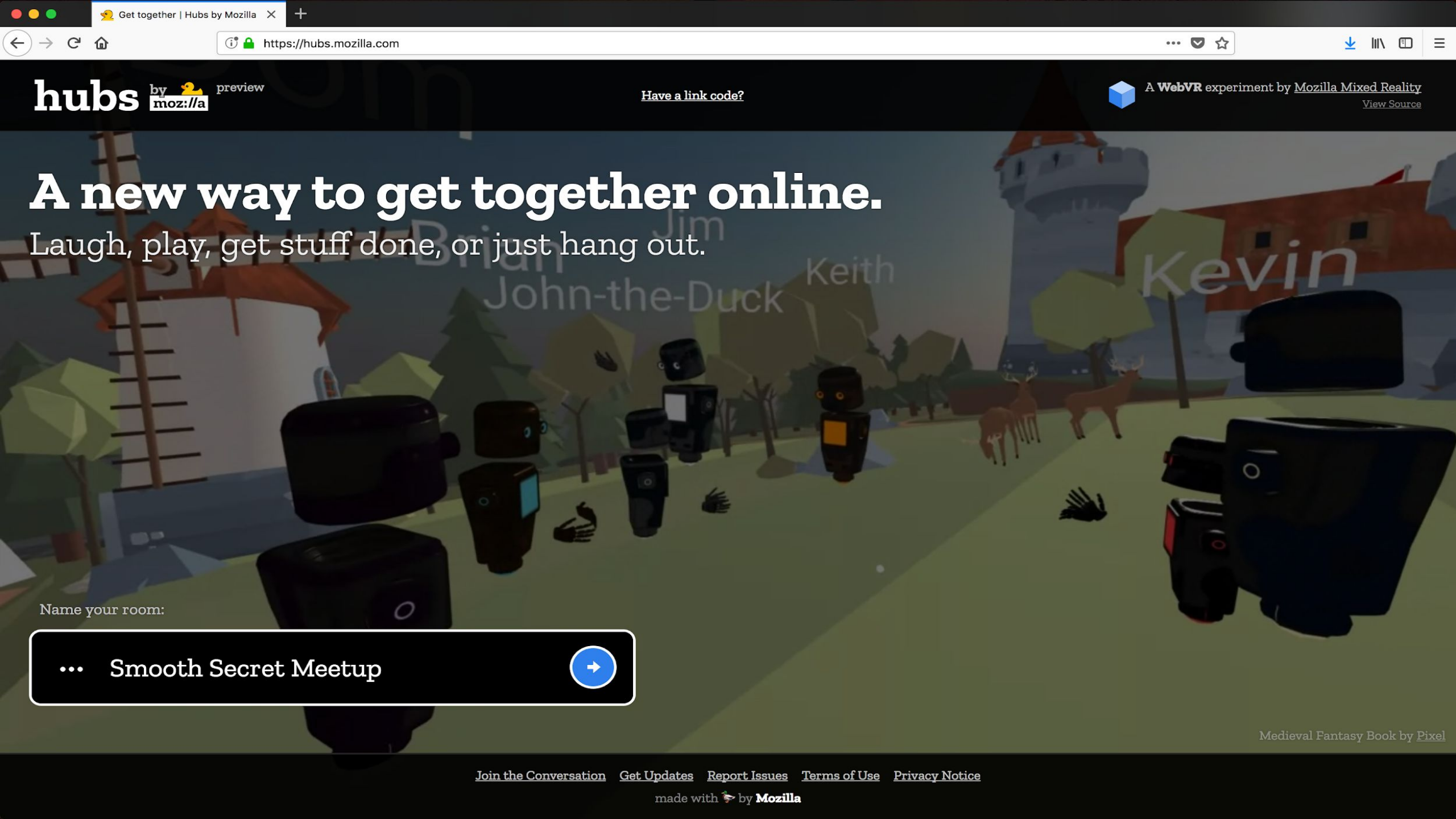
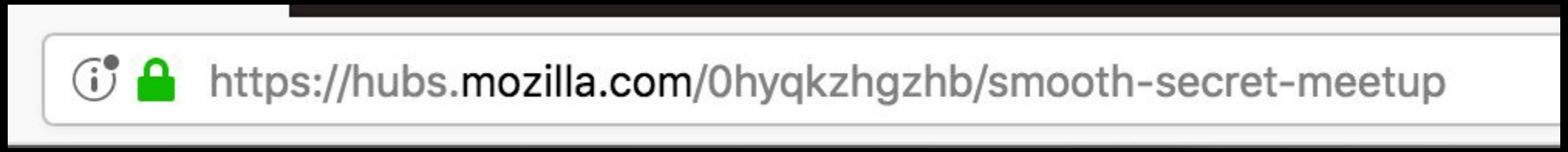
moz://a

