MDCC 2016
中国移动开发者大会
Mobile Developer Conference China 2016

VR在Web3D中的应用与优化

mdcc.csdn.net

Web3D的现状

不完美的现状，预示着更好的未来

Web3D的能力

WebGL2.0主要特性

- Multi-sampled render-buffers
- 3D textures
- Sampler objects
- Uniform buffer objects
- Sync objects
- Query objects
- Transform feedback

WebGL2.0主要特性

- Multi-sampled render-buffers
- 3D textures
- Sampler objects
- Uniform buffer objects
- Sync objects
- Query objects
- Transform feedback

WebGL2.0的可用扩展

- Multiple render targets: WEBGL_draw_buffers
- Instancing: ANGLE_instanced_arrays
- Vertex array objects: OES_vertex_array_object
- Fragment depth: EXT_frag_depth

WebGL2.0中可以实现的特性

- Deferred shading
- Occlusion culling
- Light propagation volumes
- Voxel Cone tracing GI
- Image based lighting
- HDR

VR设备概览

VR设备千百种，总有一款适合您

移动VR设备

- 手机应用
- 分屏显示
- 手机作为处理器与显示器
- 内部陀螺仪检测旋转和朝向
- 沉浸体验稍弱
- 对Web支持度较高

桌面VR设备

- 标准PC桌面应用
- 头载设备作为独立显示单元
- 内、外置的传感器检测头部（全身）的朝向和位置
- 强大的沉浸体验
- 对Web支持度最高，体验也最好

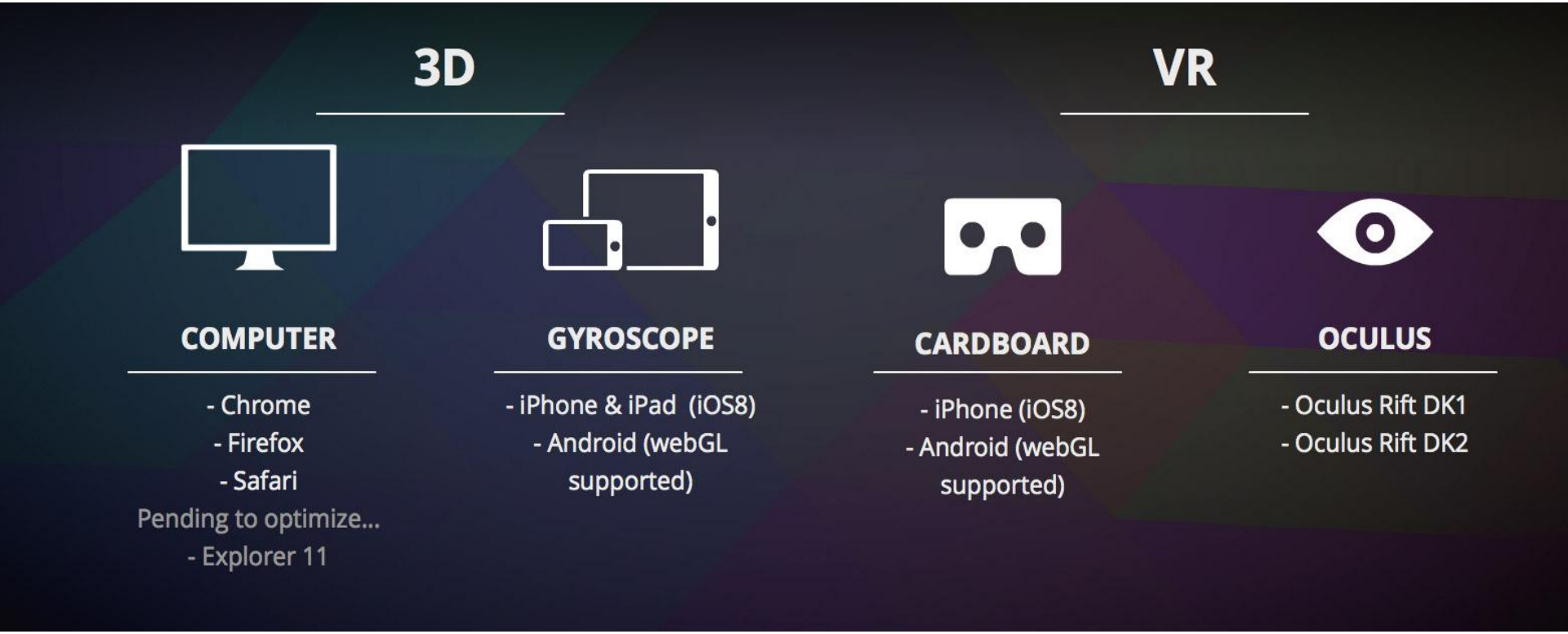
独立VR设备

- 一体化设备
- 处理器和显示器均包含
- 一般是基于移动系统（安卓等）
- 体验和性能处于中间

VR在Web中的应用

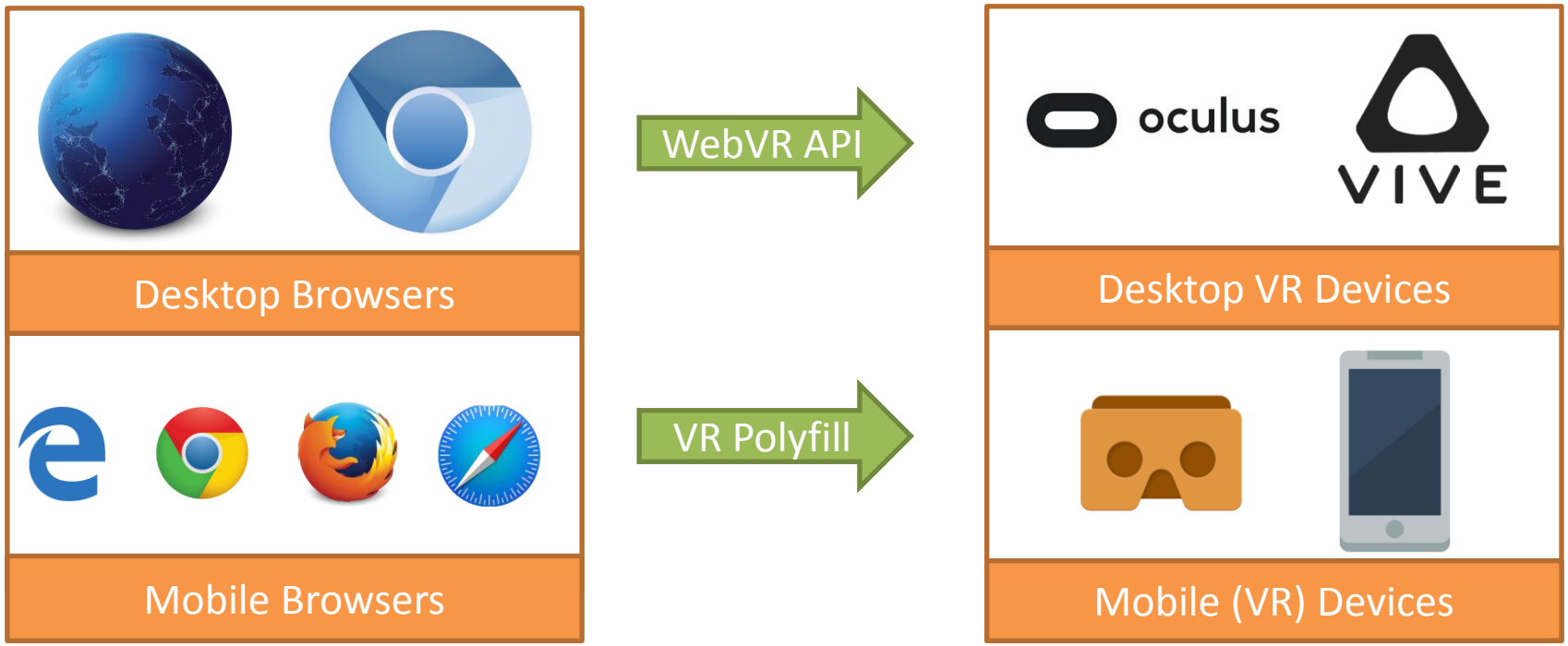
相得益彰，应用更有价值

Web端接入VR



Feature	Safari iOS	Android Browser	Samsung Internet	Google Chrome	Amazon Silk	BlackBerry Browser			Nokia Browser		Internet Explorer		Opera Mobile	Opera mini	Firefox	
Platform	iPhone, iPad	Phones & Tablet	Android devices	Android 4.0+	Kindle Fire	Phones	BB10	Tablet	Nokia X	Symbian	Windows Phone	Windows 8	Android & Symbian	Java,iOS Android	Android, MeeGo*	Firefox OS
WebGL Khronos Group API 3D Canvas for the web	✓ 8.0b+	✓ Specific device	✓	✓ 30+			✓	✓ 2.0+			✓ 11+	✓ 11+	✓ 12+ (android)		✓	✓

