

# Solar Stage

**A metaverse demo made with Uranus Tools for PlayCanvas**

Presentation by Leonidas Maliokas

# Solar Stage Demo

<https://solargames.io/demos/solar-stage>



# Solar Stage Demo

<https://solargames.io/demos/solar-stage>



# Solar Stage Demo

<https://solargames.io/demos/solar-stage>



# Solar Stage Demo

<https://solargames.io/demos/solar-stage>



# ABOUT ME



- Hi, I'm Leonidas! I'm a full stack ... PlayCanvas developer 😊
- I've been working with WebGL since 2012.
- First contact with PlayCanvas in 2013.
- Seeing the PlayCanvas editor for a first time was a💡 moment.
- I have been working full time with PlayCanvas for almost a decade.

## MORE... ABOUT ME



- Master at Civil Engineering, worked in hydraulic construction works.
- Turned my hobby, coding, into a full time job.
- Started as a web developer (PHP, Drupal, Node.js, Angular, React).
- My real passion... real time 3D rendering.

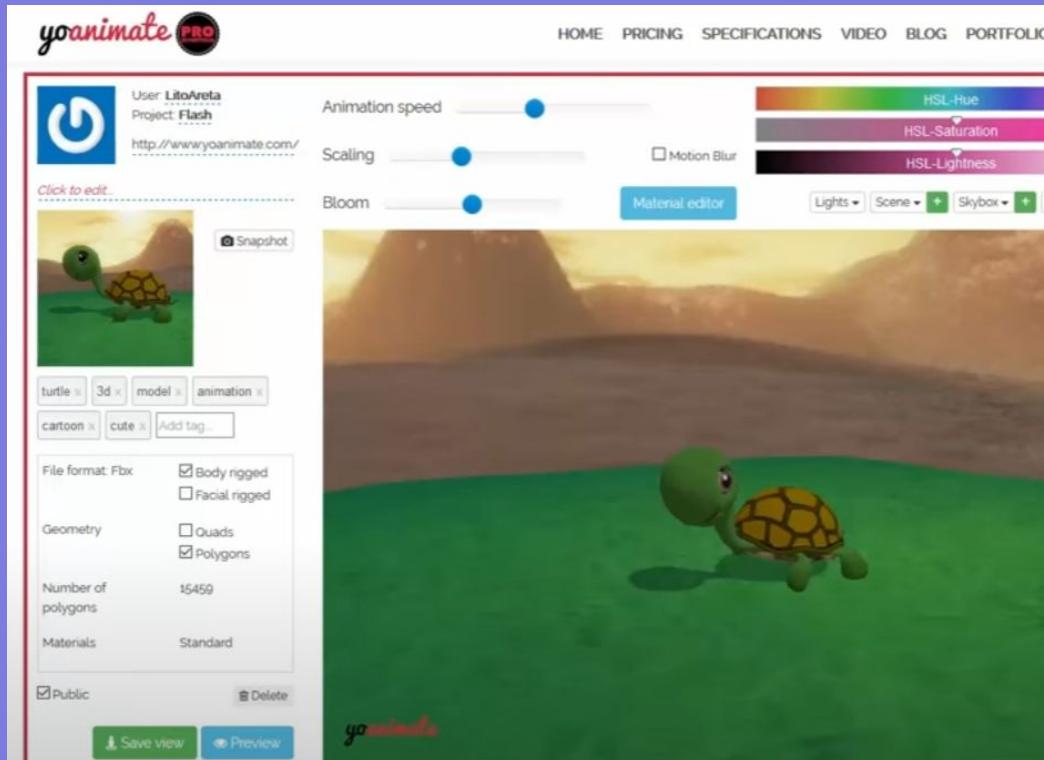
## MORE... ABOUT ME



- First attempts on Arch Viz with Unity.
- Authoring in Unity was great, but clients required accessibility.
- PlayCanvas to the rescue! What if we run them in a webpage?
- Ultimately learned game development through PlayCanvas.

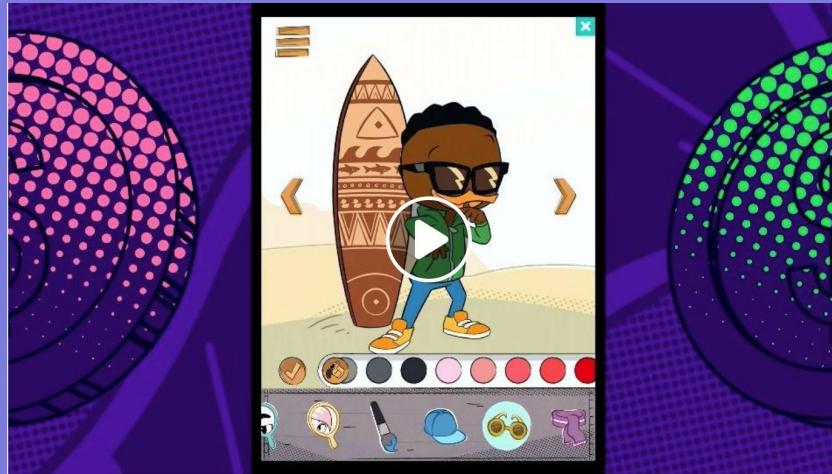
# Yoanimate

Sketchfab like portal, first PlayCanvas contract, 2014



# Disney Ducktales Creator

Avatar creator for a TV series, 2016



DuckTales ✓

July 3 ·

It's your ducky day! 🦆 Don a bill, wings, and a boat load of accessories with the DuckTales All Ducked Out! avatar creator, available now in the Disney XD App. ➡️

iOS: <http://di.sn/60008UO1s>

Android: <http://di.sn/60038UO1v>

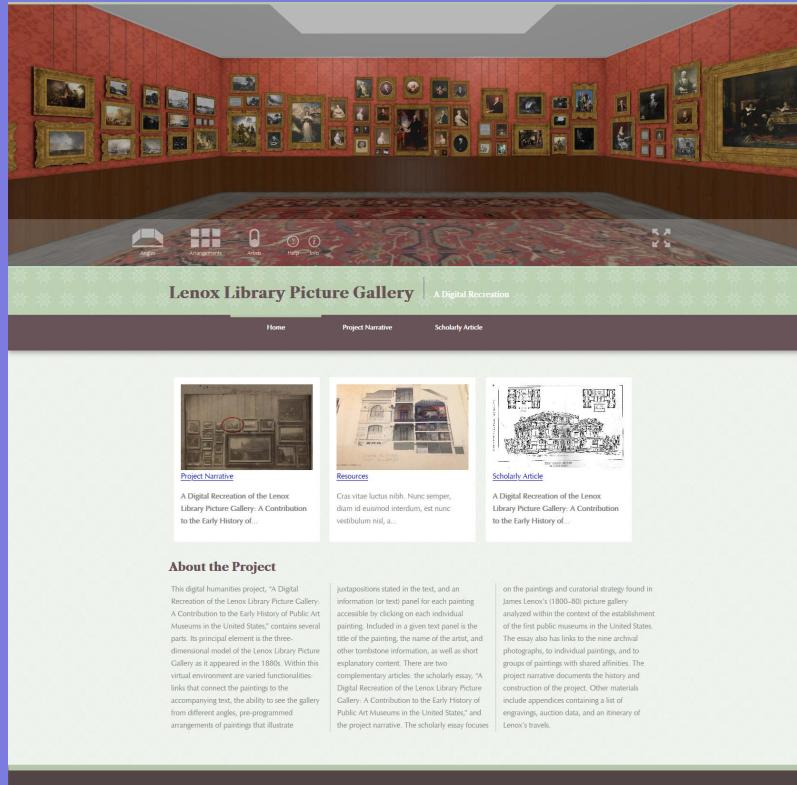
115K Views

1.2K Likes 51 Comments 122 Shares

Share

# Lenox Library Picture Gallery

## A digital recreation of an art museum, 2018



# Playing in Canvas Assets

The first PlayCanvas extensions marketplace, 2019



# Aritelio

Procedurally generated open world social game



# ABOUT THE DEMO

- Originally made in UE4.
- Scene included dynamic lights, shadows & reflections.
- Video playback on TV screens.
- High quality PBR textures.



