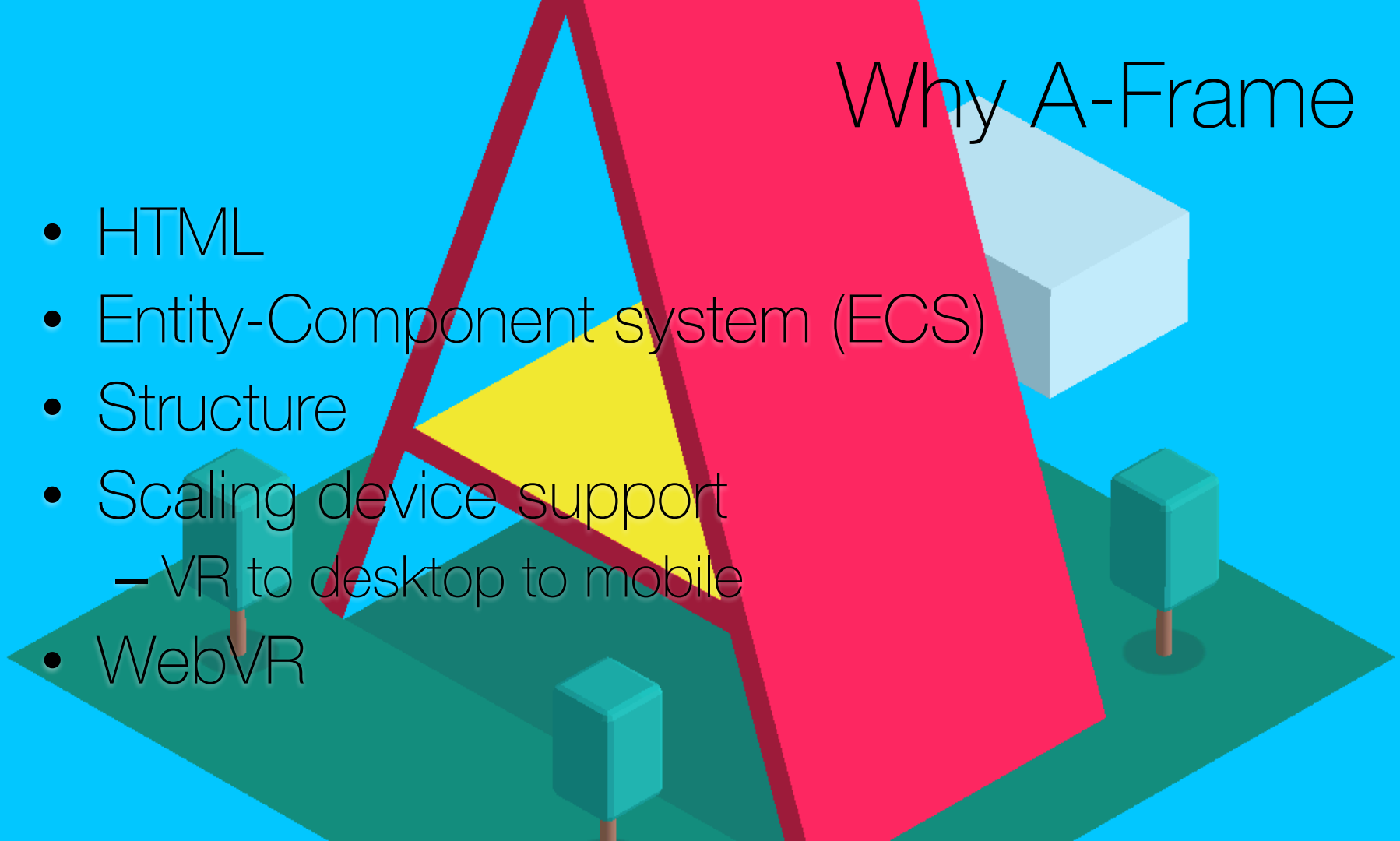


Framework for

- WebGL
- WebVR
- three.js

By Mozilla

Why A-Frame



- HTML
- Entity-Component system (ECS)
- Structure
- Scaling device support
  - VR to desktop to mobile
- WebVR

Why A-Frame | HTML

```
<body>
    <canvas></canvas>
    <script type="text/javascript">
        $(document).ready(function(){
            App.init();
        });
    </script>
</body>
```

Why A-Frame | HTML

```
<body>
    <a-scene>
        <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
        <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
        <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"></a-cylinder>
        <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#7BC8A4"></a-plane>
        <a-sky color="#ECECEC"></a-sky>
    </a-scene>
</body>
```

Why A-Frame | ECS

- Game development approach
- An *entity* is just a holder
- *Components* define behaviour
- *System* manages the larger picture logic

Why A-Frame | Structure

- Promotes reusability through
  - Components
  - HTML
  - Styling
- As opposed to single file apps