Web端接入VR

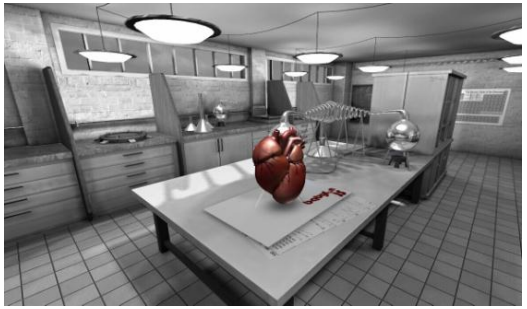

Desktop Browsers
Mobile Browsers
WebVR API
VR Polyfill
oculus
VIVE
Desktop VR Devices
Mobile (VR) Devices

Web端的VR的优势

- 天生轻量化
- 更具移动性
- 商业化属性更强
- 用户体验门槛更低
- Web端比App端更有优势

Web&VR的应用

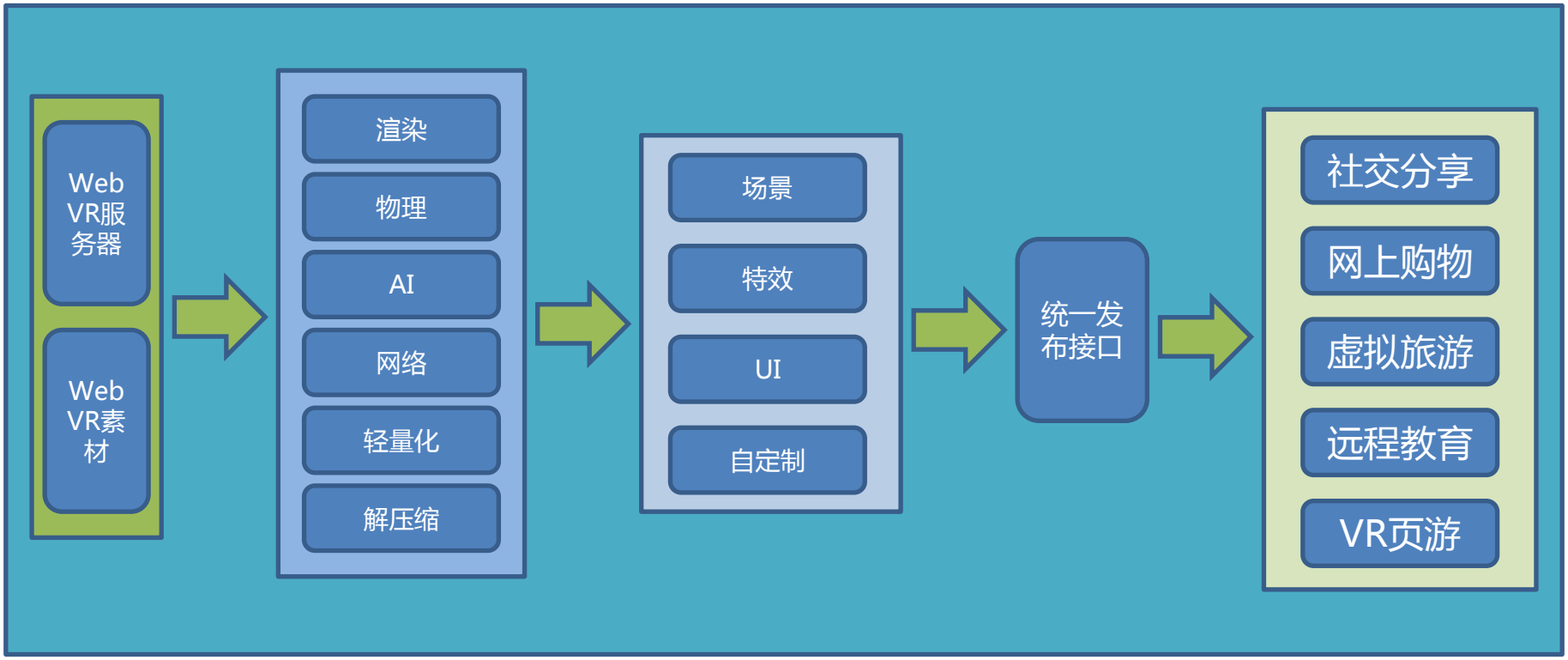
- 网上购物
- 虚拟旅游
- 在线展示
- 3D打印
- 三维仿真
- ...