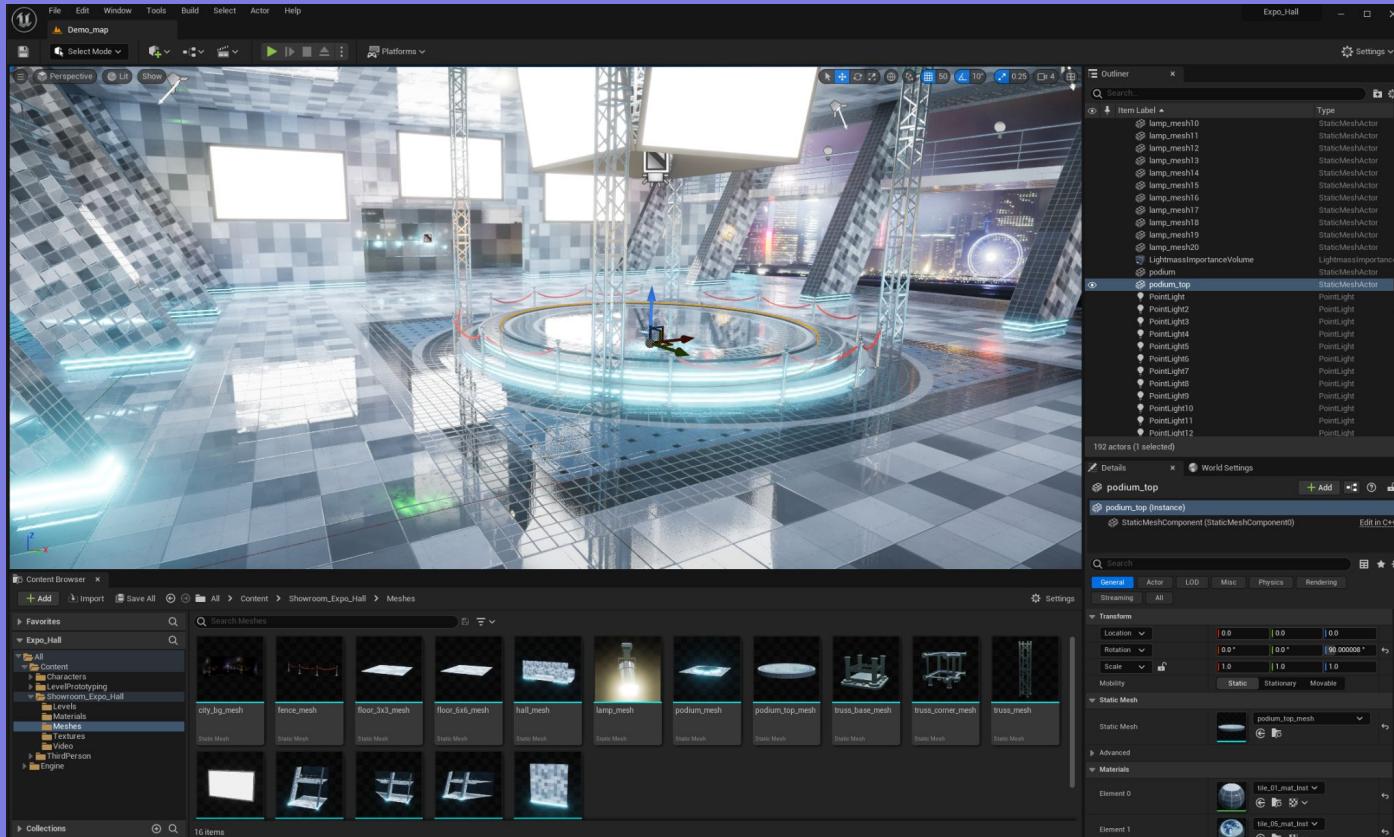
# WHY PLAYCANVAS OVER UNREAL?

- Accessibility: PlayCanvas builds require no installation / long download times.
- Shareable: Easily share urls for both dev and production builds.
- Performance: Optimized to run on older phones.
- Compatibility: Works like any other web based application.

# MAIN CHALLENGES

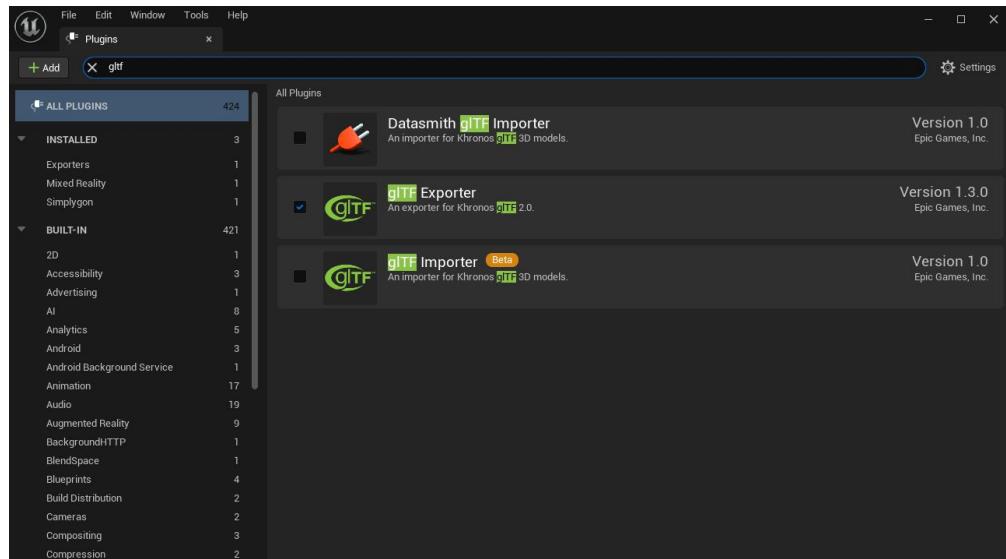
- Export effort to transfer assets and scene from Unreal to PlayCanvas.
- Resources size to reduce download times and memory allocation.
- Effects required to get similar render quality.
- Performance optimization to avoid low frame rates.

# Original scene in Unreal Engine



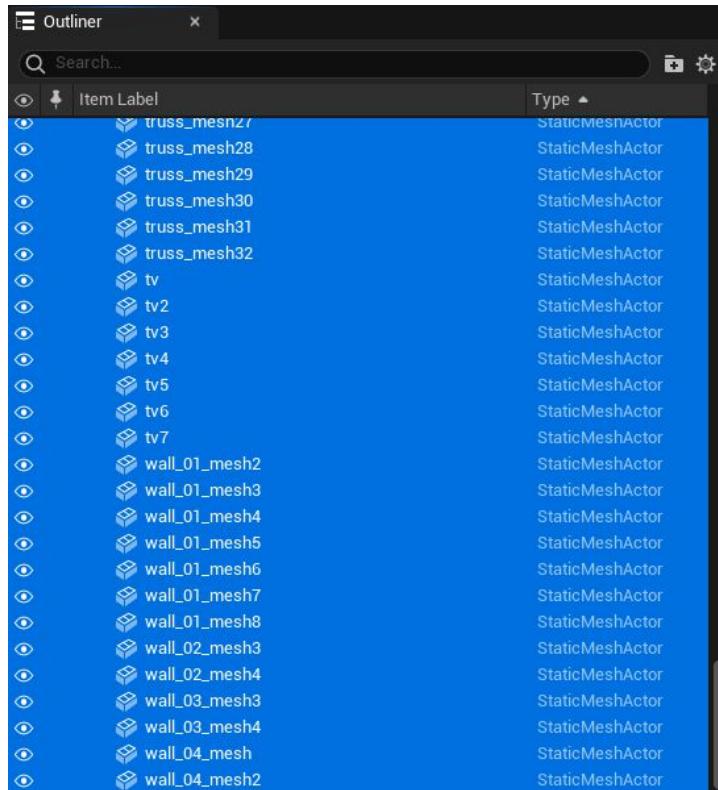
# EXPORTING FROM UNREAL

- glTF Exporter to the rescue!
- Powerful plugin that can export:
  - Models
  - Materials/textures
  - Actors
  - Lights/cameras (not supported in PlayCanvas though)



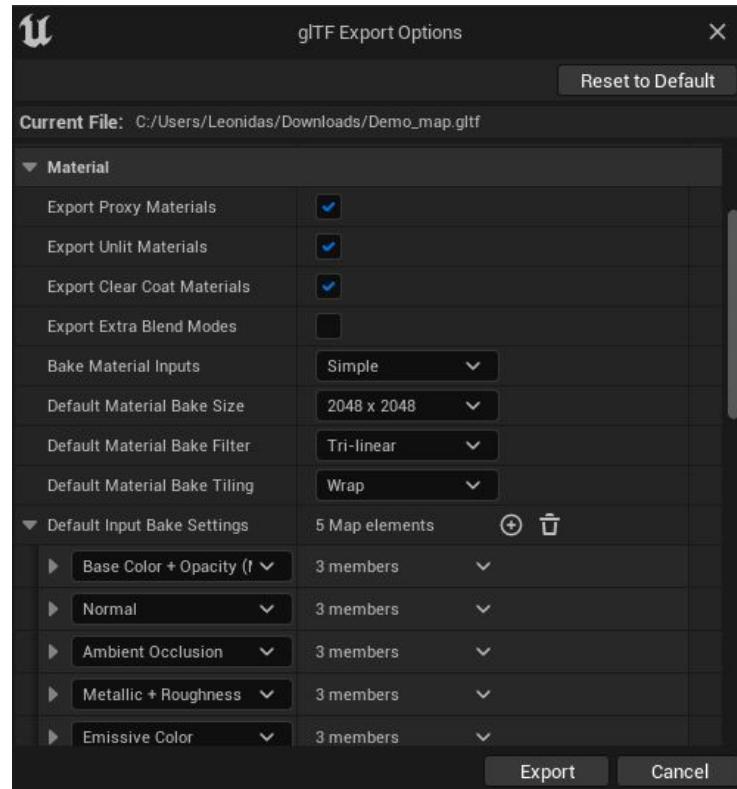
# EXPORTING FROM UNREAL

1. Enable the plugin in Unreal Editor  
(reload is required).
2. Select all of your Static Mesh  
Actors in the Outliner.
3. Go to File -> Export Selected...