Why A-Frame | Device Support

- Out of the box works from
  - Desktop VR (with controls)
  - Mobile VR (with controls)
  - Desktop browsers
  - Mobile browsers

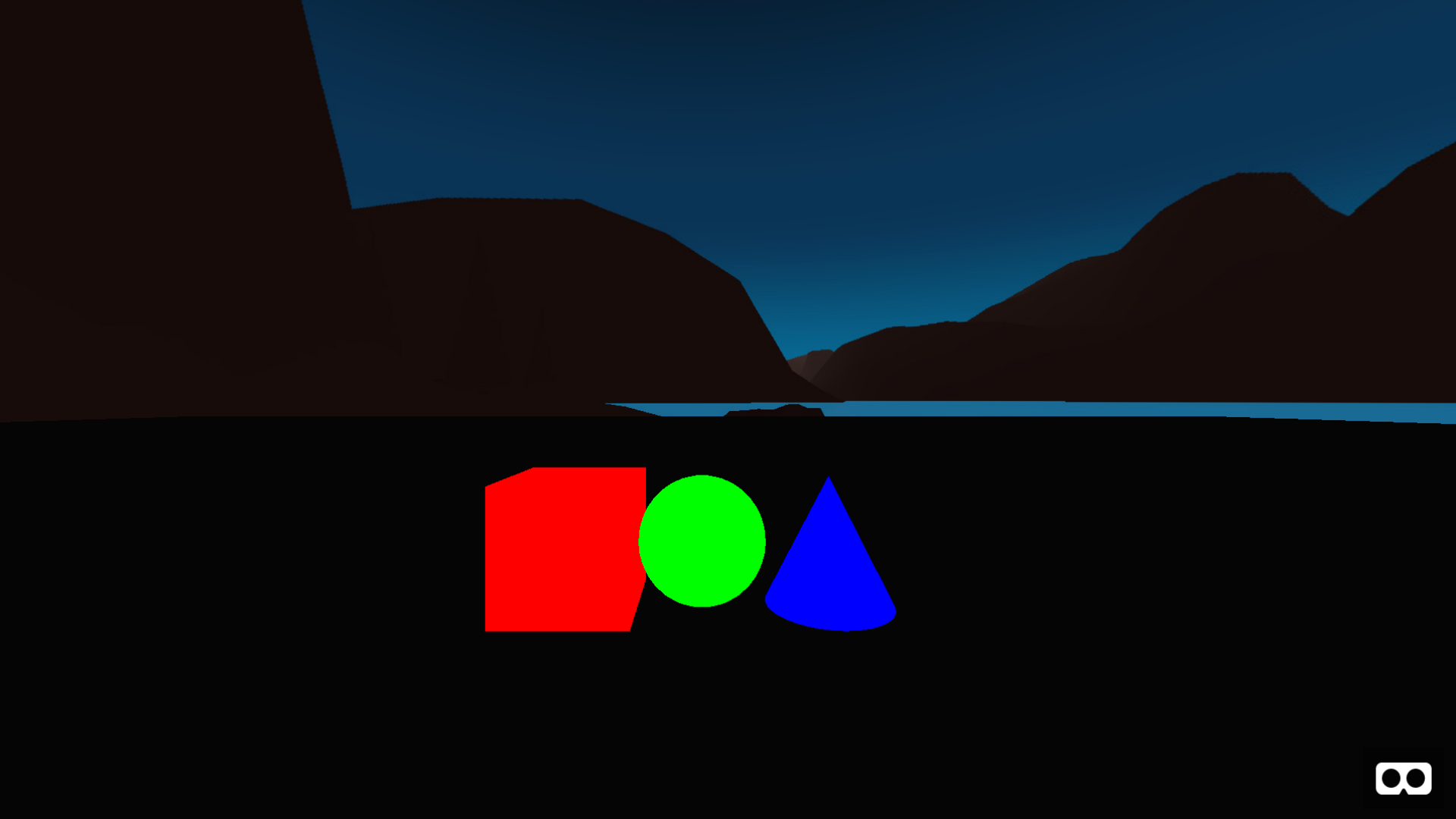
Why A-Frame | WebVR

- Work-in-progress Web standard
- Public in some Android devices
- In beta versions of Chrome and Firefox
- No app store
- No installation
- Standard and open!

Questions?

