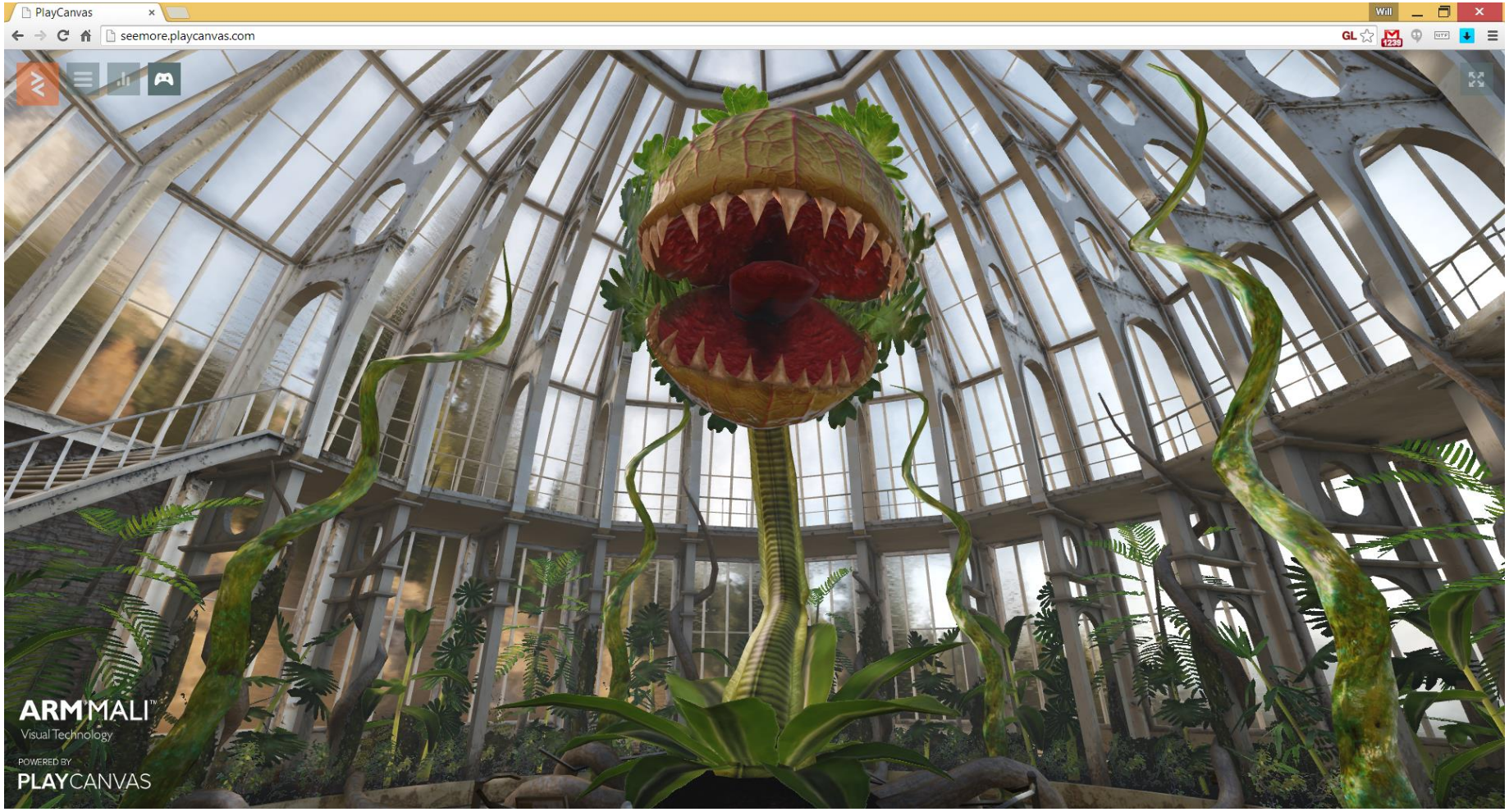
The Making of Seemore WebGL

Will Eastcott, CEO, PlayCanvas

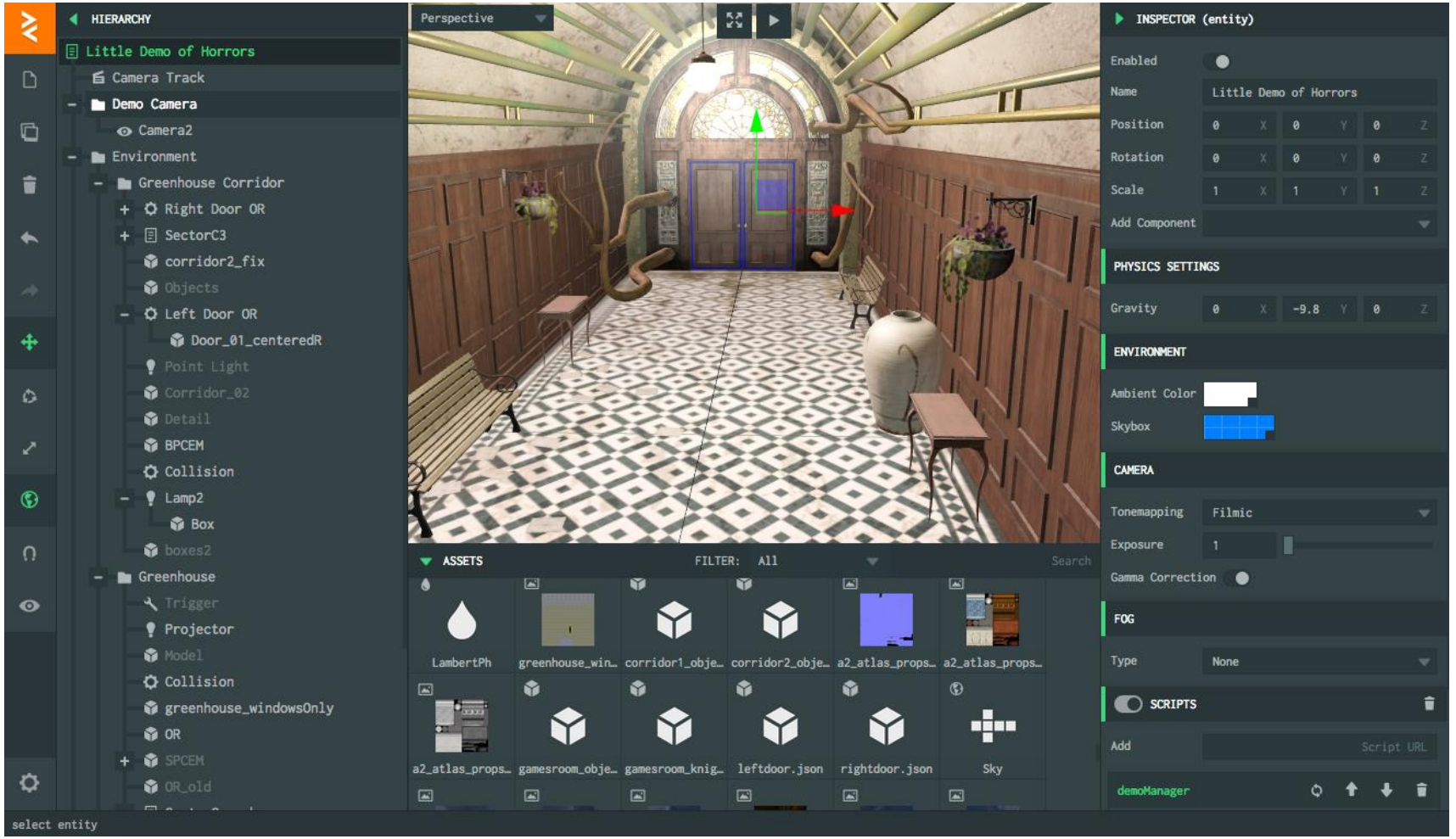
What is Seemore WebGL?

A mobile-first, physically rendered game environment powered by HTML5 and WebGL



PlayCanvas: Powering Seemore WebGL