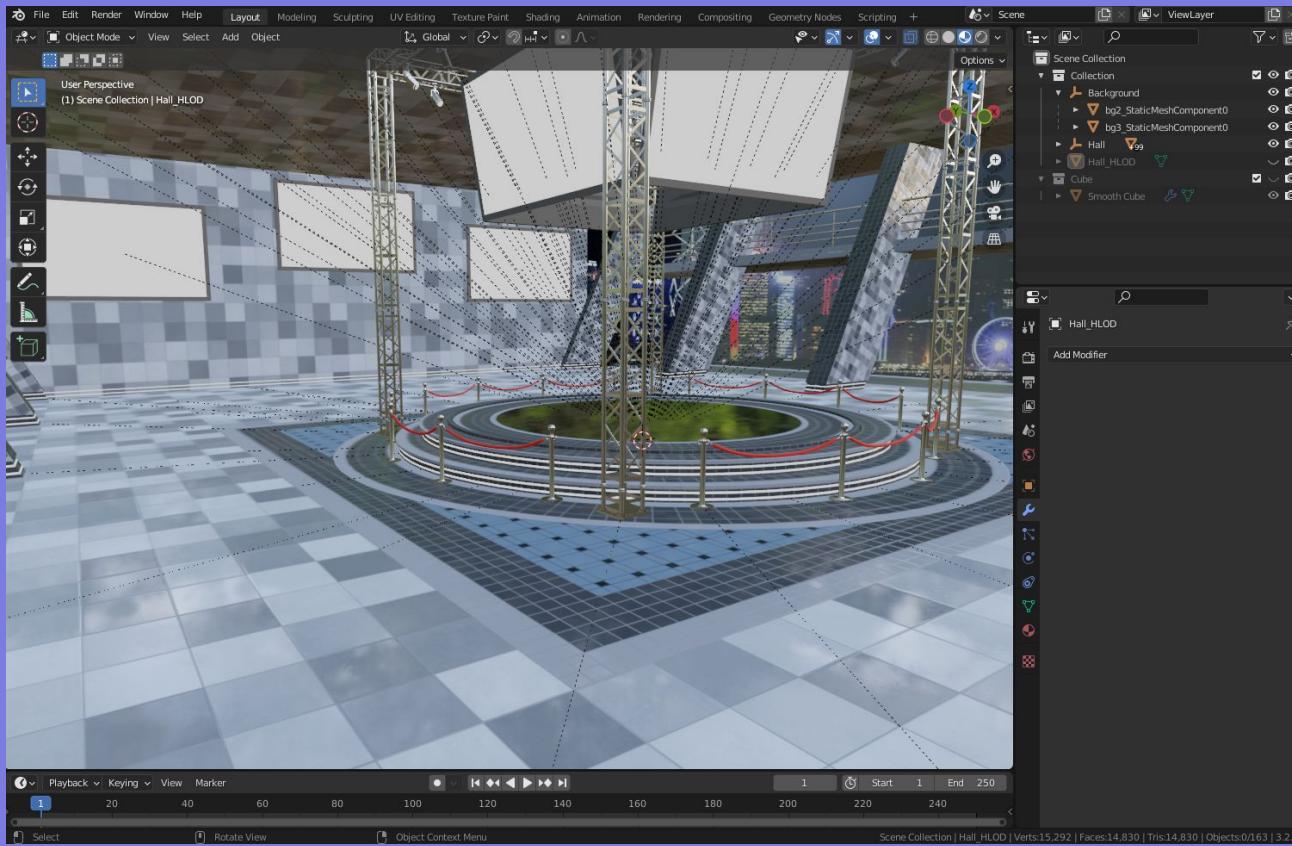


# EXPORTING FROM UNREAL

1. Select .gltf and a location on disk.
2. Add your required material channels.
3. Set the texture export resolution.



# Unreal scene imported in Blender

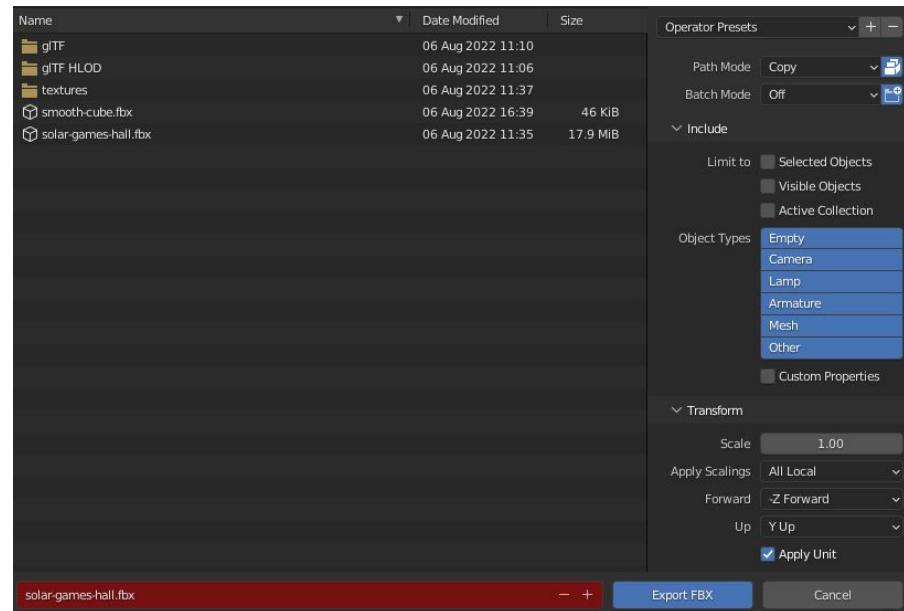


# PREPARE SCENE IN BLENDER

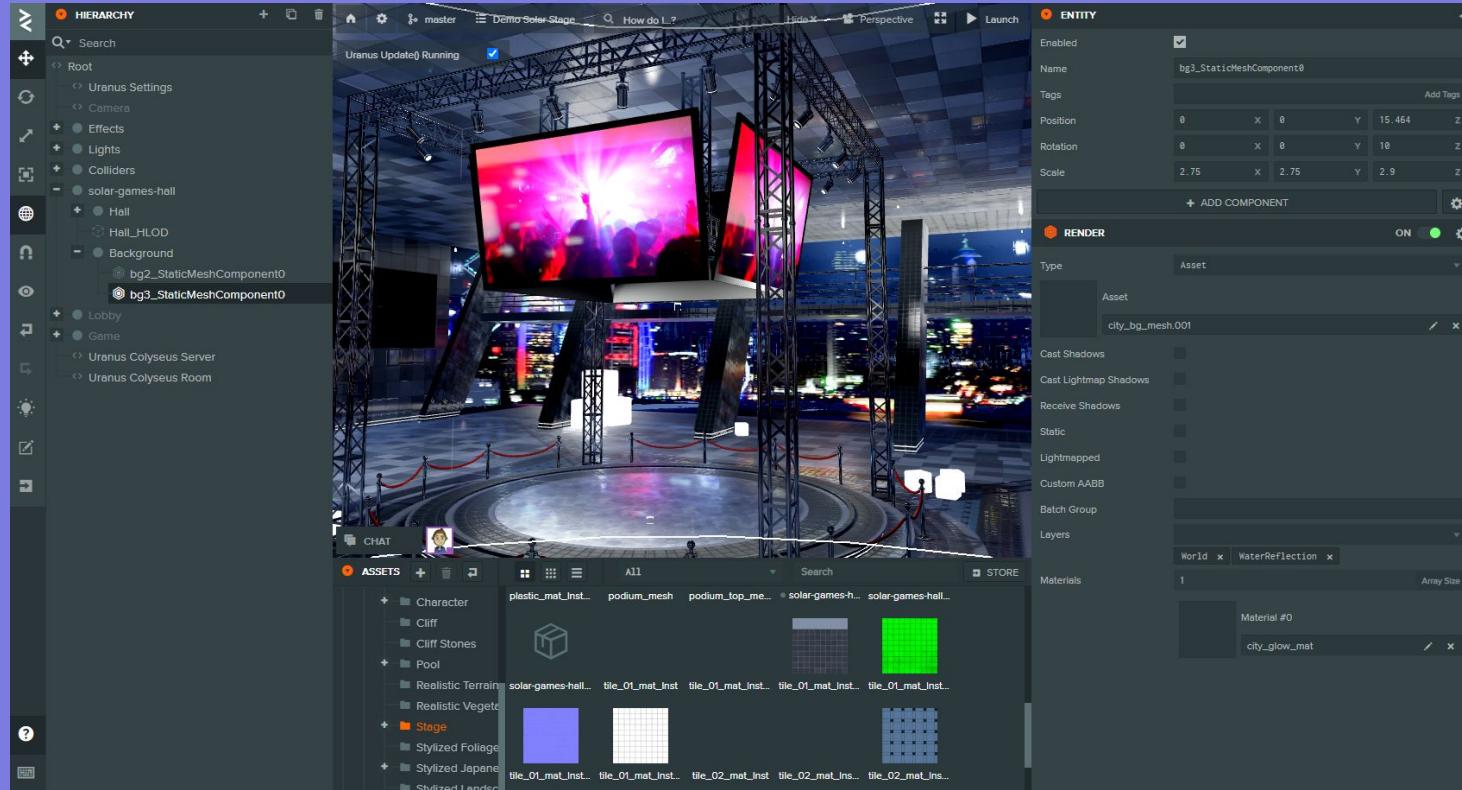
- Clean/delete unused nodes.
- Add empty nodes to better organize the hierarchy.
- Update mesh names with \_LOD suffixes where required.
- Add local texture file references to materials (used in FBX exporting).