Web + VR的生态链

有人的地方就有江湖，有江湖的地主就有生态

WebVR生态链示意图



生态链模块

- 后端：强大的服务器
- 中间：Web&VR的引擎
- 前端：面向中间用户的交互前端
- 接口：面向产品化的统一发布接口
- 商业：商业或产品化面向用户

WebVR的开发与工具

人之所以进化，在于学会了使用工具

利用WebVR js库

- WebVR：一个实验性的jsVR库，主要用来实现浏览器与VR硬件之间的交互
- 可以跟踪VR的头显设备
- 传送渲染数据给到VR设备
- 允许应用头显的空间属性
- 浏览器有Firefox nightly和Chromium的内核

利用WebVR js库

- VRDisplay:　VR设备的基础实现，可以获取包括位置、朝向、姿态、远近平面以及其它属性
- VRPose：获取特定时刻下的头显VR设备的姿态
- VRStageParameters：描述空间位置的关键参数
- 其它扩展：可以得到交互UI与控制输入设备
- 更多的接口：https://mozvr.com/webvr-spec/

利用WebVR js库

- 有专门的实现接口给Three.js和Babylon.js
- VRControls:　将头显的空间变换控制到对应的场景摄像机中
- VREffect：在摄像机下控制针对VR的分屏显示
- WebVRManager：切换VR的模式及整体WebVR属性与状态的管理

- WebVR Polyfill：针对没有原生WebVr支持的间接实现

使用传统引擎-Unity

- 在Unity中开发游戏内容
- 添加Jump Gaming WebVR插件
- 发布基于WebGL的HTML5游戏内容
- 即具有对于浏览器的VR支持

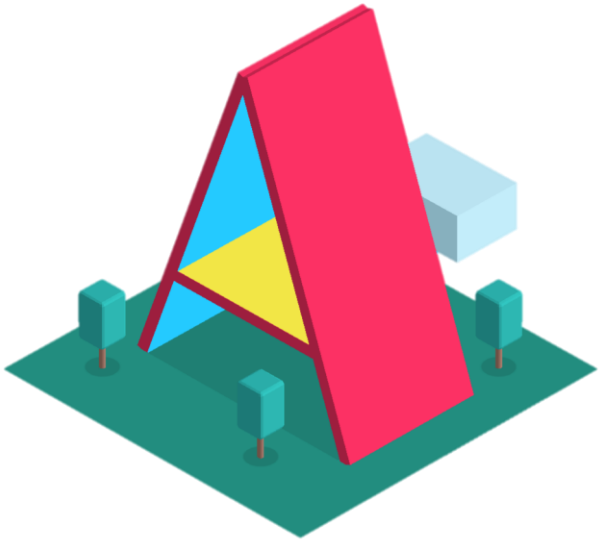
使用传统引擎-Unreal

- Unreal4.7之后的版本已经有对直接发布到html5的支持
- 但对发布到WebVR的支持目前还没有
- 可以使用UE4+Emscripten的方法来实现
- Emscripten最新版本已经添加对于WebVR的支持

第三方工具-AFrame

- 标记式的语言
- 组件-属性的模型
- 支持2D和3D对象：包括几何何，球面贴图、视频、模型等。

```
<script src="aframe.min.js"></script>
<a-scene>
 // VR code!
</a-scene>
```