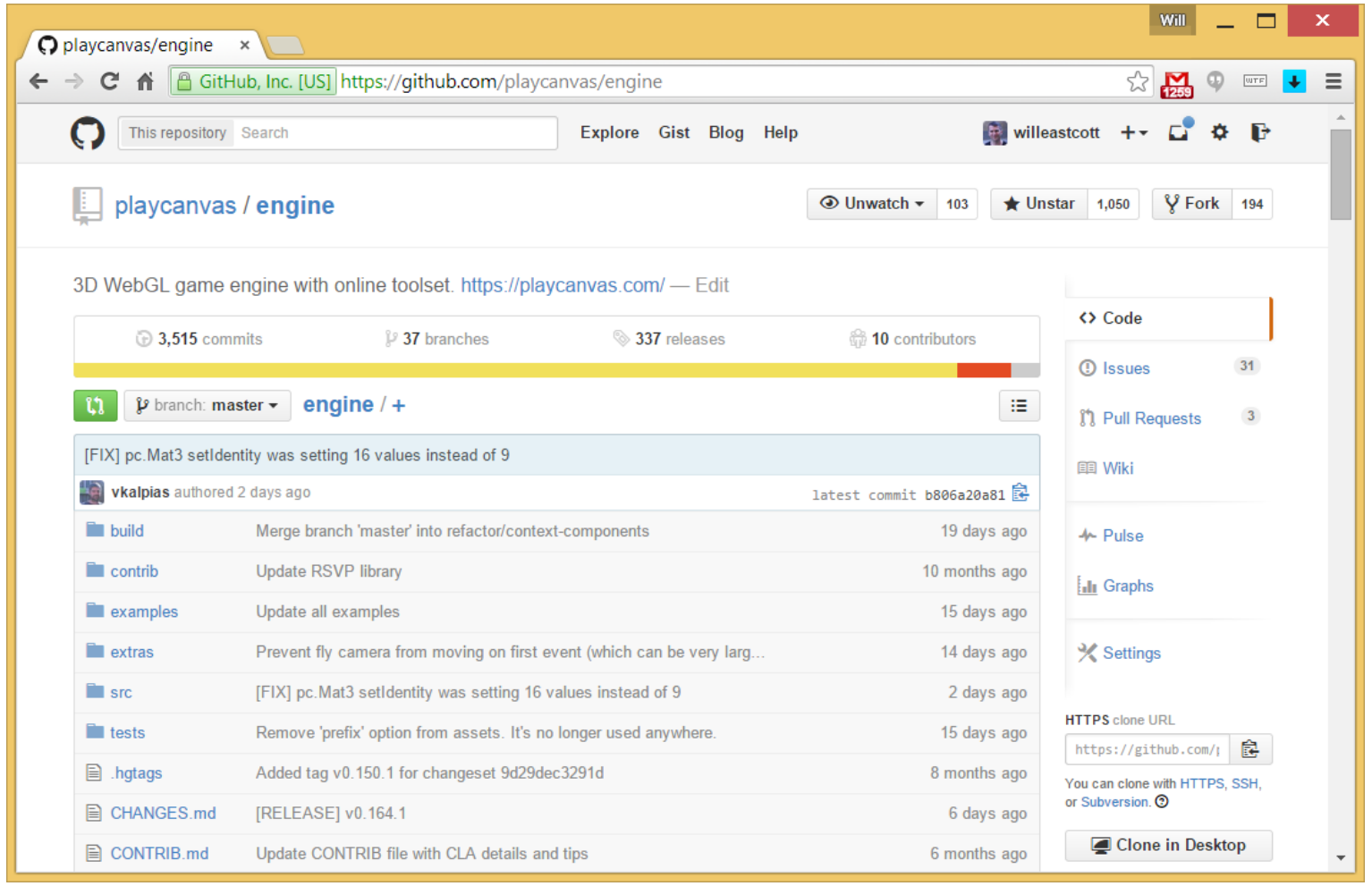
Open-source, WebGL game engine with cloud-based visual tools



ARM®

Did I Mention PlayCanvas is Open Source?

https://github.com/playcanvas/engine



Updating a Classic

The original C++ Seemore demo was significantly upgraded in terms of visuals

ARM®

So How Did We Do It?