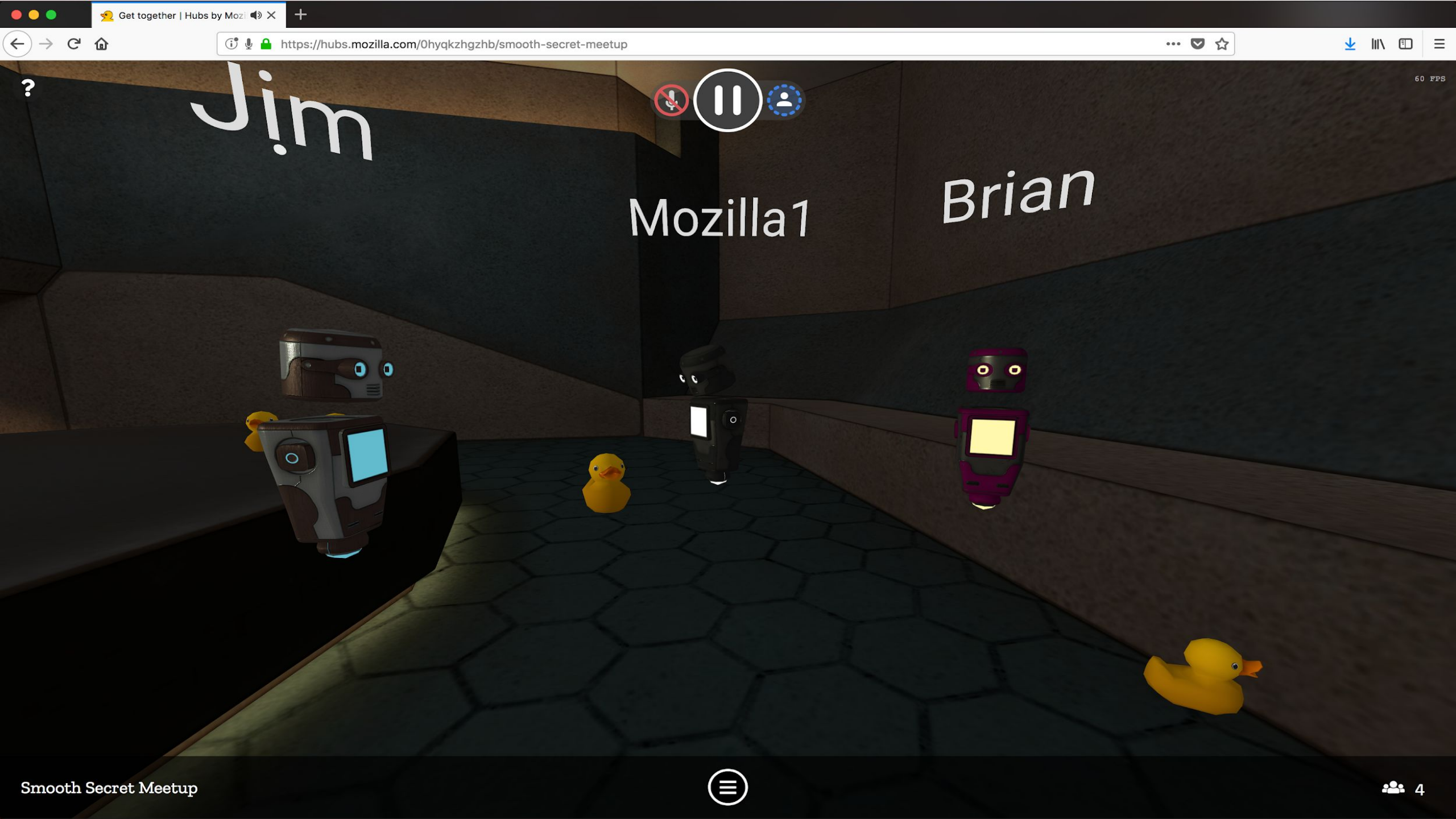
Publishing Virtual Worlds with glTF

Robert Long
@arobertlong


hubs
by
moz://a


Get together | Hubs by Mozilla
https://hubs.mozilla.com
hubs by moz://a preview
Have a link code?
A WebVR experiment by Mozilla Mixed Reality
View Source
A new way to get together online.
Laugh, play, get stuff done, or just hang out.
Jim
Brian
Keith
John-the-Duck
Kevin
Name your room:
Smooth Secret Meetup
Medieval Fantasy Book by Pixel
Join the Conversation
Get Updates
Report Issues
Terms of Use
Privacy Notice
made with by Mozilla