第三方工具-VizorCreate

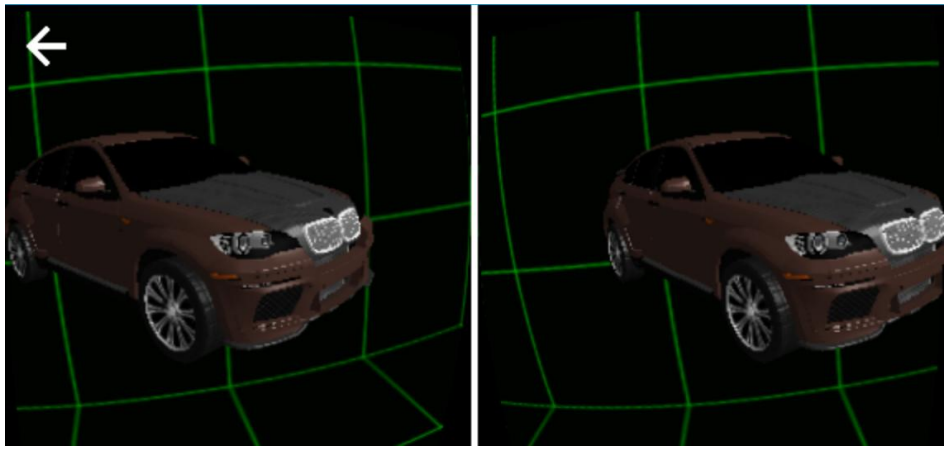
- 基于结点的可视化编辑模式
- 多用户编辑以及实时可视化

Hello, WebVR

你好，外勃维啊的新世界

体验HelloWebVR

- 手机扫描二维码+cardboard即可体验
- http://www.shxt3d.com/webvr/index.html



实现说明-WebGL的框架

- 基于Three.js来实现
- 可以快速搭建Web3D的模型，具有较为丰富的插件
- 其它有较多成熟WebGL框架或引擎可选：Babylon.js, PlayCanvas, Minko等。

PLAYCANVAS

实现说明-基本要素

- 首先是建立scene、renderer、camera三个基本要素

```
// Setup three.js WebGL renderer. Note: Antialiasing is a big performance hit.
// Only enable it if you actually need to.
var renderer = new THREE.WebGLRenderer({antialias: true});
renderer.setPixelRatio(window.devicePixelRatio);

// Append the canvas element created by the renderer to document body element.
document.body.appendChild(renderer.domElement);

// Create a three.js scene.
var scene = new THREE.Scene();

// Create a three.js camera.
var camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 10000);
```

实现说明-初始化VR

- 对WebVR库中的VRControls、VREffects、WebVRManager进行调用并初始化

```
// Apply VR headset positional data to camera.
var controls = new THREE.VRControls(camera);
controls.standing = true;

// Apply VR stereo rendering to renderer.
var effect = new THREE.VREffect(renderer);
effect.setSize(window.innerWidth, window.innerHeight);

// Create a VR manager helper to enter and exit VR mode.
var params = {
  hideButton: false, // Default: false.
  isUndistorted: false // Default: false.
};
var manager = new WebVRManager(renderer, effect, params);
```

实现说明-绘制并渲染

- 加载指定的对象到场景中，并进行绘制

```
var loader = new THREE.ObjectLoader();
loader.load('./Classroom/scene.json', function (obj){
    mesh = obj;
    // Add cube mesh to your three.js scene
    scene.add(mesh);

    mesh.traverse(function (node) {
        if (node instanceof THREE.Mesh) {
            node.geometry.computeVertexNormals();
        }
    });
    // Scale the object
    mesh.scale.x = 0.2;
    mesh.scale.y = 0.2;
    mesh.scale.z = 0.2;

    targetMesh = mesh;

    // Position target mesh to be right in front of you.
    targetMesh.position.set(0, controls.userHeight * 0.8, -1);
});
```

示例代码下载

- 示例代码可于github上下载
- https://github.com/bugrunnerzhang/hellowebvr.git

优化与经验

关于优化，我们可以做的更多

一些需要注意的点-WebGL

- 确保WebGL的运行不能产生任何错误
- 不要再使用#ifdef GL_ES的宏
- 使用mediump代替highp
- 在初始化前查询硬件的支持能力
- 在顶点着色器中使用贴图时需要先确认是否支持
- 在使用webgl扩展前必须先进行能力查询

一些需要注意的点-WebGL

- 所有需要同步CPU和GPU的操作都会比较慢，在主渲染循环中慎用。
- 合并drawcall，使用尽可能大的drawArray和drawElements
- 减少渲染状态的切换，比如对于图片使用atlas
- 使用mipmapping
- 尽可能地将操作在顶点着色器中进行
- 要确保顶点属性0通道是开启的

其它的优化策略

- 使用Deferred shading来实现多光源的支持（需要先查询扩展能力）
- 使用实例化来提高某些重复物件的渲染效率（需要先查询扩展能力）
- 使用oculusion culling来进行渲染前的裁剪（需要先查询扩展能力）
- 使用emscripten来转化某些库到js中，可能性能会比js库要高
- 使用SIMD.js及其它 的一些性能库