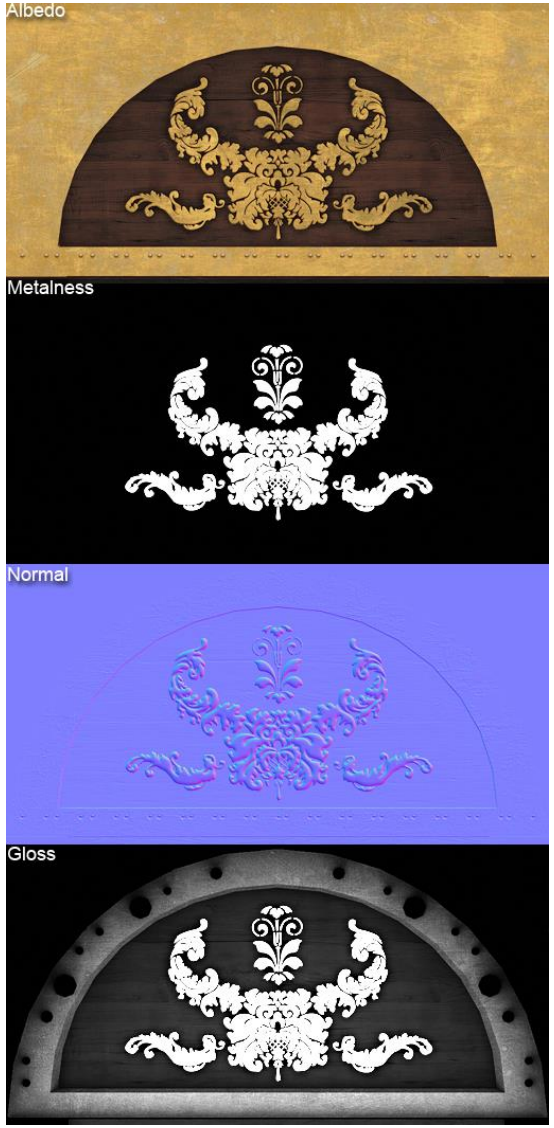
- Physically Based Rendering (PBR)
- Lightmapping
- Ambient occlusion
- Dynamic shadow mapping
- Box projected cube map reflection mapping
- …and a serious amount of optimization

Physically Based Rendering: The Basics

- Energy Conservation
- Meaningful maps
- HDR data
- Tonemapping
- Gamma Correction

Physically Based Rendering: Material Input Data

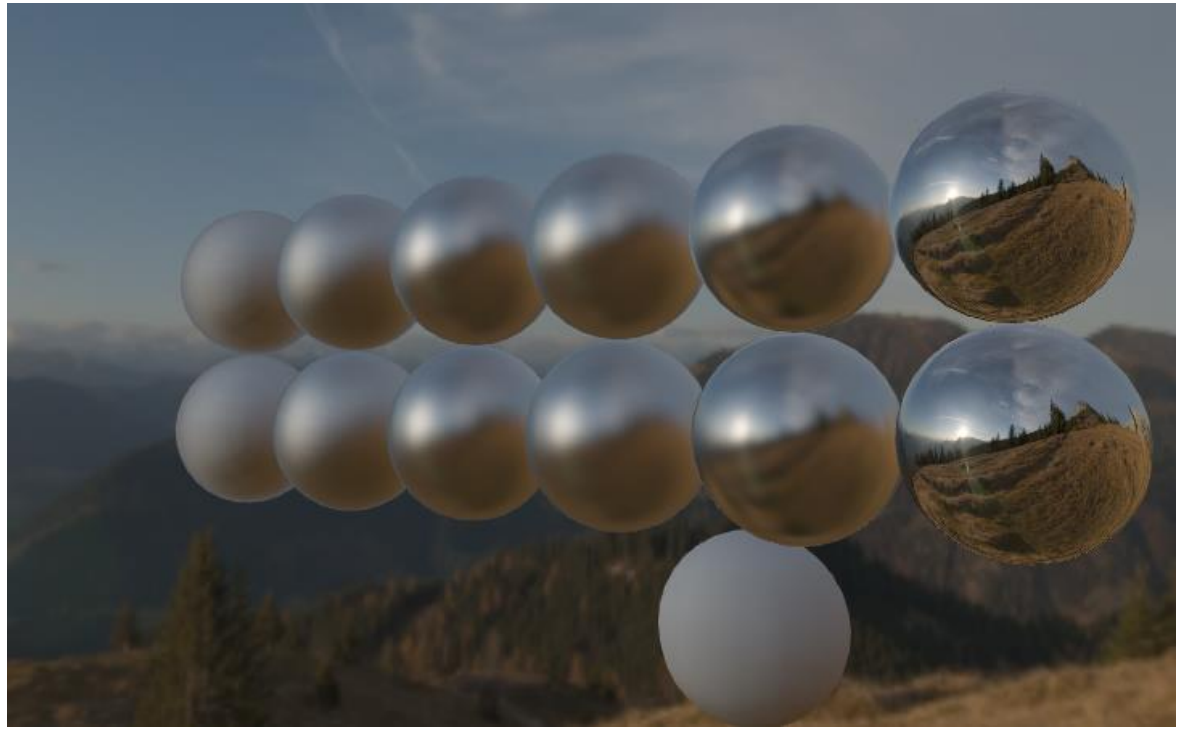
- Material
  - Albedo (color information)
  - Normal (surface information)
  - Gloss (microsurface information)
  - Metalness (optional)
  - Opacity (optional)