Hello World in A-Frame

- We will start building a Hello World page
  - Check point code is provided
  - Hands on approach is encouraged
  - Feel free to experiment



Hello World-1

- Let's go to our project folder
- Start *gulp watch* in terminal or command line
- Open *index.html* in your editor
- Our target is *helloworld1.html*
  - You can open it in another tab of the editor and browser

Hello World-1

- Remove "*Time to read the readme.md!*"
- Let's add some geometries
  - <a-box>
  - <a-sphere>
  - <a-cone>
    - They will appear where the camera is and white!
    - position="-1 0.5 -4" color="#ff0000"

Hello World-1

- Let's add sky, light and ground
  - <a-light>
    - type="ambient"
  - <a-sky>
    - color="#101033"
  - <a-entity>
    - a-entity is THE generic element

Hello World-1

- <a-entity>

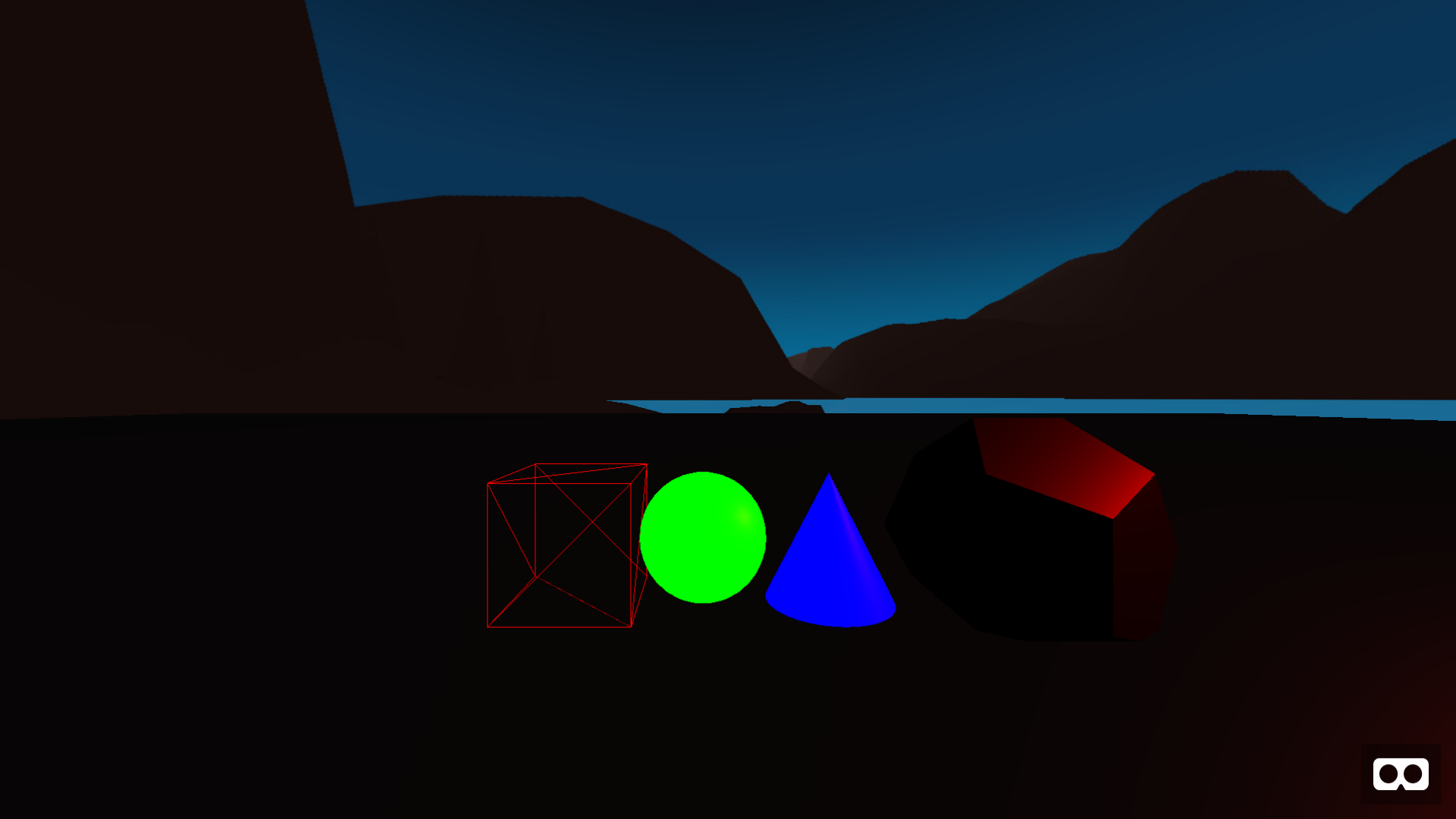
```
<!-- Ground -->
<a-entity id="ground"
  geometry="primitive: plane; height: 100; width: 100"
  material="side: double; color: rgb(5,5,5)"
  position="0 0 0"
  rotation="-90 0 0"
></a-entity>
```

Hello World-1

- Let's add some texture to the sky
  - The sky texture needs to be equirectangular
  - And use A-Frame's asset management
  - <a-asset>
    - <img src="./assets/images/sechelt-sky.jpg" id="skyTexture">
  - Link to sky using id
    - <sky src="#skyTexture">