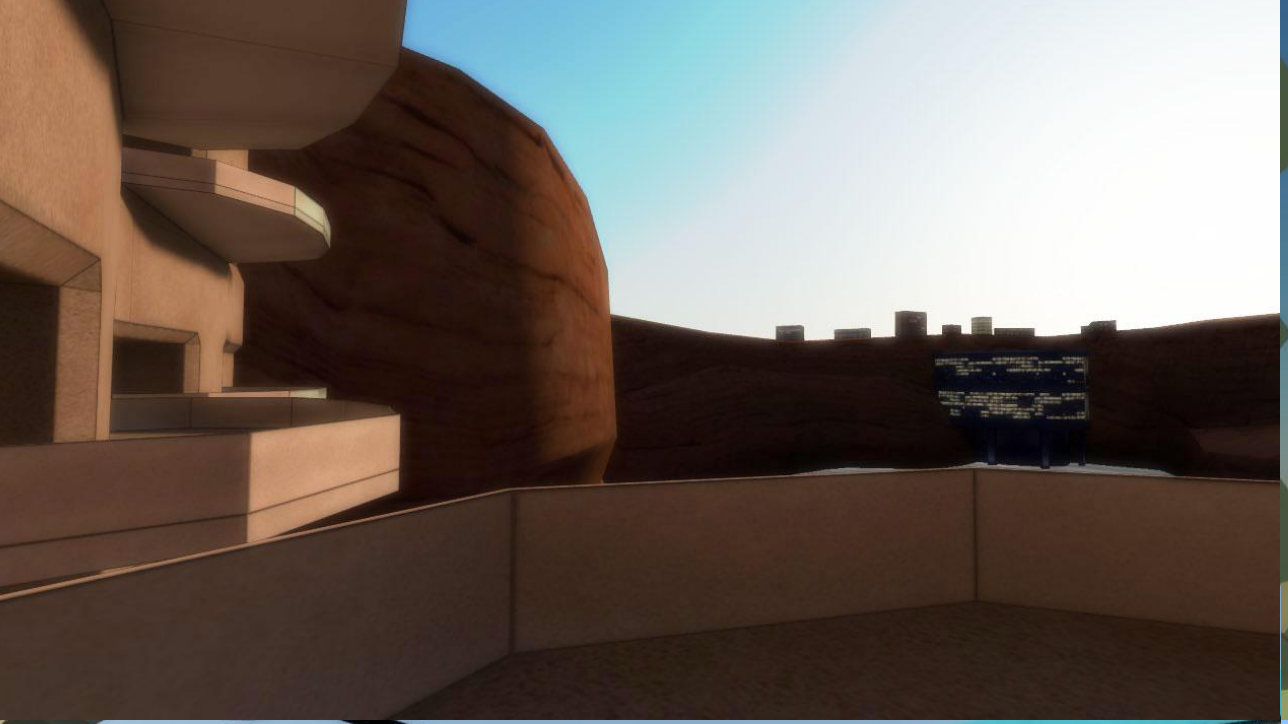

https://hubs.mozilla.com/0hyqkzhgzhb/smooth-secret-meetup


Get together | Hubs by Moz
https://hubs.mozilla.com/0hyqkzhgzhb/smooth-secret-meetup
?
Jim
Mozilla1
Brian
60 FPS
Smooth Secret Meetup
4


60 FPS

+

hubs by moz://a




moz://la

Your display name:

Robert

SAVE

Terms of Use Privacy Notice

Why glTF?

- Hubs is built on AFrame
- AFrame scenes are defined in HTML
- We wanted to add user generated content to Hubs (environments and avatars)
  - Limit what components you can use (ex. no camera component on a throwable duck)
  - No loading of untrusted Javascript for security reasons
- User defined HTML scenes weren't going to work and glTF seemed like the perfect fit.
- We were already using glTF for all of our assets, but didn't have a way to attach AFrame components to glTF nodes.
- Examples:
  - Playing animations
  - Player spawn points
  - Collisions
  - Duck spawners

Defining AFrame Scenes in glTF

```
<a-scene>
 <a-entity
   class="fan"
   position="1 3 0"
   shadow="castShadow: true; receiveShadow: true;"
   loop-animation="clip: Fan01"
 ></a-entity>
</a-scene>
```



```
{
 "nodes": [
   {
     "name": "fan",
     "translation": [1, 3, 0],
     "extras": {
       "MOZ_shadow": {
         "castShadow": true,
         "receiveShadow": true
       },
       "MOZ_loopAnimation": {
         "clip": "Fan01"
       }
     }
   }
 ]
}
```

Current AFrame gltf-model Component

- Attached to an <a-entity>
- Uses THREE.GLTFLoader