# EXPORT TO FBX

- Can't yet import glTF models in the PlayCanvas editor.
- Select Path mode Copy and tick Embed Textures.

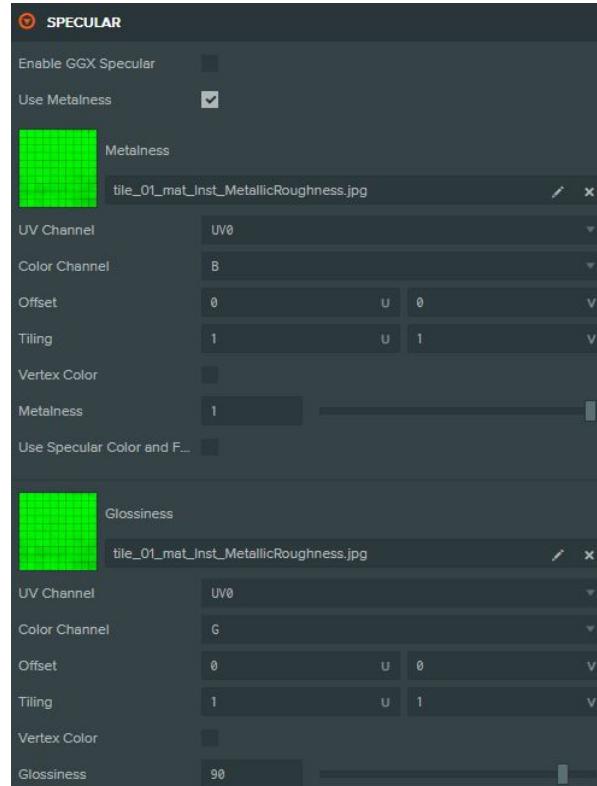


# Unreal scene imported in PlayCanvas



# IMPORT SCENE IN PLAYCANVAS

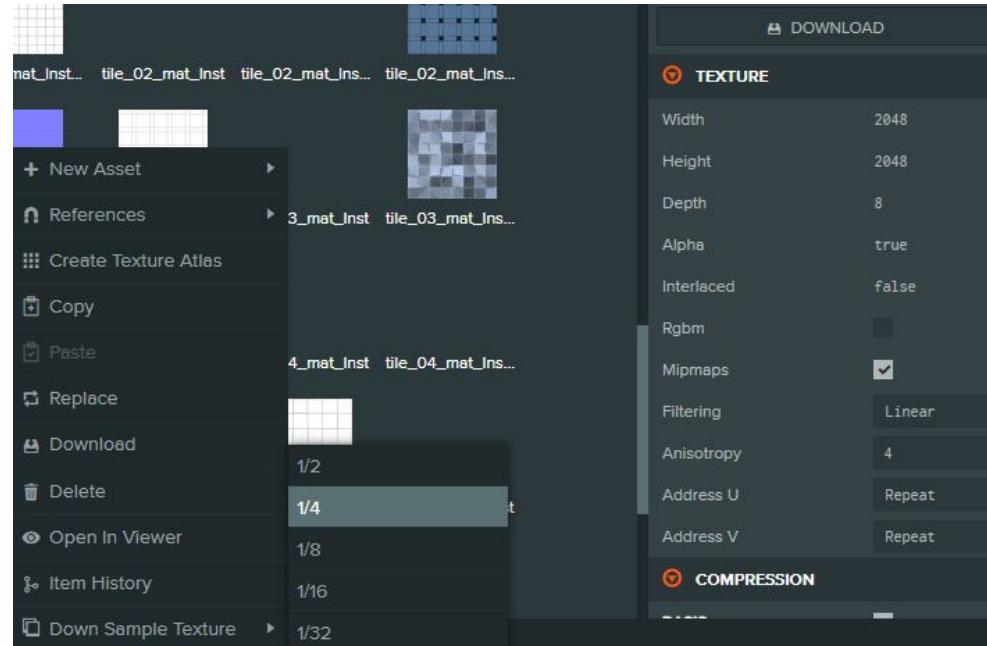
- Lights are imported separately and are hand placed.
- Update materials with the right channel mappings and blending.
- Set scene exposure and tonemapping.



# MANAGE RESOURCES

- Reduce model polycount.
- Downscale texture resolution.
- Use Basis Texture compression.
- Invert roughness maps!

PlayCanvas uses glossiness.



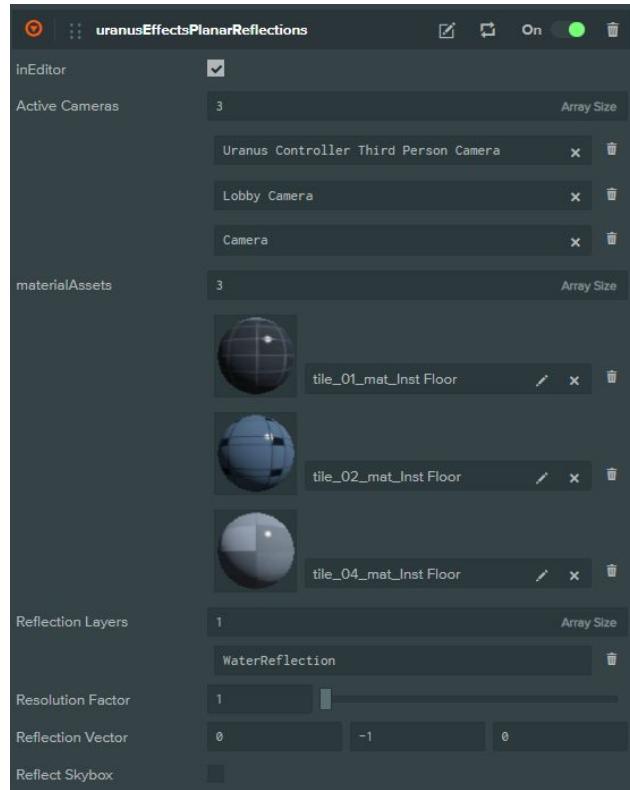
# REALTIME REFLECTIONS

- Original demo uses screen space reflections, too heavy on mobile.
- Planar Reflections for the floor.
- Box Reflections, rendering every 0.1ms, for everything else.



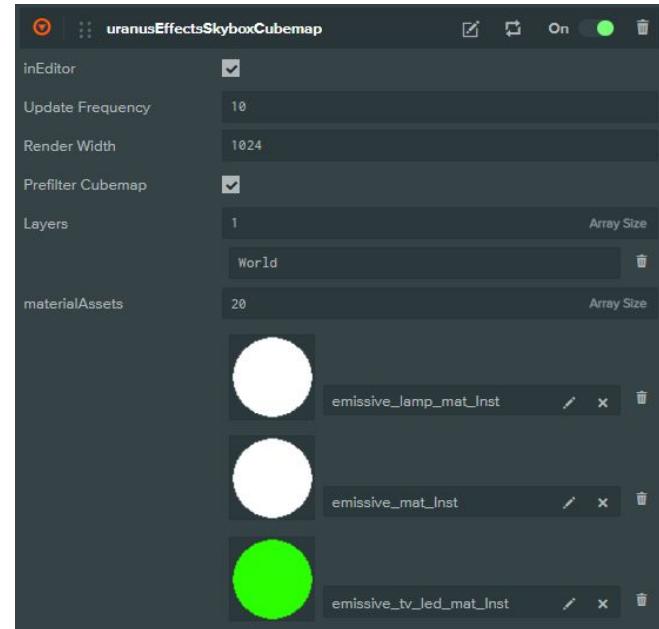
# PLANAR REFLECTIONS

- Effect included in the Uranus Tools SDK.
- Filter reflected entities with a layer.
- Reflections are rendered using an inverted secondary camera.