Hello World-1

- Questions?
- Checkpoint code in *helloworld1.html*



Hello World-2

- Our target is *helloworld2.html*
  - You can open it in another tab of the editor and browser

Hello World-2

- A-Frame has its own inspector
  - Not an editor but an inspector
  - Designed for 3D web but with the similar capabilities of 2D inspector
  - Hit *ctrl + alt + i*
    - Let's investigate the *material* options for the cube

Hello World-2

- Let's add an *entity* with *dodecahedron* as geometry
  - Make a fancy *material*
    - *flatShading: true*
    - *metalness: 1*
  - copy the entity and paste to html file
    - cleanup

Hello World-2

- Fancy materials only show up with fancy lights!
  - <a-light type="point" color="red" position="0 5 -4" intensity="1">
- See the faces of the new object appear and shine based on the light

Hello World-2

- Let's animate the light to highlight our fancy object
  - <a-animation>
    - Create as child of the fancy light

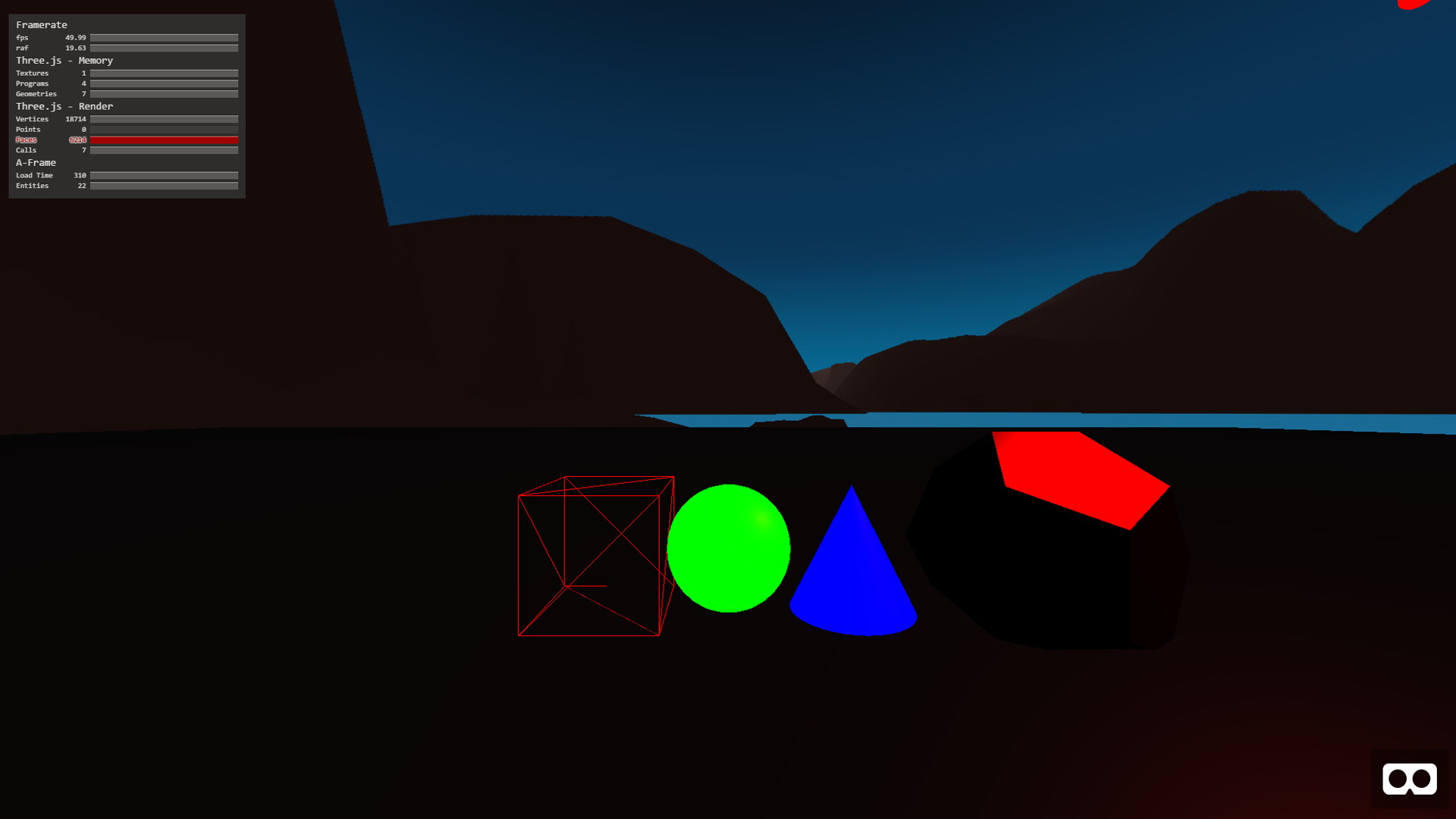
Hello World-2

- Setting up the animation

```
<a-animation
  attribute="position"
  from="-10 5 -4"
  to="10 5 -4"
  duration="5000"
  repeat="indefinite"
  direction="alternate"
  easing="ease-in-out-cubic">
</a-animation>
```