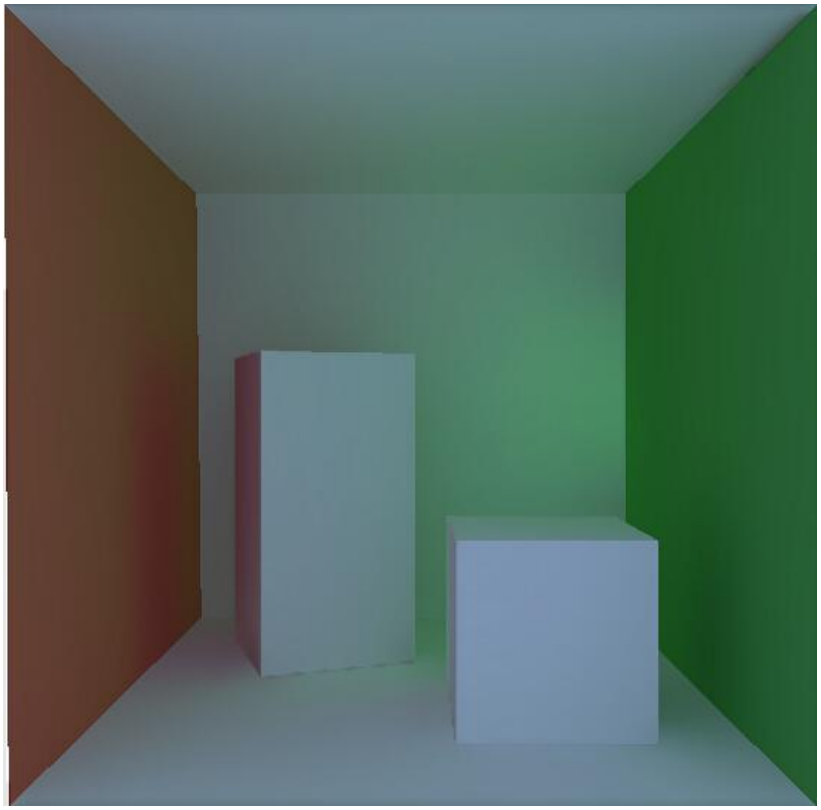
我们所做的...

我们的征途是星辰大海

基于云端的WebVR

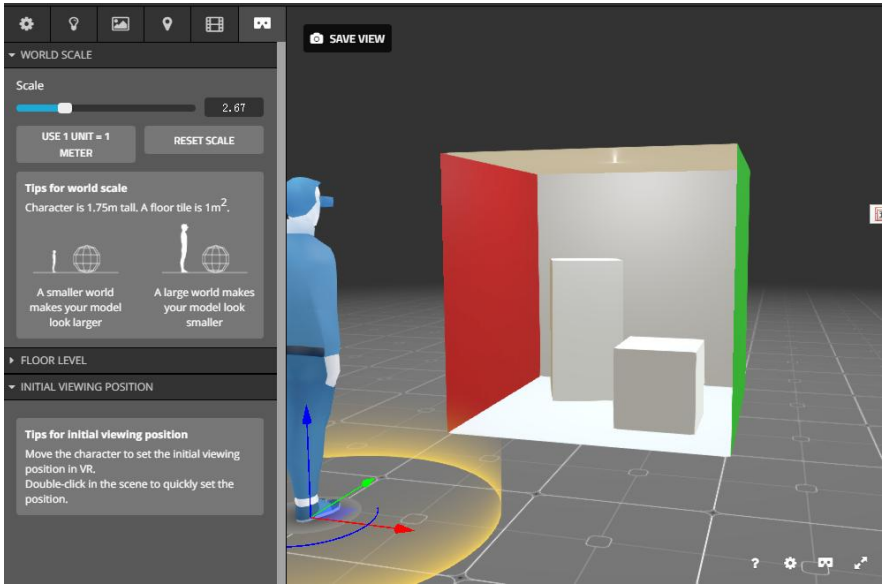
•云端渲染，分流前端的渲染压力，提升Web端的渲染效果，达到实时动态GI

•云端计算，减轻Web端的渲染与计算压力。合理地使用前端与网络资源，通过减少前端的计算来流畅VR体验



轻量化的WebVR内容编辑器

- 所见即所得的编辑方式，普通用户的较低使用门槛
- 与后端及发布端的无缝结合
- 渐进与流式化模型、贴图的压缩与传输



THANKS