- Adds the loaded glTF scene as a child of the entity.

Current AFrame gltf-model Component

- Attached to an <a-entity>
- Uses THREE.GLTFLoader
- Adds the loaded glTF scene as a child of the entity.

- Problem:
  - Loaded ThreeJS scene is not exposed as AFrame entities.
  - No ability to attach AFrame components to nodes of the loaded scene.

```
<a-entity
   gltf-model="atrium.gltf"
></a-entity>
```



```
<a-entity
   gltf-model="atrium.gltf"
></a-entity>
```

Inflating glTF Scenes with gltf-model-plus

```
<a-entity
   gltf-model-plus="inflate: true; src=atrium.gltf;"
></a-entity>
```

```
<a-entity
   gltf-model-plus="inflate: true; src=atrium.gltf;"
>
...
   <a-entity
       class="fan"
       position="1 3 0"
       shadow="castShadow: true; receiveShadow: true;"
       loop-animation="clip: Fan01"
   ></a-entity>
...
</a-entity>
```

Registering Components with gltf-model-plus

```
AFRAME.GLTFModelPlus.registerComponent("MOZ_shadow", "shadow");
```

How it Works

- THREE.GLTFLoader puts glTF node.extras data on ThreeJS's object3d.userData
- gltf-model-plus calls:

```
el.setAttribute(componentName, object3D.userData);
```

- Example:

```
el.setAttribute("shadow", object3d.userData.MOZ_shadow);
```

Progressive Enhancement

- Hubs needs to run well on mobile phones and look good on desktops with high end GPUs.
- Low end platforms should use simpler lighting or no lighting to hit 60 FPS
- High end platforms should use physically based lighting and look as good as possible.