Hello World-2

- Questions?
- Checkpoint code in *helloworld2.html*


Framerate
fps 49.99
raf 19.63
Three.js - Memory
Textures 1
Programs 4
Geometries 7
Three.js - Render
Vertices 18714
Points 0
Faces 6214
Calls 7
A-Frame
Load Time 310
Entities 22

Hello World-3

- Our target is *helloworld3.html*
  - You can open it in another tab of the editor and browser

Hello World-3

- A-Frame comes with a built-in profiler
  - Just add *stats* attribute to <a-scene>
- We talked about A-Frame promoting reusability
  - What if we want multiple objects to have the same animation
  - <a-mixin>

Hello World-3

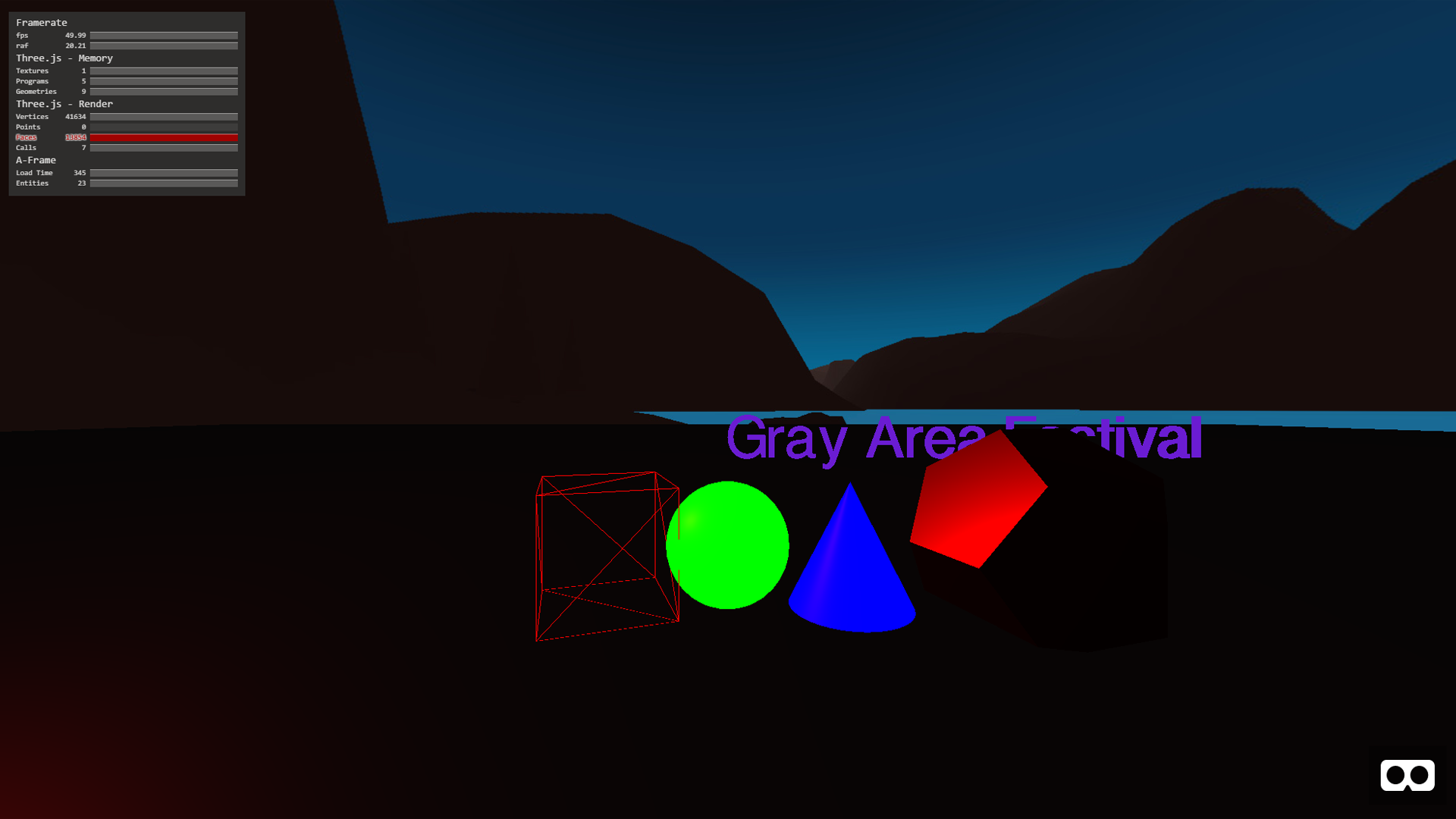
- We talked about A-Frame promoting reusability
  - What if we want multiple objects to have the same animation
  - <a-mixin>
    - Mixins go inside *<a-assets>*
    - Like textures mixins are assigned by *id*