ARM®

Physically Based Rendering: Environmental Input Data

- IBL probes – prefiltered, box projected cubemaps
  - Used custom filtering tool
  - PlayCanvas editor users will soon be able to generate these cubemaps in the interface
- Analytical lights (point/directional, energy conserving Blinn-Phong)
- Lightmaps
- Everything must be HDR!
  - Encode HDR to RGBM



Physically Based Rendering: Materials

- Schlick's Fresnel approximation
- Toksvig's factor for specular anti-aliasing

ARM®

Tonemapping

- Important for getting rid of overly bright or dark spots
- Generally simulates how the eye or film perceives colors
- Seemore uses the ‘well known’ Uncharted 2 tonemap. See: http://filmicgames.com/archives/75

ARM®

Lightmapping

Single map for scenes with static light sources

ARM®

Lightmapping

Direct and indirect lightmaps for scenes with slightly moving lights

- Use indirect lightmap as base ambient
- Store directional static shadows in a separate lightmap
  - Can store multiple shadows in a texture's color channels for a slightly moving light
  - Mask real-time diffuse/specular light with shadows
  - Combine with real-time shadows

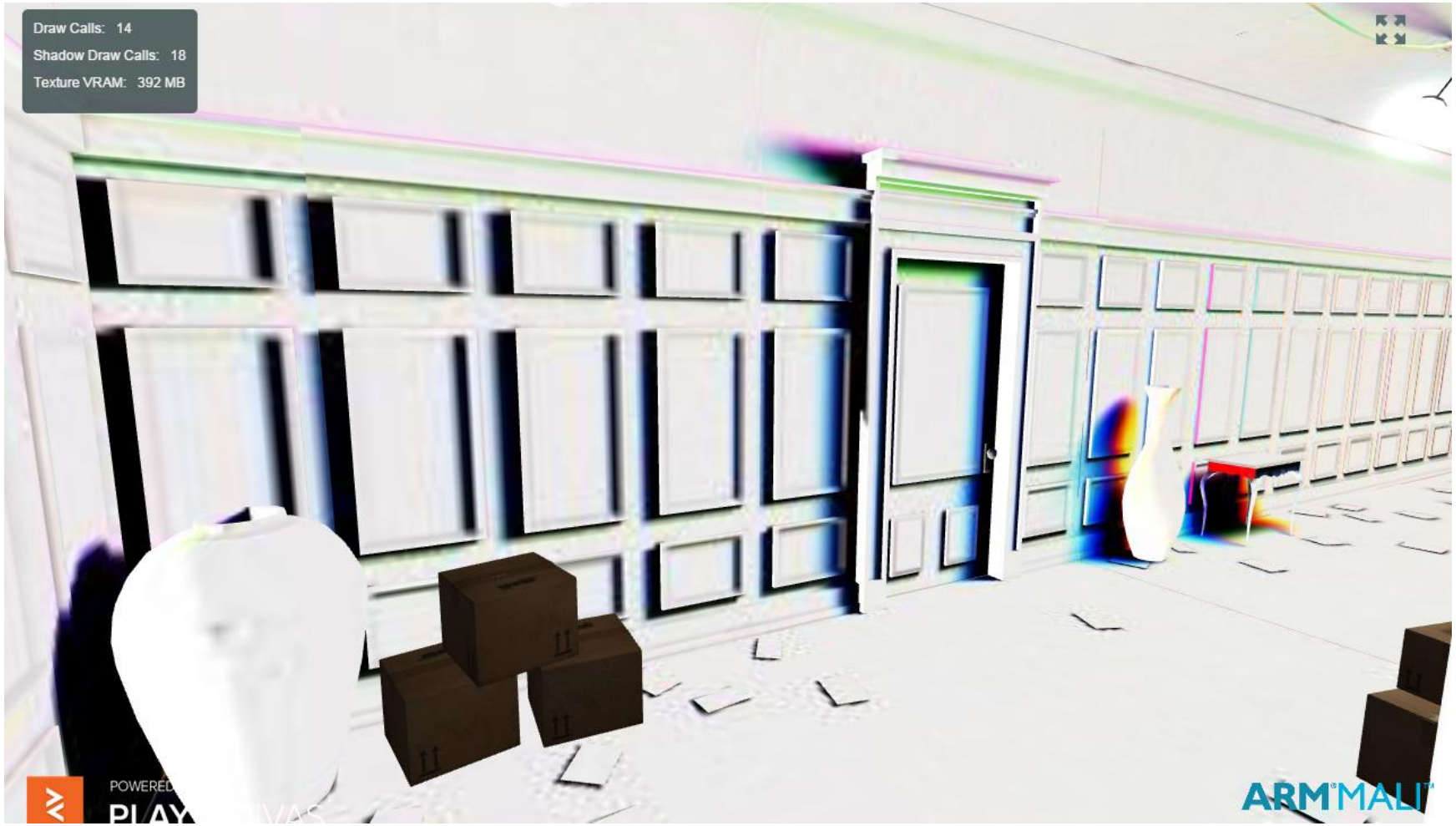
Lightmapping

Indirect lightmap

ARM®

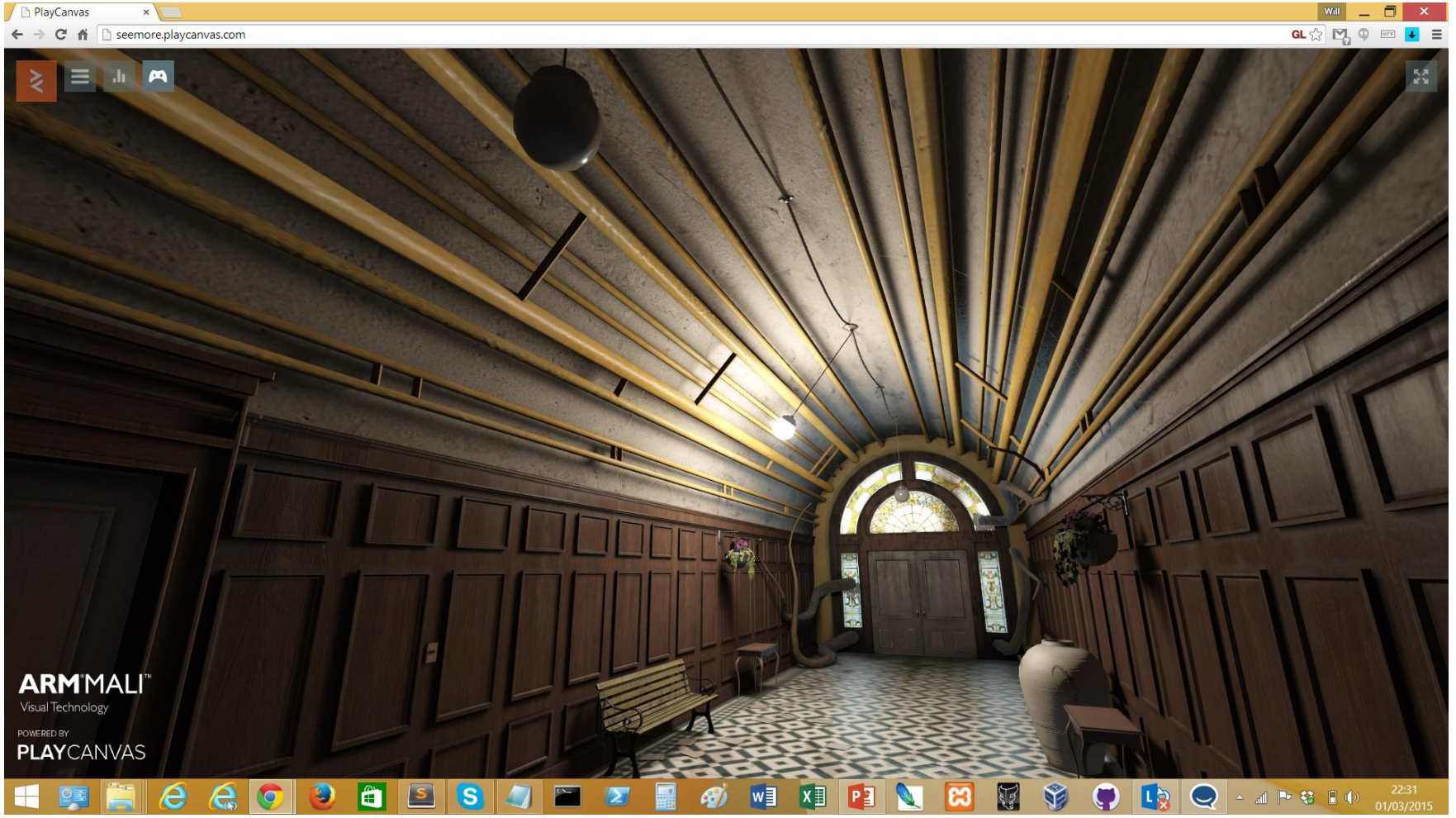