KHR_materials_unlit

- Render with flat shading on low end platforms.

```
"materials": [
    {
        "name": "red_unlit_material",
        "pbrMetallicRoughness": {
            "baseColorFactor": [ 1.0, 0.0, 0.0, 1.0 ]
        },
        "extensions": {
            "KHR_materials_unlit": {}
        }
    }
]
```

MOZ_alt_materials

- Define alternate materials that can be used to render a given mesh.

```
"materials": [
   {
       "pbrMetallicRoughness": ...,
       "extensions": {
           "MOZ_alt_materials": {
               "KHR_materials_unlit": 1
           }
       }
   },
   {
       "pbrMetallicRoughness": ...,
       "extensions": {
           "KHR_materials_unlit": {}
       }
   }
]
```

MOZ_alt_materials

- Lets the application decide the appropriate material to download and use at runtime.

- gltf-model-plus has a global and component “preferredTechnique” property

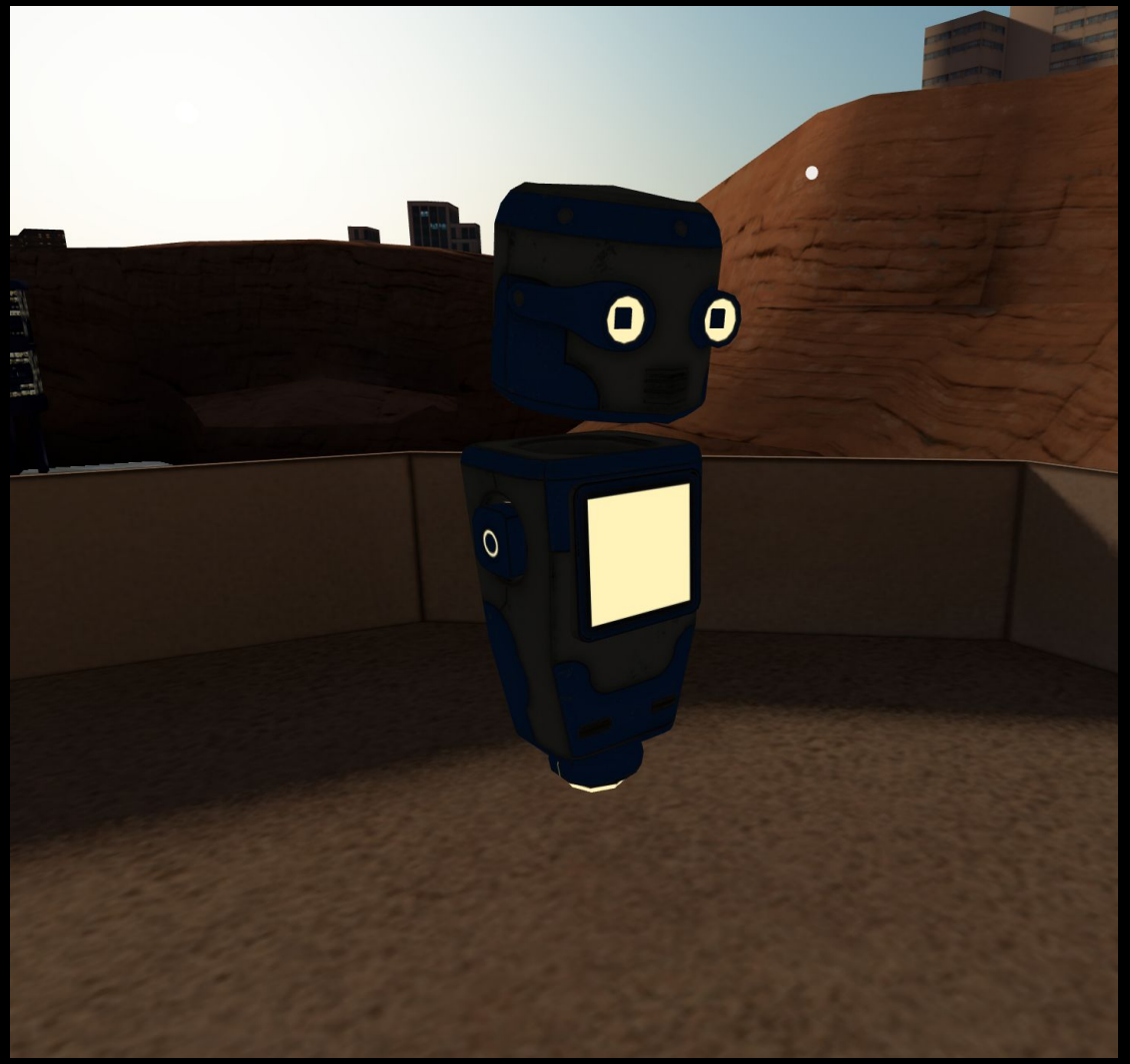
```
<a-entity
   gltf-model-plus="inflate: true; src=atrium.gltf; preferredTechnique=KHR_materials_unlit;"
></a-entity>
```