Hello World-3

```
<a-mixin id="rotationAnimation"
  attribute="rotation"
  dur="10000"
  fill="forwards"
  to="0 360 0"
  repeat="indefinite">
</a-mixin>
```

Hello World-3

- Questions?
- Checkpoint code in *helloworld3.html*


Framerate
fps 49.99
raf 20.21
Three.js - Memory
Textures 1
Programs 5
Geometries 9
Three.js - Render
Vertices 41634
Points 0
Faces 18850
Calls 7
A-Frame
Load Time 345
Entities 23
Gray Area Festival

Hello World-4

- Our target is *helloworld4.html*
  - You can open it in another tab of the editor and browser

Hello World-4

- A-Frame allows for custom components
  - Also provides community-driven online component databases
  - Components are how you create complex interactions
  - Also requires javascript and eventually three.js knowledge

Hello World-4

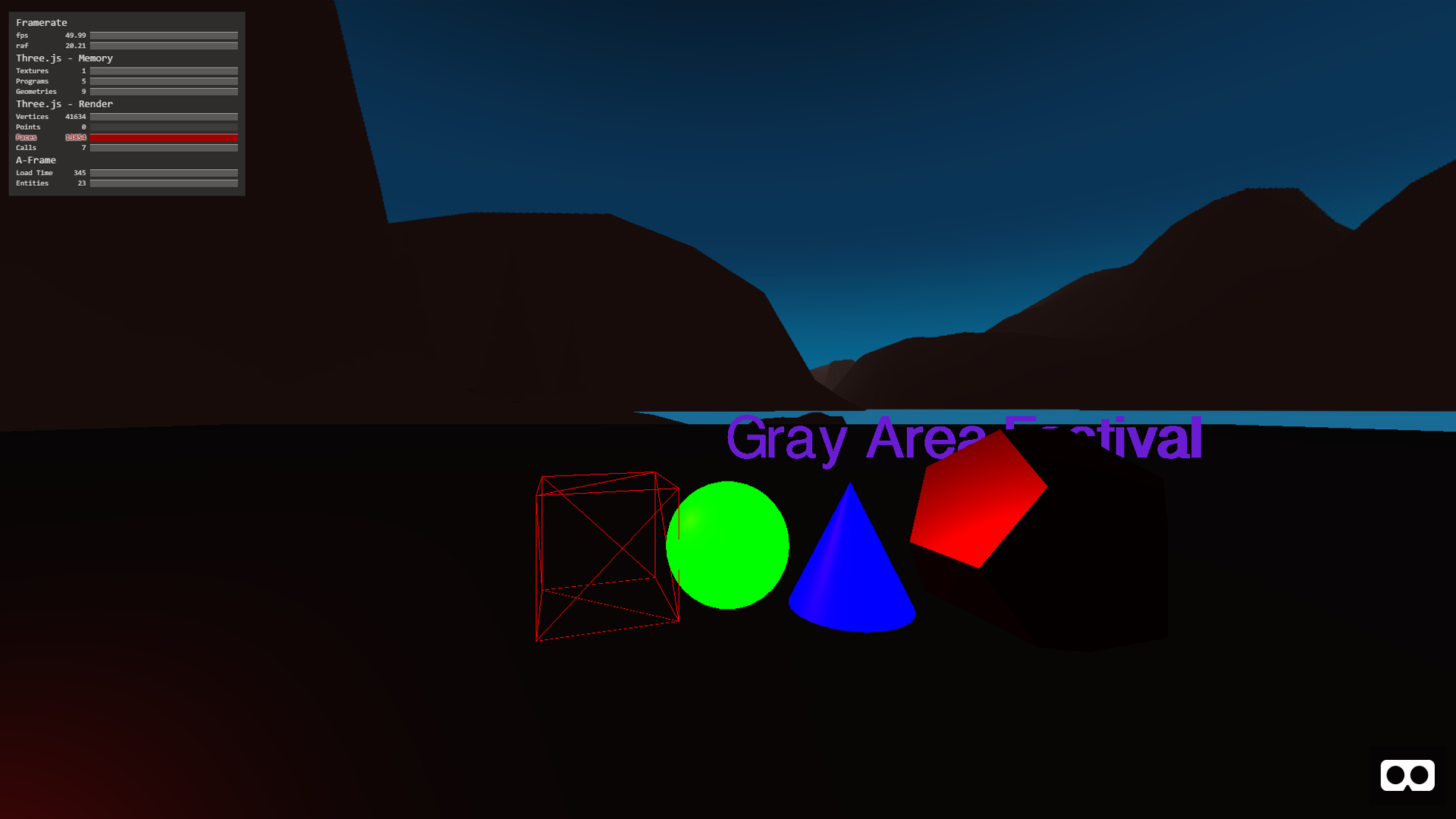
- A-Frame component registry
  - Let's choose *Text Geometry*
  - Multiple ways to integrate components
    - The code for *Text* Geometry is already in the provided package
    - Just link it as a regular script
      - <script src="./js/components/aframe-text-geometry-component.min.js"></script>