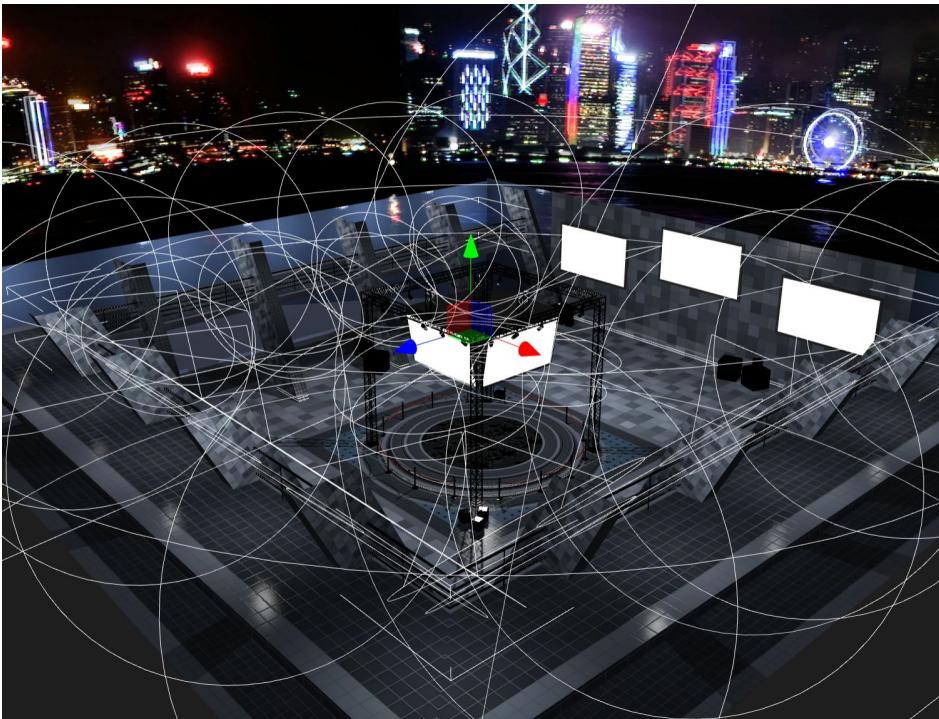
# BOX REFLECTIONS

- Effect included in the Uranus Tools SDK.
- Filter reflected entities with a layer.
- Reflections are rendered on a cubemap.
- Skip frames to improve performance.



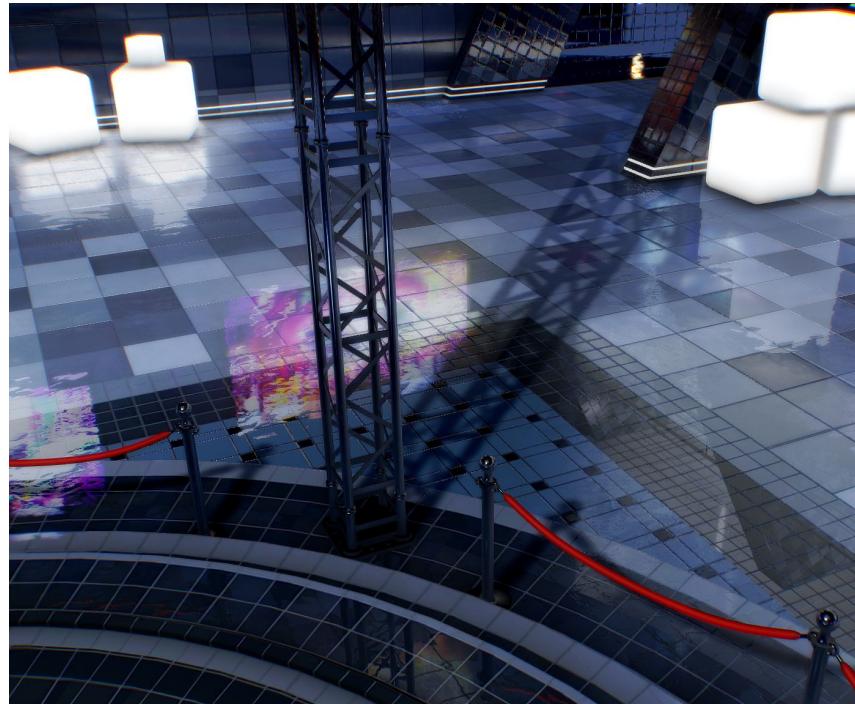
# REALTIME LIGHTS

- 30 dynamic point and spot lights.
- PlayCanvas clustered lights to the rescue! Enough said 😊
- Works great even on mobile.



# REALTIME SHADOWS

- If not careful draw calls can increase dramatically.
- Only some spot lights render shadows.
- Uranus Tools can render shadows at a lower frequency to solve that.



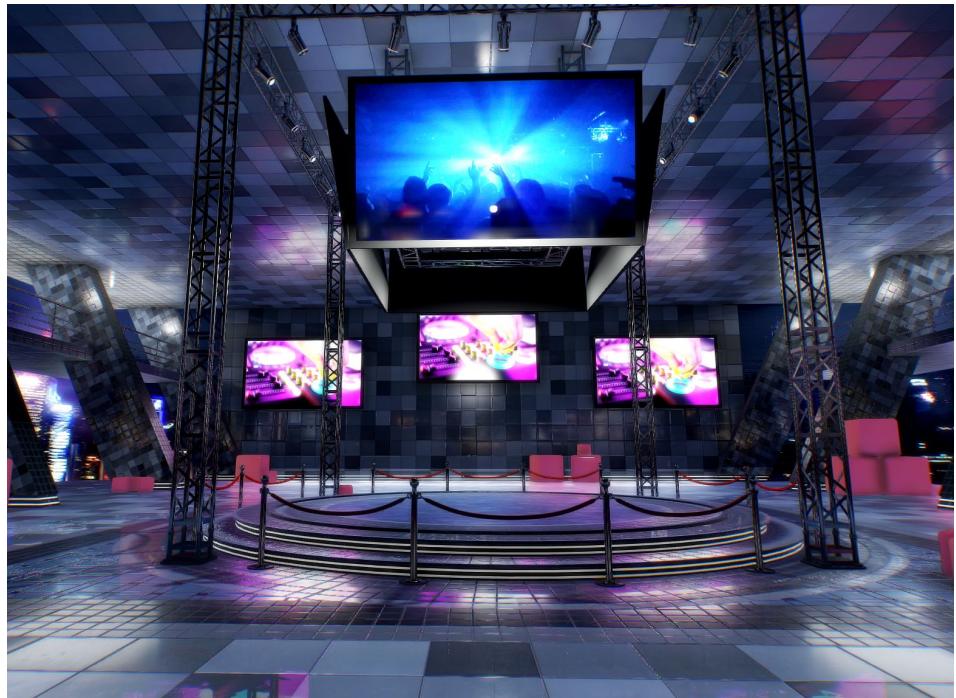
# VIDEO PLAYBACK

- Effect included in the Uranus Tools SDK (variation of the PlayCanvas Video Texture example).
- Local asset or remote url supported.
- Video streaming supported using a cloud provider like Agora.