Lightmapping

Direct lightmap



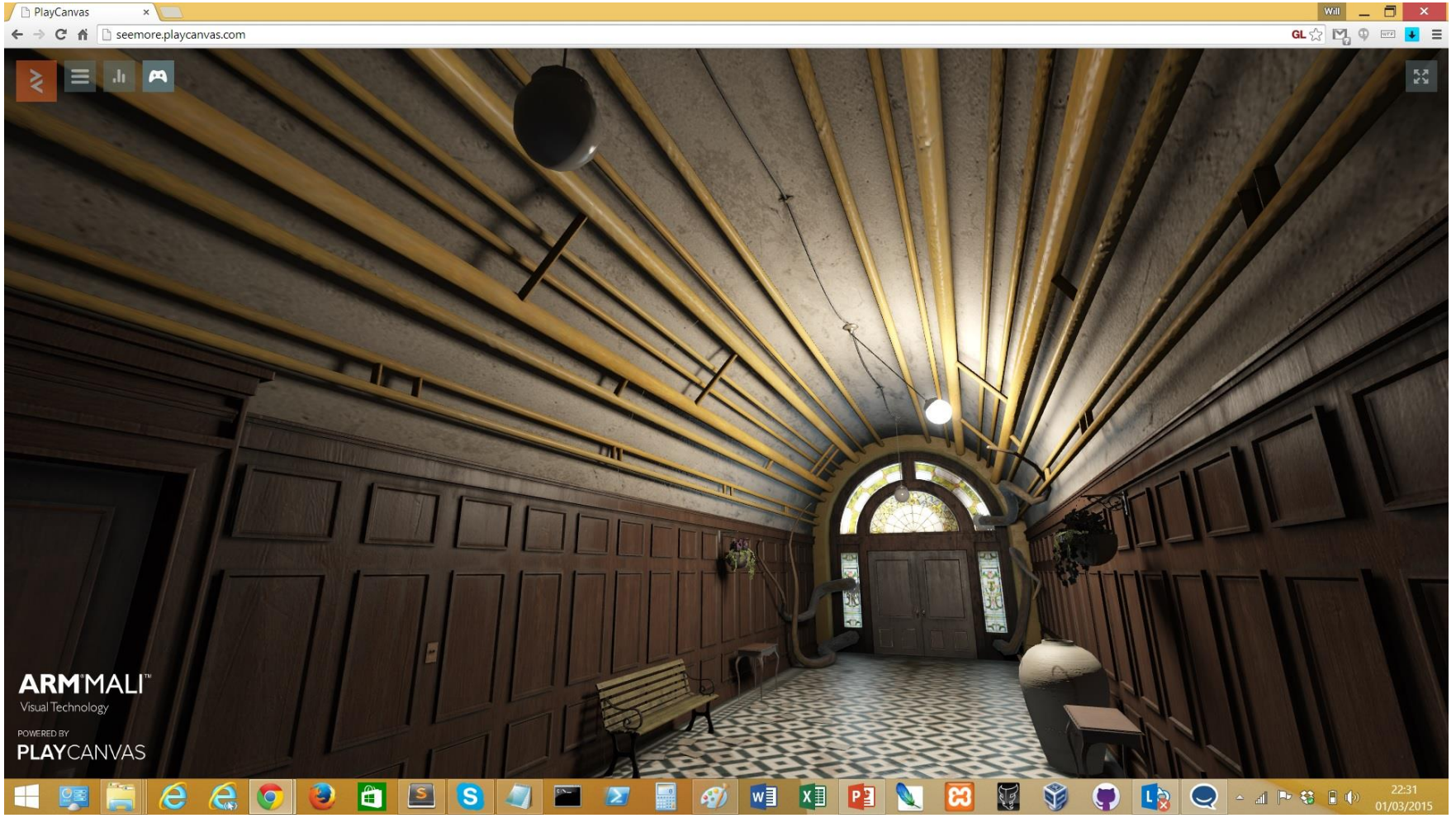
ARM

Dynamic Shadows: Interpolated Lightmaps



ARM

Dynamic Shadows: Interpolated Lightmaps



ARM

Dynamic Shadows: Shadow Mapping

- Real-time shadow maps for dynamic objects
  - Very costly on mobile devices (draw calls, projection, reads)
  - Only update shadow map every 3rd frame if objects are not moving
  - Only use real self-shadowing with multiple taps on dynamic objects
  - Just use bilinear lookup without depth comparison on static receivers
  - Shadow is encoded as RGB (depth) and A (mask for bilinear lookup) in an 8-bit 1024x1024 texture
  - Use Normal Offset bias to reduce self-shadowing artifacts
  - See http://www.dissidentlogic.com/old/#Normal%20Offset%20GDC%20Materials

Dynamic Shadows: Shadow Mapping

ARM®

Ambient Occlusion

- Occluding IBL lighting (especially specular) is vital
- Specular occlusion should ideally be view-dependent but it's too expensive for mobile
- Baked AO maps (tweak attenuation for specular occlusion)
- Can store multiple AO maps in channels (the plant's mouth, for example)
- Last resort: derive some occlusion factor from single/indirect lightmaps
- Procedural AO for special cases (e.g. plant tentacles)