Hello World-4

- Using the entity-component relation
  - <a-entity text-geometry="value: Gray Area Festival" position="0 1 -6"></a-entity>

Hello World-4

- Questions?
- Checkpoint code in *helloworld4.html*


Framerate
fps 49.99
raf 20.21
Three.js - Memory
Textures 1
Programs 5
Geometries 9
Three.js - Render
Vertices 41634
Points 0
Faces 18850
Calls 7
A-Frame
Load Time 345
Entities 23
Gray Area Festival

Hello World-5

- Our target is *helloworld5.html* and *best-component.js*
  - You can open them in another tab of the editor and browser

```
AFRAME.registerComponent('best-component', {
  // public values, accesible through HTML
  schema: {},

  // initialization function
  init: function () {},

  // called when initialized and component is updated via setAttribute
  update: function () {},

  // called per render loop
  tick: function () {},

  // called when component is removed
  remove: function () {},

  // called when entity/scene is paused
  pause: function () {},

  // called when entity/scene is played
  play: function () {}
});
```

Hello World-5

- Let's create a js file under *components* folder and name it to *my-component.js*
  - Using the structure from previous slide let's create a component and add some console.logs for debug.
  - Link the js file in HTML
  - Create a new entity with a geometry and assign the new component
    - Observe the console logs

Hello World-5

- Setup the component such that
  - it changes the color of the object
    - One time, goes to *init*
  - Make object rotate with controllable speed
    - Continuous, goes to *tick*
  - Use *schema* to make the variables controllable in HTML

Hello World-5

- Questions?
- Checkpoint code in *helloworld5.html*
- End of A-Frame introduction
- Now we will switch to creating a city using a music file
- (optional) Take a backup of your index.html file

Music as a resource

- Time
- Pitch
- Timbre
- Rhythm
- Volume

Song City - Basics

- Copy contents of *songcity-basics.html* into *index.html*
  - Plug in your own music
  - It is a very barebones scene
    - Except for an entity that refers to *song-city component* and *audio-analyser* component.
    - So let's create a my-song-city.js component

Song City – Audio Analyser

- Our targets are
  - song-city-audioanalyser.js
  - songcity-audiaoanalyser.html
    - You can open them in another tab of the editor and browser

Song City – Audio Analyser

- We need something that can understand music
  - A-Frame Registry already has a component for that (it uses WebAudio API)
  - *aframe-audioanalyser-component.js* from components folder

Song City – Audio Analyser

- Create a new *entity* (in index.html)
  - Add *audioanalyser* attribute
    - audioanalyser="src: #song“
    - This feeds our music to the analyser

Song City – Audio Analyser

- In *my-song-city.js*
  - Connect the *analyser* through *schema*
  - Attach *onended* event to song
  - Attach *audioanalyser-beat* to the *analyser*
  - Add logic to *tick* to react to end of the song
  - Once done watch the console.logs

Song City – Audio Analyser

- Questions?
- Checkpoint codes in *song-city-audioanalyser.js* and *songcity-audioanalyser.html*

Song City – Buildings

- Our targets are
  - song-city-buildings.js
  - songcity-buildings.html
    - You can open them in another tab of the editor and browser

Song City – Buildings

- We will start creating geometry based on the frequency of the sounds when a beat is detected.
  - The volume level of each frequency slice defines size of the building
  - The frequency defines location in X
  - And time defines location in Z

Song City – Buildings

```
schema: {
  analyserEl: {type: 'selector'},
  maxScale: {default: 20},
  multiplier: {default: 100},
  frequencyDivisions: {default: 128},
  buildingMixin: {default: ''},
  layout: {default: ''},
  zSpeed: {default: 0.01},
},
```

Song City – Buildings

- document.createElement('a-entity');
  - Let's create a container
    - Set the position of the container based on time
    - Create buildings based-on frequency as children of the container
    - Assign the stylings
    - Control the number of buildings with variable

```
<a-mixin id="building"
  geometry="primitive: box"
  material="
    color: rgb(10, 20, 50);
    flatShading:true;
    shader:flat
  "
  scale-y-color="
    from: 30 30 30;
    to: 250, 250, 250;
    maxScale: 45
  "
  position="{x: 0, y: 0, z: 0}"
></a-mixin>
```