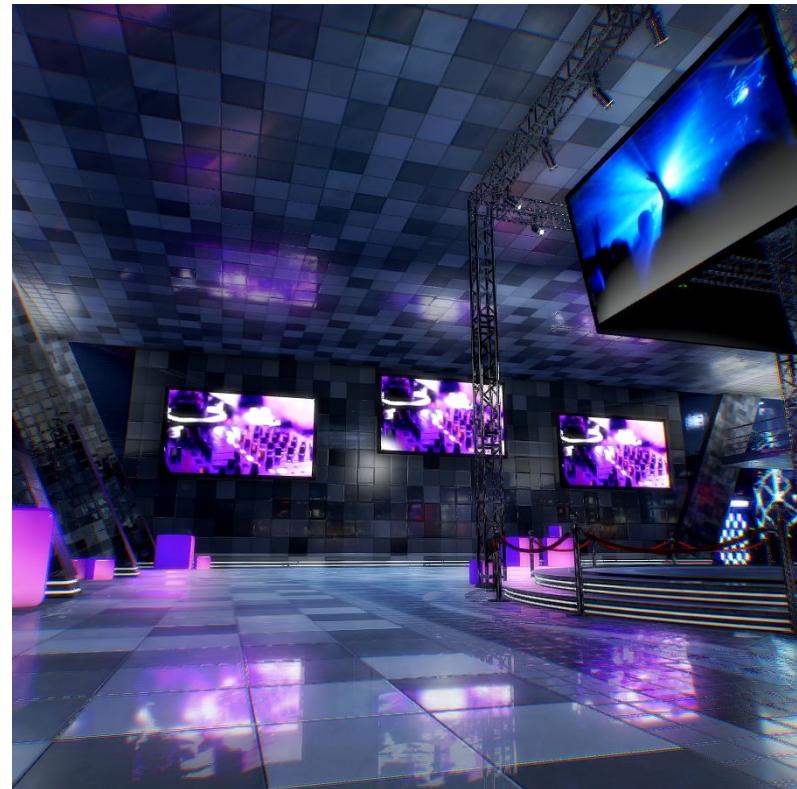


# POST EFFECTS

- Screen space Ambient Occlusion
- Bloom
- Chromatic Aberration
- Sharpen
- FXAA

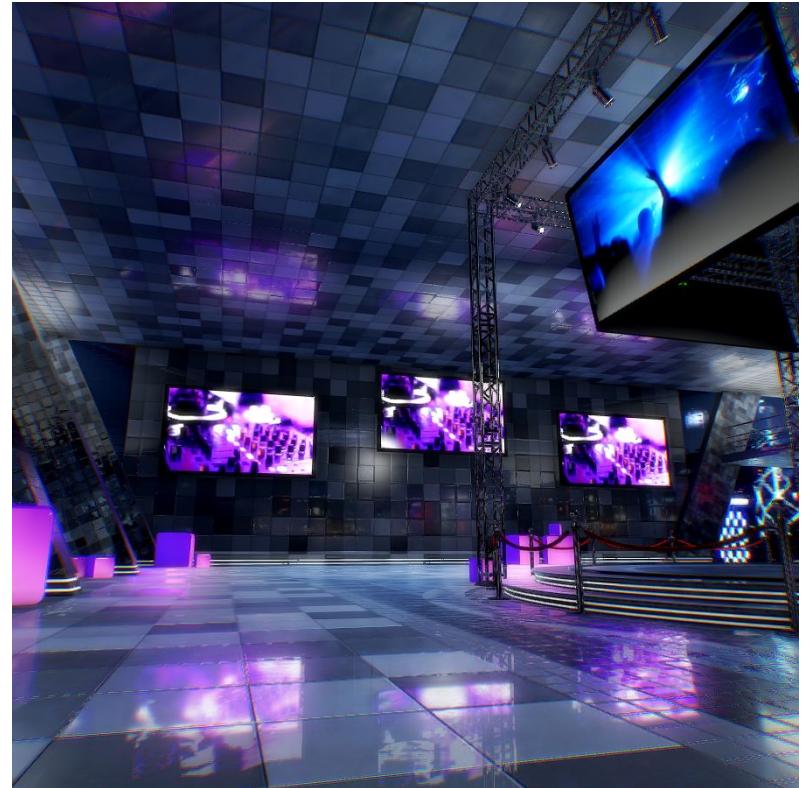


# BRIGHTNESS & CONTRAST



# BRIGHTNESS & CONTRAST

```
UranusEffectsMaterialContrast.prototype.prepare = function () {  
  
    if (!UranusEffectsMaterialContrast.overiddenGlobalChunks) {  
  
        UranusEffectsMaterialContrast.overiddenGlobalChunks = true;  
  
        // --- override global chunks  
        pc.shaderChunks.basePS = this.basePS();  
        pc.shaderChunks.endPS = this.endPS();  
    }  
  
    // --- update attributes  
    this.updateAttributes();  
};
```



## OPTIMIZE PERFORMANCE: DRAW CALLS

- Modular scene with too many draw calls: **250** with no shadows/reflections.
- Hardware instancing to the rescue! PlayCanvas batching could also be used, but there are limitations.
- Uranus SDK includes a plug and play auto instancer.

# OPTIMIZE PERFORMANCE: DRAW CALLS

- With all effects enabled the scene usually renders at ~120 draw calls with HW instancing.
- Big win in exercising exact control over shadows and reflections rendering.

DRAW CALLS		
Total	114	
Forward	108	
Skinned	0	
Shadow	0	
Depth	0	
Instanced	0	
Instancing Benefit	-0	
Immediate	0	
Misc	6	
Camera Drawcalls Limit	Disabled	▼

# OPTIMIZE PERFORMANCE: POLYCOUNT

- Luckily this scene has a relatively low number of polygons.
- HW Instancer supports level of details (LOD) for more complex scenes.

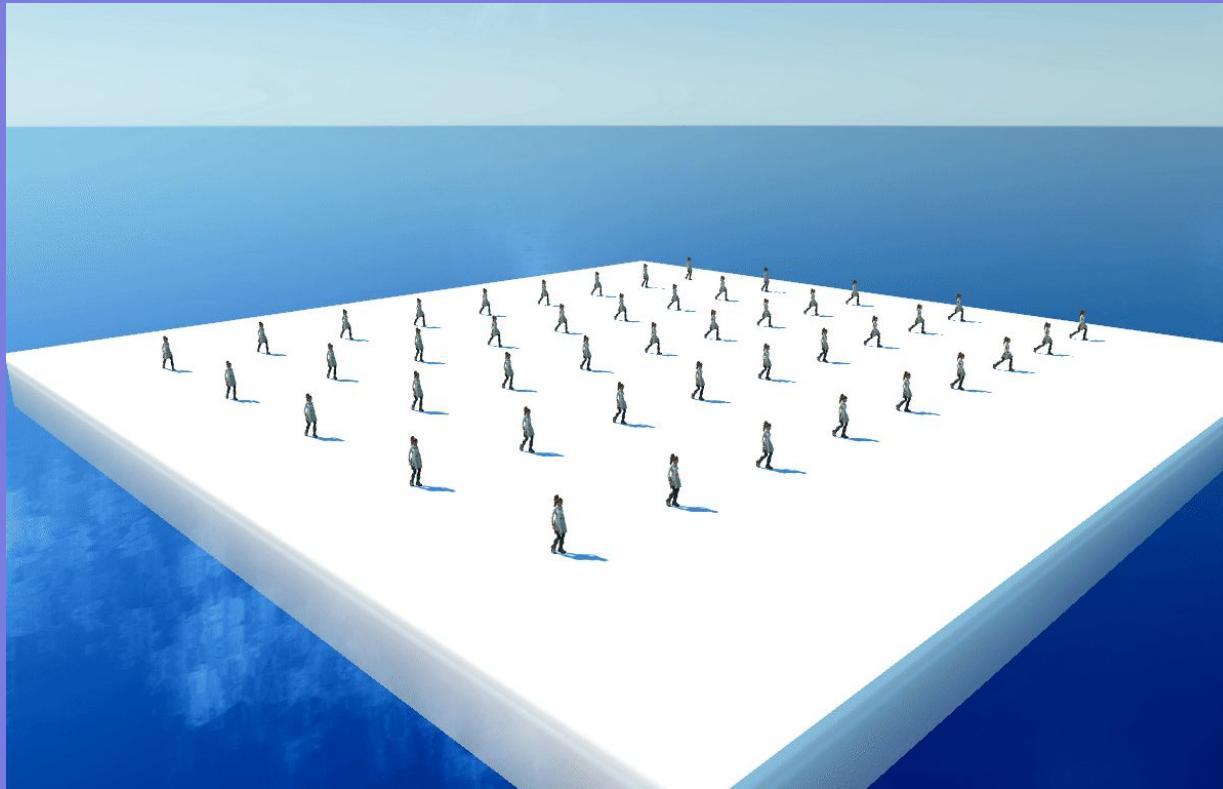
FRAME	
FPS	238
MS	4.17
Cameras	2
Cull Time	1.100
Sort Time	0.000
Shaders	47
Materials	42
Triangles	163,706
Other Primitives	0
ShadowMaps Updates	1
ShadowMaps Time	0.00
Update Time	0.10
Physics Time	0.00
Render Time	1.80
Forward Time	0.60

# OPTIMIZE PERFORMANCE: ANIMATION SKINNING

- Multiplayer support requires multiple avatars.
- PlayCanvas uses CPU skinning, expensive!
- Uranus Tools SDK animation LOD to reduce playback frequency on distant/non visible models.

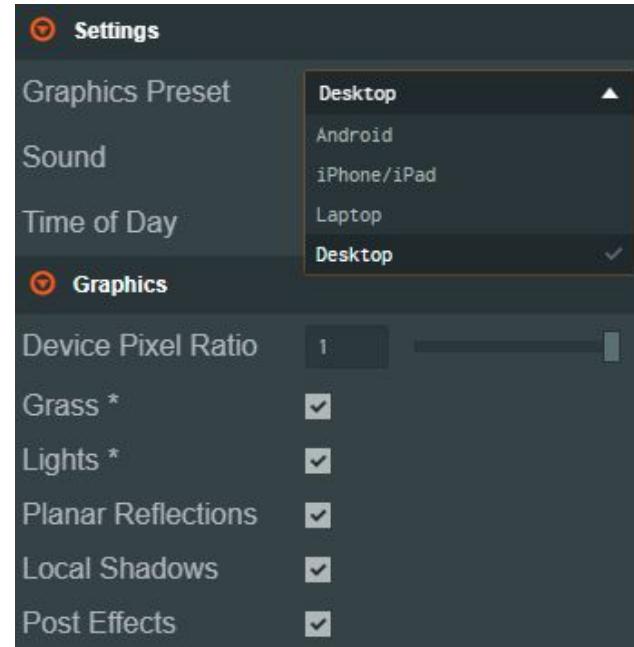
FRAME	
FPS	238
MS	4.17
Cameras	2
Cull Time	1.100
Sort Time	0.000
Shaders	47
Materials	42
Triangles	163,706
Other Primitives	0
ShadowMaps Updates	1
ShadowMaps Time	0.00
Update Time	0.10
Physics Time	0.00
Render Time	1.80
Forward Time	0.60

# Animation LOD in Uranus Tools



# OPTIMIZE PERFORMANCE: QUALITY PRESETS

- Demo should run from old Android phones to high end gaming PCs.
- Scale performance using Quality Presets.
- Auto detect preset but do allow the user to play with them! It's fun 😊



## ADDED FEATURE: REAL TIME MULTIPLAYER

- Using Colyseus.io! Easy to get started, free plan provided.
- Uranus Tools include drag and drop relay multiplayer (no code required).
- Third person controller automatically syncs with the server.
- Colyseus server script (TypeScript) is easy to extend with custom logic for authoritative multiplayer.