- We currently use AFRAME.utils.device.isMobile() to determine if we should use unlit materials. We would like to move to something that is based on GPU capabilities.

- Use .gltf vs .glb so that only the necessary textures are loaded at runtime.

Physically Based Rendering

KHR_materials_unlit



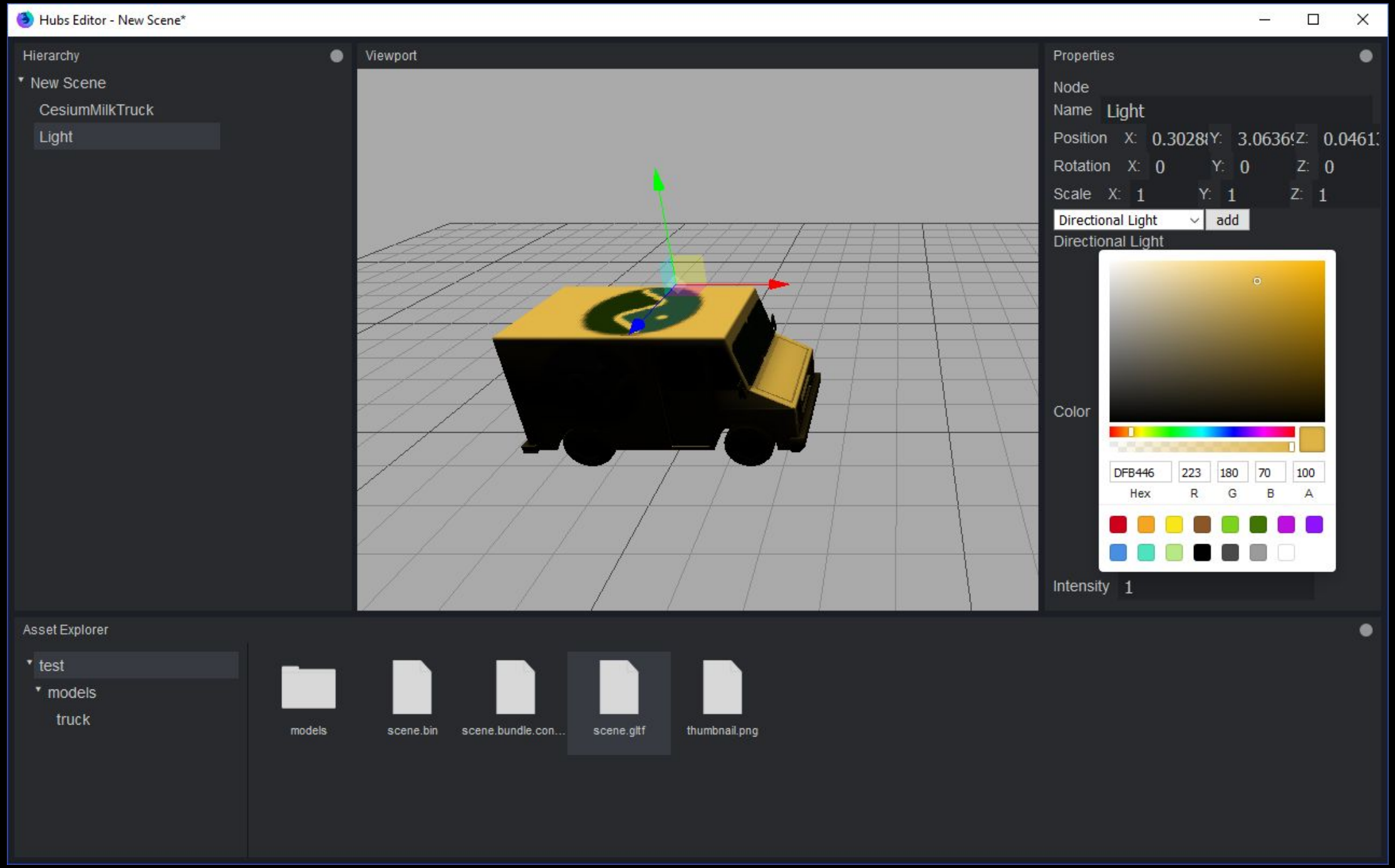
Automating Our Asset Pipeline

- gltf-bundle is a command line utility built out of modular parts
  - FBX2glTF (created by Pär Winzell)
    - Converts any FBX models to glTF
  - gltf-component-data
    - Adds component data stored in a separate JSON file
  - gltf-unlit-generator
    - Generates unlit materials from the PBR materials
    - Combines baseColorMap, occlusionMap, and emissive map to approximate the PBR material.
    - Add KHR_materials_unlit material as an alternative material via MOZ_alt_materials.
  - gltf-content-hash
    - Gives assets content-hashed file names to improve caching
    - mygltf-h8sg2.gltf -> gl8sd.bin and s2n3f.png
    - If only the gltf file changes then gltf file will be redownloaded.

Next Steps

- Automating our asset pipeline was only a partial success.
- Editing component data in JSON format isn’t ideal.
- Maintaining configuration files for the gltf-bundle tool isn’t great either.