Song City – Buildings

- scale-y-color
  - Another component from the registry
  - Changes object's color based scale-y
  - Available in the *components* folder
  - Needs to be referenced in *index.html*

Song City – Buildings

```
<a-entity id="buildings"
  song-city="
    analyserEl: #analyser;
    maxScale: 50;
    multiplier: 0.06;
    frequencyDivisions: 16;
    buildingMixin: building;
    layout: layout-mixin
  "
  position="0 0 0"
></a-entity>
```

Song City – Buildings

- Questions?
- Checkpoint codes in *song-city-buildings.js* and *songcity-buildings.html*

Song City – Ambient Lights

- Our targets are
  - song-city-ambientlights.js
  - songcity-ambientlights.html
    - You can open them in another tab of the editor and browser

Song City – Ambient Lights

- Let's add some reactiveness
  - New component: *audioanalyser-volume-bind.js*
    - Refer to the js file in HTML.
  - Lighting will respond to music

Song City – Ambient Lights

```
<a-light
  type="ambient"
  color="#ffffff"
  audioanalyser-volume-bind="
    analyserEl: #analyser;
    component: light;
    property: intensity;
    max: 2.2;
    multiplier: .1
  "
></a-light>
```

Song City – Ambient Lights

- Questions?
- Checkpoint codes in *song-city-ambientlights.js* and *songcity-ambientlights.html*

Song City – Fancy Lights

- Our targets are
  - song-city-fancylights.js
  - songcity-fancylights.html
    - You can open them in another tab of the editor and browser

Song City – Fancy Lights

- Let's add some more reactiveness
  - Let's create two visible point lights and move them with music

Song City – Fancy Lights

```
<a-mixin id="light-ball"
  material="
    wireframe: true;
    color: #ffffff
  "
  geometry="
    radius: 5;
    segments-height: 8;
    segments-width: 8
  "
></a-mixin>
```

```
<a-entity id="fancy-lights">
  <a-light
    audioanalyser-volume-bind="
      analyserEl: #analyser;
      component: light;
      property: intensity;
      max: 0.1;
      multiplier: .01
    "
    position="-2000 2 1"
    type="point"
  >
    <a-sphere mixin="light-ball">
  </a-light>
</a-entity>
```

Song City – Fancy Lights

- Update schema for our component to refer the fancy lights (both javascript and HTML)
- Inside *tick* function move the lights on Z-axis with time and Y-axis with average volume

Song City – Fancy Lights

- Questions?
- Checkpoint codes in *song-city-fancylights.js* and *songcity-fancylights.html*

Song City – Mountains

- Let's add some mountain drawings
  - Using three.js logic we will create lines based on the volume that resembles outlines of mountains
    - One color of lines for the outline
    - Another color of lines as vertical
  - This is completely three.js usage
    - Underneath A-Frame is three.js

Song City – Mountains

```
// simple geometry composed of two dots in space (enough for a line)
var geometryMountainTop = new THREE.Geometry();
geometryMountainTop.vertices.push(
  positionPreviousMountainTop,
  positionCurrentMountainTop
);

// create a mesh from geometry and material as usual
var lineMountainTop = new THREE.Line(geometryMountainTop, materialMountainTop );
// document.querySelector('a-scene').object3D is "scene" from three.js point of view
document.querySelector('a-scene').object3D.add(lineMountainTop );
```

Song City – Mountains

- Questions?
- Checkpoint codes in *song-city-mountains.js* and *songcity-mountains.html*

Song City – Clouds

- Let's add some Cloud-like objects
  - Using three.js logic we will create 3D geometries based on the volume above the buildings
  - Once the music has ended all the collected points are turned into cloud-like abstract shapes.
  - This is completely three.js usage again

Song City – Clouds

- For this workshop we won't get into details but feel free to look at it later
  - Schema needs to be updated
  - Points for the clouds need to be collected/filtered
  - Faces need to be constructed from three points and faces should not cut each other

Song City – Clouds

- Questions?
- Checkpoint codes in *song-city-final.js* and *songcity-final.html*

Explorations

- What else can we add/change?
  - Colors and shaders
  - Geometries
  - VR
  - Different layout of buildings
  - Other components

Additional Resources

- https://aframe.io
- https://aframe.io/blog/
- https://twitter.com/aframevr
- https://aframevr-slack.herokuapp.com
- https://aframe.io/aframe-registry/
- https://github.com/aframevr/awesome-aframe
- https://www.meetup.com/A-Frame/
- https://www.facebook.com/groups/aframevr/