# ADDED FEATURE: REAL TIME MULTIPLAYER

- Broadcasting player state at a set frequency (e.g. 10 times a frame).
- State includes minimum information like position, angleY and velocity.
- Everything else, like animation or jumping, is assumed from state.

```
269
270 UranusControllerThirdPerson.prototype.getNetworkState = function () {
271
272     const currentPos = this.entity.getPosition();
273
274     this.networkState = {
275         x: currentPos.x,
276         y: currentPos.y,
277         z: currentPos.z,
278         angleY: this.currenRotation,
279         speed: this.animationBlend,
280     };
281
282     return this.networkState;
283 };
284
285 UranusControllerThirdPerson.prototype.syncState = function (state) {
286
287     if (!this.entity.enabled) this.entity.enabled = true;
288
289     this.remoteState.targetPos.set(state.x, state.y, state.z);
290     this.remoteState.targetAngleY = state.angleY;
291     this.remoteState.targetSpeed = state.speed;
292 };
293
```

# Cloud hosted Colyseus Server

Server Code

Uranus Tools SDK Relay

UPLOAD CREATE DEPLOY

The screenshot shows a dark-themed server code editor interface. At the top, there's a header with "Server Code" and "Uranus Tools SDK Relay". Below the header are three buttons: "UPLOAD" (with an upward arrow icon), "CREATE" (with a plus icon), and "DEPLOY" (with a circular arrow icon). On the left, there's a sidebar showing the project structure:

- arena.config.js
- arena.env
- index.js** (highlighted)
- package.json
- rooms
- RelayRoom.js

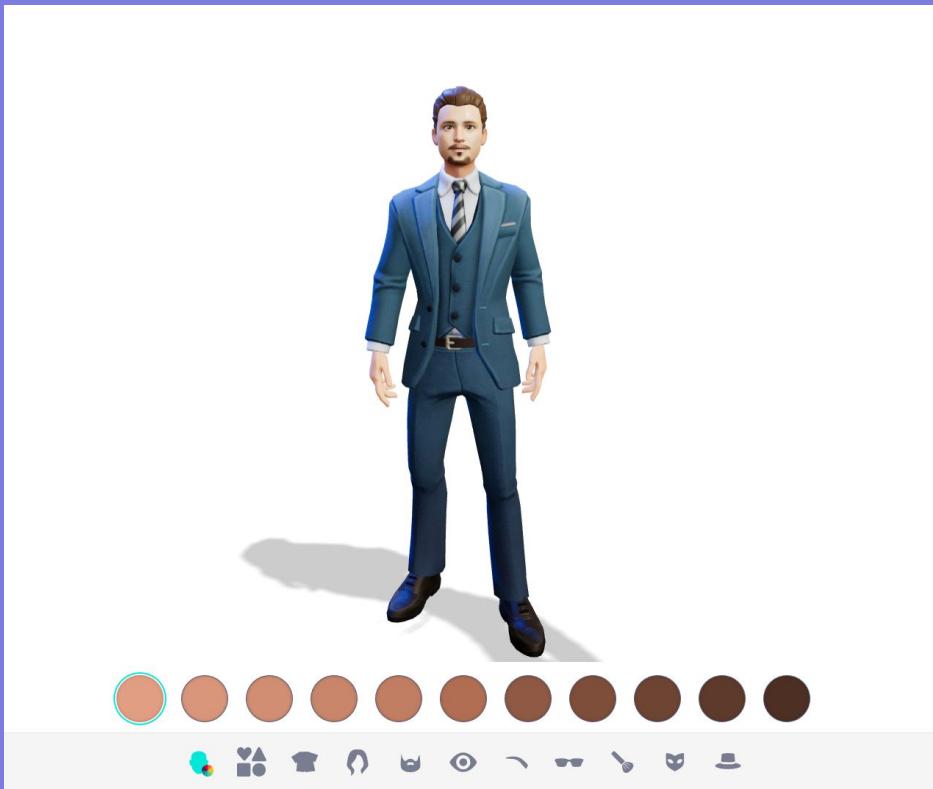
The main area displays the content of the "index.js" file:

```
1 "use strict";
2 var __importDefault = (this && this.__importDefault) || function (mod) {
3   return (mod && mod.__esModule) ? mod : { "default": mod };
4 };
5 Object.defineProperty(exports, "__esModule", { value: true });
6 /**
7 * IMPORTANT:
8 * -----
9 * Do not manually edit this file if you'd like to use Colyseus Arena
10 *
11 * If you're self-hosting (without Arena), you can manually instantiate a
12 * Colyseus Server as documented here: =I https://docs.colyseus.io/server/api/#constructor-option
13 */
14 const arena_1 = require("@colyseus/arena");
15 // Import arena config
16 const arena_config_1 = __importDefault(require("./arena.config"));
17 // Create and listen on 2567 (or PORT environment variable.)
18 arena_1.listen(arena_config_1.default);
19
```

## ADDED FEATURE: READY PLAYER ME AVATARS

- Super easy to use avatar creator! Sign up at <https://readyplayer.me> for your own custom url.
- A GLB container is loaded that contains model, materials and textures.
- Uranus Tools drag and drop script to easily get a networked, animated and user controlled Ready Player Me avatar.

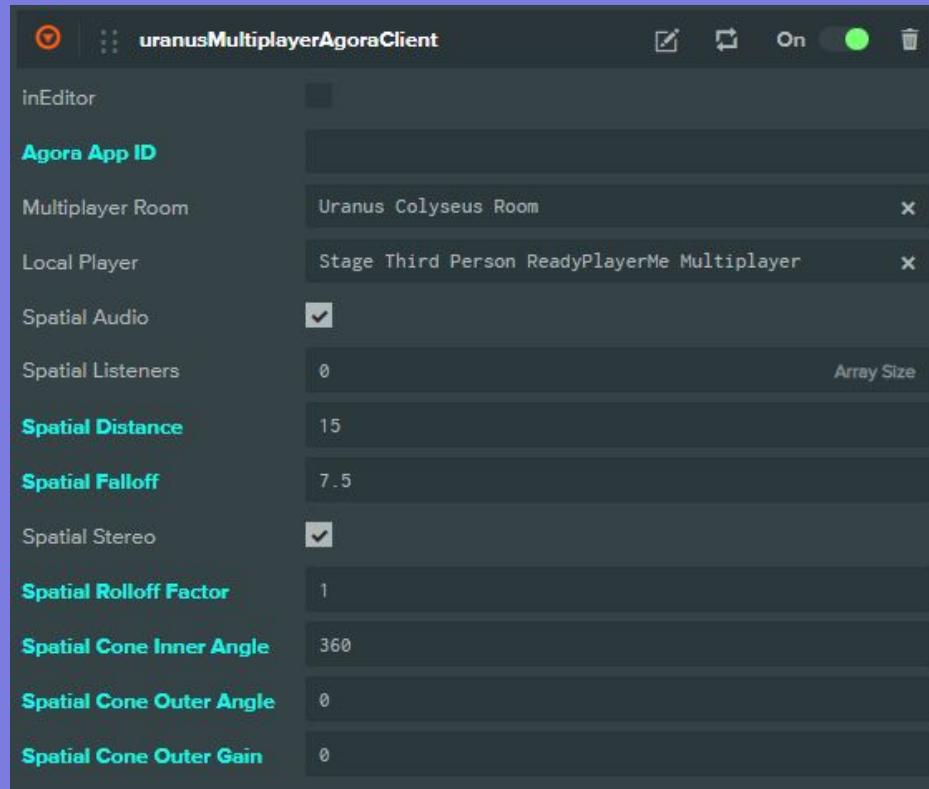
# Ready Player Me Avatars



## ADDED FEATURE: 3D SPATIAL AUDIO CHAT

- Using the powerful <https://agora.io> SDK for multi-party audio chat.
- Agora doesn't support spatial audio with their JavaScript SDK... browser Audio API to the rescue! Stereo Panners work with the agora audio stream.
- Uranus Tools provide a drag and drop mono/stereo spatial audio chat script.