- We’re building an editor on top of what we’ve learned with gltf-bundle to help improve this experience.


Hubs Editor - New Scene*
Hierarchy
New Scene
CesiumMilkTruck
Light
Viewport
Properties
Node
Name
Light
Position X: 0.3028 Y: 3.0636 Z: 0.0461
Rotation X: 0 Y: 0 Z: 0
Scale X: 1 Y: 1 Z: 1
Directional Light
add
Directional Light
Color
DFB446
223
180
70
100
Hex
R
G
B
A
Intensity
1
Asset Explorer
test
models
truck
models
scene.bin
scene.bundle.con...
scene.gltf
thumbnail.png

glTF Editor

Goals:

- One stop shop for composing and publishing glTF assets.
- Make importing/exporting content drag and drop or pasting a URL.
- Edit component data visually
- Make Blender/Maya/Substance Painter to glTF/Hubs iteration times quick and easy
- APIs for extending the editor’s functionality
- Integrate with your existing source control systems
- Preview scenes in local copy of Hubs
- Publish to Hubs

Thank You!

Hubs by Mozilla: hubs.mozilla.com
Mozilla Reality GitHub: github.com/MozillaReality
Social Mixed Reality Slack: webvr.slack.com #social channel
Mozilla Reality Twitter: @mozillareality

gltf-bundle: github.com/MozillaReality/gltf-bundle
gltf-model-plus: github.com/mozilla/hubs/blob/master/src/components/gltf-model-plus.js
hubs-editor: github.com/MozillaReality/hubs-editor

Robert Long
@arobertlong