Ambient Occlusion

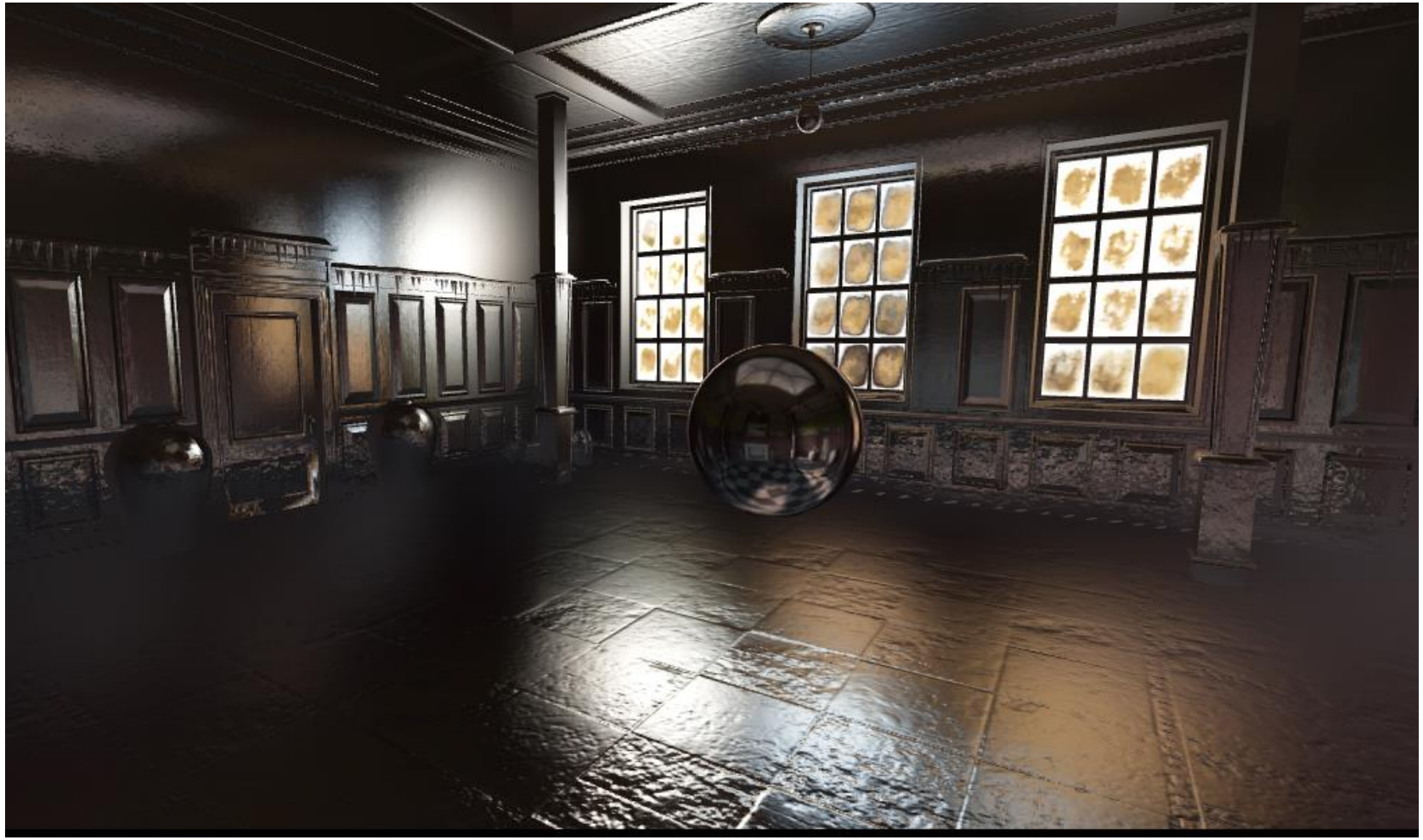
ARM®

Box Projected Cube Map Reflections

- Widely used in games to make cube maps fit in a more correct way with the scene
- Used for both IBL reflections and refractions
- Excellent primer on the topic here: http://www.gamedev.net/topic/568829-box-projected-cubemap-environment-mapping/

ARM®

Box Projected Cube Map Reflections



ARM®

Optimization

- ETC1 Texture Compression (WEBGL_compressed_texture_etc1)
  - Compressing textures is vital for low memory mobile devices
  - WebGL is limited in the formats it exposes but does provide ETC1 (and DXT)
  - Good results for diffuse, specular, gloss and lightmaps
  - Bad results for normal maps
  - Compressed the majority of normal maps anyway
  - Remainder were left uncompressed when the results were particularly bad
- Use lowp in fragment shaders where possible
  - Only precision-critical shaders (shadow mapping) retained highp
- Avoid costly math ops (pow, sqrt)
- Cull draw calls
- Sort opaque draw calls front to back

Additional Tricks: Foliage

- Plants are translucent and pass light to the back side
  - Seemore plants are in a hemispherically lit room
  - Just use –normal.y as a translucency factor
  - Additional analytical per-vertex translucency occlusion for the main plant

ARM®

Additional Tricks: Halos

- Avoid full-screen effects
  - Incurs high fill rate costs
  - You lose hardware antialiasing
  - Although it is possible to implement AA as a post process, again, this is expensive
- Use a cheap alternative
  - Transparent, camera aligned sprite