# Uranus Tools Agora Client script



## SOME METRICS

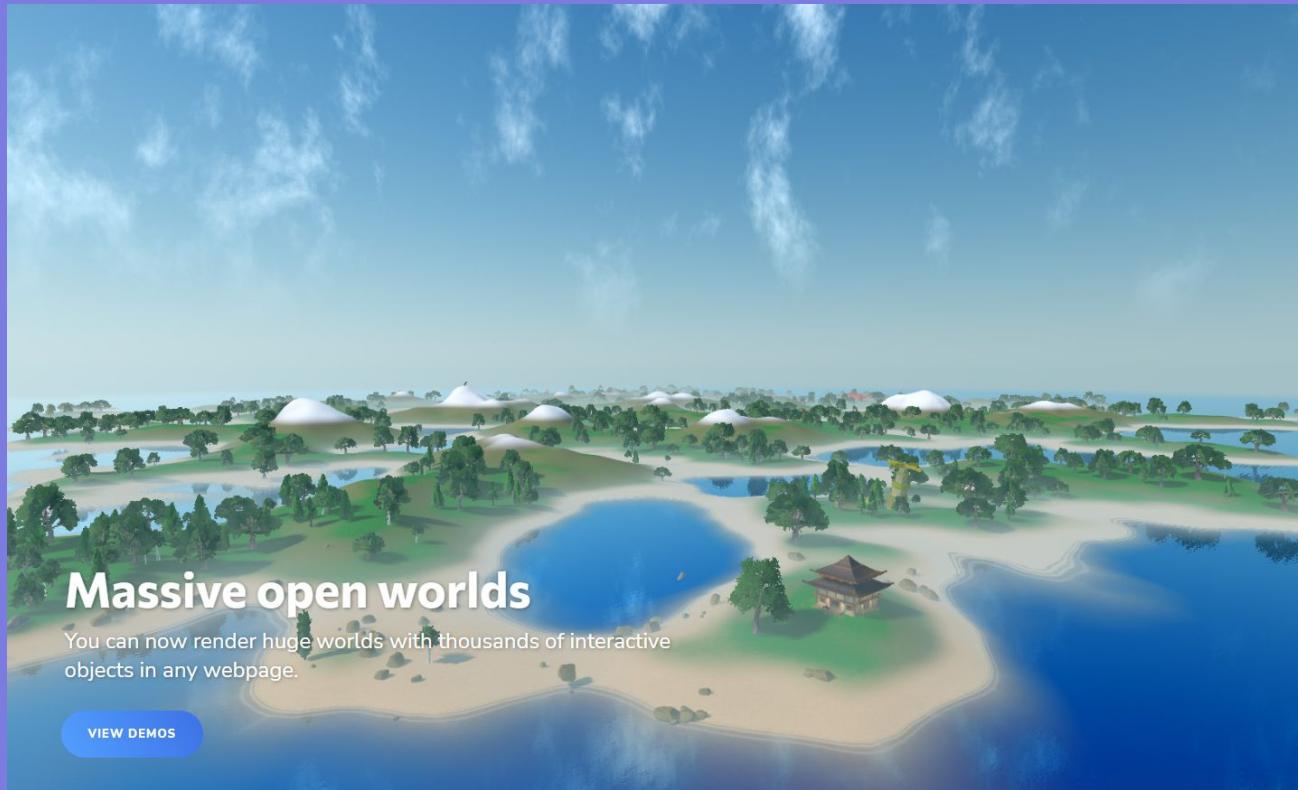
- Demo was built in less than a day! From the Unreal export to the first build.
- Zero code written! Everything was put in place using scripts available in the Uranus Tools SDK.
- Initial download size until the first frame 17MB (~7 seconds).
- Total download size 26MB (~13 seconds), with 80 http requests.

# Solar Games

<https://solargames.io>

The screenshot shows the homepage of the Solar Games website. At the top left is the Solar Games logo, which features a stylized sun icon with orange rays and the text "Solar Games" in a yellow, sans-serif font. At the top right are navigation links: "Demos", "Tools", "Mentorship", and "Contact". The main visual is a 3D rendering of a tropical island with sandy beaches, green trees, and rocky cliffs, set in a vast blue ocean with other smaller islands in the distance. Overlaid on this image is the text "Digital worlds for the Metaverse" in a large, white, sans-serif font. Below this, a smaller text block reads: "The most innovative companies choose our tools to bring their real-time 3D experiences to the Web." At the bottom left, there is a screenshot of a software interface, likely a 3D modeling or game development tool, showing a 3D preview window and various panels with data and controls.

# Uranus Tools for PlayCanvas



**Massive open worlds**

You can now render huge worlds with thousands of interactive objects in any webpage.

[VIEW DEMOS](#)

# Uranus Tools for PlayCanvas



## Gorgeous AAA rendering

Deliver high fidelity graphics in the browser, with real-time lighting and dynamic reflections, that scale from low end mobile devices to high end desktop computers.

[VIEW DEMOS](#)

# Uranus Tools for PlayCanvas

## Special Effects

A library of special material effects and shaders to bring your 3D models to life.

[VIEW DEMOS](#)



16.75MM

Platinum

MATERIALS

CONTROLS

- Animate View
- Change View
- Hide Dimensions

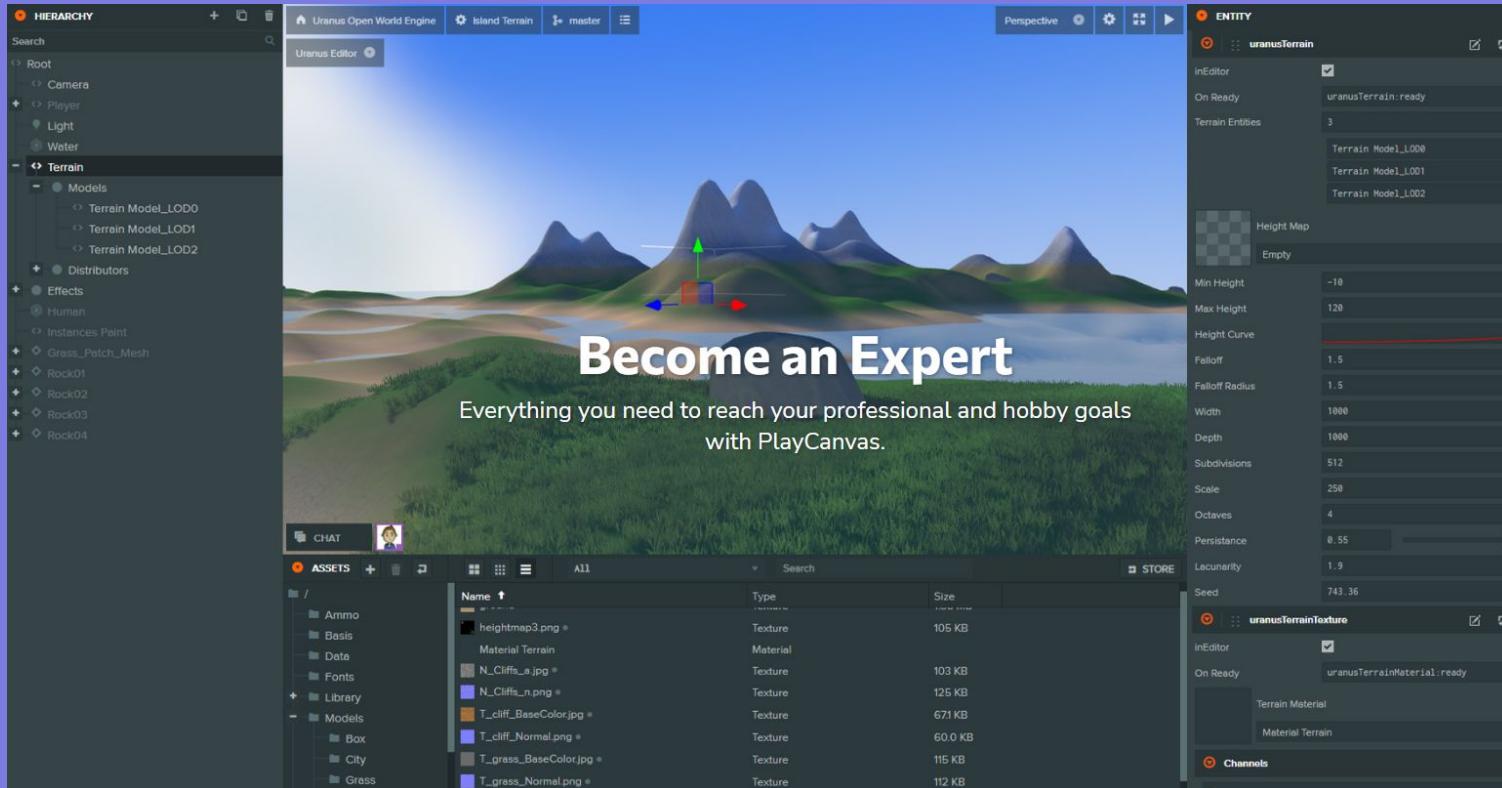
# Uranus Tools Demos

<https://solargames.io/demo>



# PlayCanvas Mentorship

<https://solargames.io/mentorship>



## Become an Expert

Everything you need to reach your professional and hobby goals  
with PlayCanvas.

# THANK YOU!



- Play the demo: <https://solargames.io/demos/solar-stage>
- Find me on Twitter: <https://twitter.com/PlayingInCanvas>
- Find out about Uranus Tools for PlayCanvas: <https://solargames.io/tools>
- Get started with PlayCanvas now: <https://playcanvas.com>
- Join the PlayCanvas Community: <https://forum.playcanvas.com>