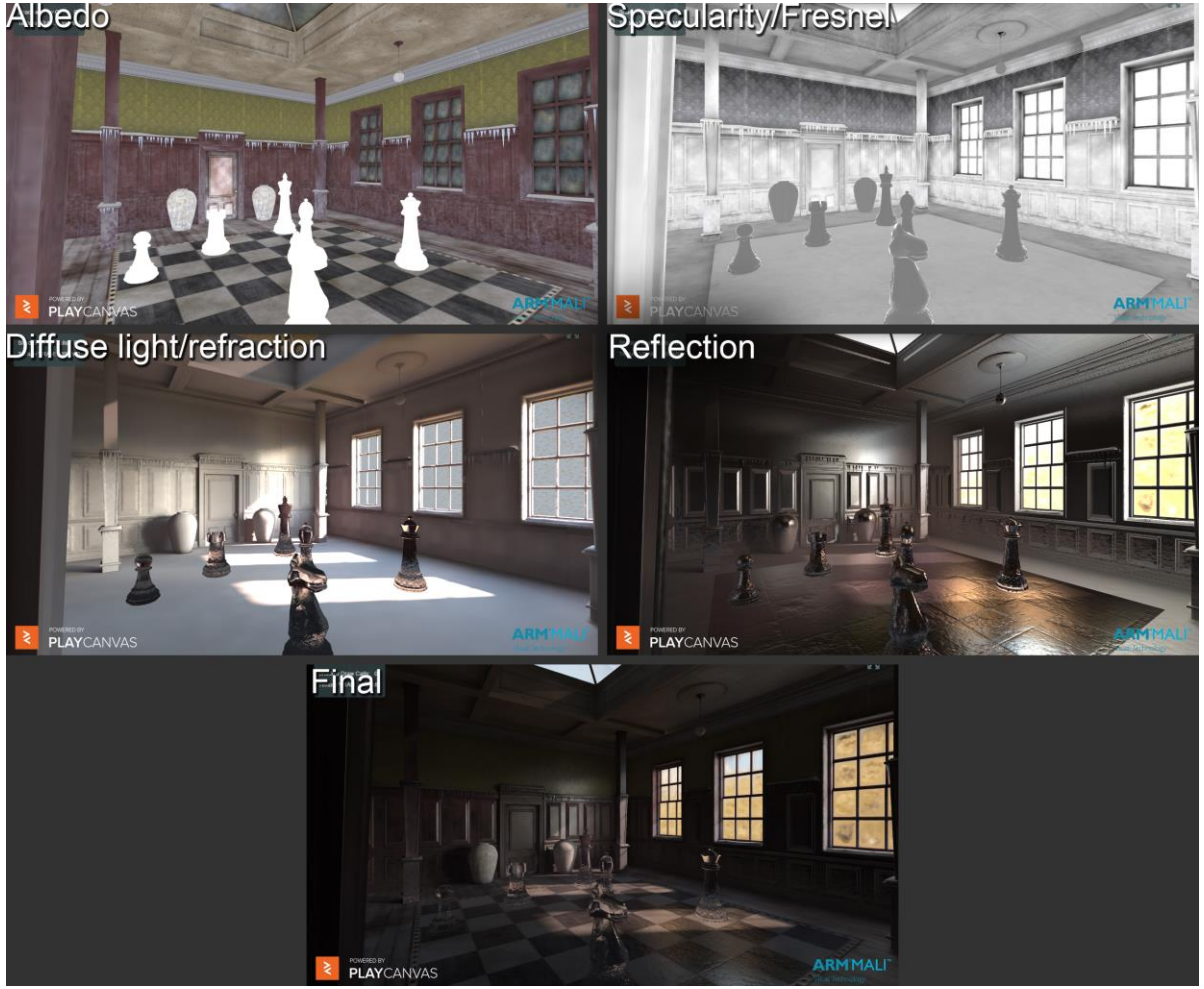
Additional Tricks: Dynamic Exposure

- We avoided an analytical approach
- Exposure controlled through script
- Dependent on location and view vector

Things That Didn't Make The Cut

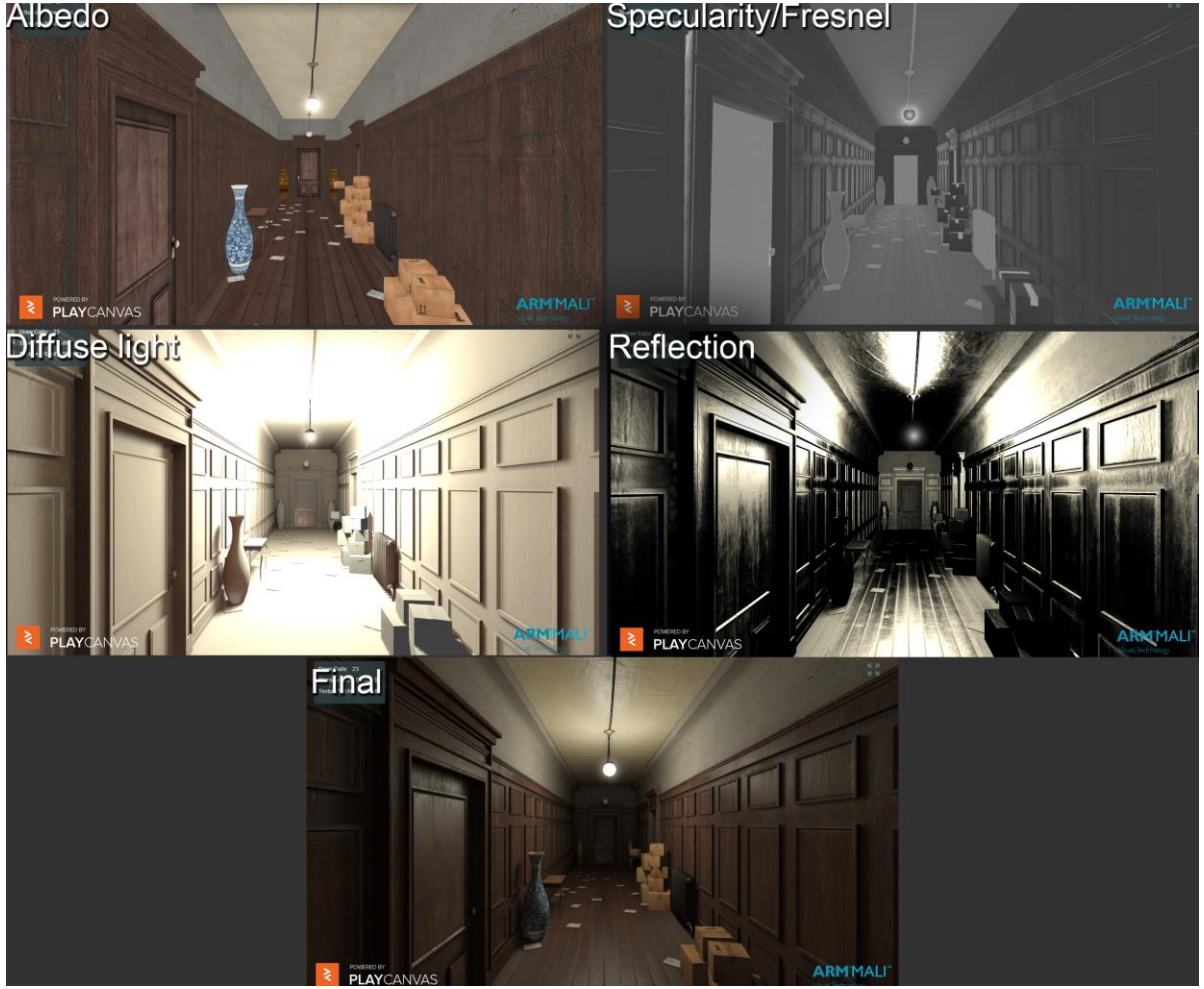
- Instancing (ANGLE_instanced_arrays)
  - Seemore was limited more on the GPU (fragment operations)
  - The demo didn't render enough instances to make a significant difference
- Variance Shadow Mapping
  - Works well on desktop machines but a bug in Chrome's float texture extension was problematic
  - Encoding the shadow map to RGBA8 lost too much precision

Putting It All Together



ARM®

Putting It All Together



ARM®

Some Links…

- The Open Source PlayCanvas project: https://github.com/playcanvas/engine
- The Cloud-Hosted PlayCanvas toolset: https://playcanvas.com
- The Seemore Demo: http://seemore.playcanvas.com
- ARM Mali Developer Center: http://malideveloper.arm.com

tanx.playcanvas.com

ARM®