MorphoMR® Plugin for HTML5 文档

版本：0.0.0 alpha

发布日期：2018 年 7 月 5 日

目录

MorphoMR® Plugin for HTML5

MorphoMR® Plugin for HTML5（MRH5）是为幻视互动智能眼镜设备提供的混合现实网页版应用开发框架和工具集合，同时也是一套支持安卓平台移动端设备的混合现实网页应用开发插件。支持视频渲染、3D 渲染、复杂互动等功能。

当前 MorphoMR® Plugin for HTML5 0.0.0 alpha 版本可供免费试用。

开发工具

本文将向您介绍部分 MorphoMR® Plugin for HTML5（MRH5）的推荐开发工具和所支持的运行环境。

开发工具推荐：

- JetBrains WebStorm
- Visual Studio Code
- Visual Studio 2017

运行工具：

- Chrome DevTools
- FireFox Developer Tools

配置环境：

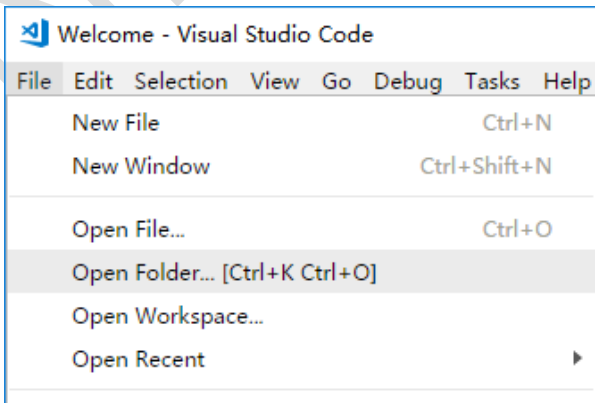
- Android WeChat 6.6.7 中文版
- 已配置 SSL 证书的 https 域名

工程导入

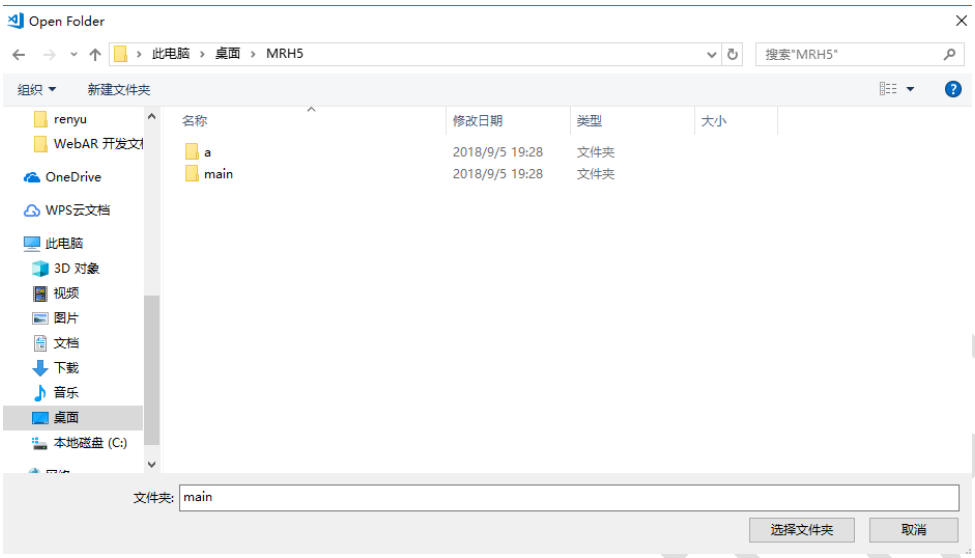
本文将向您介绍如何通过 Visual Studio Code 导入 MRH5 项目。

(一) 启动 Visual Studio Code

选择 File > Open Folder：



导入下载好的 MRH5 示例工程文件：



(二) 初始化

MRH5 项目里使用的 JS 库可通过两种途径加载：

1. 云端加载：通过连接从云端获取 JS 文件，例如：

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/webrtc-adapter/6.2.1/adapter.min.js"></script>
```

2. 本地加载：从本地加载已下载好的 JS 文件，例如：

```
<script src="js/adapter.min.js"></script>
```

(三) 文件管理

脚本以及资源文件框架可自定义，无特定要求。

CSS

index.css - 控制工程整体页面排布。

JS

adapter.min.js - 适配 WebRTC 的规范更改和前缀差异。

createjs.min.js - 实现预加载的功能。

jquery-3.3.1.min.js -提供简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互。

Three.min.js -提供多种 3D 显示功能的 WebGL 库。

3D 模型使用的代码库:

OrbitControls.js - 轨道加载控制器，可实现场景和内容的点击缩放功能。

OBJLoader.js - 将.OBJ 格式 3D 模型导入 AR-3D 场景。

MTLLoader.js - 将.MTL 格式 3D 模型导入 AR-3D 场景。

FBXLoader.js - 将.FBX 格式 3D 模型导入 AR-3D 场景。

Inflate.min.js - FBXLoader.js 需要的脚本。

index.js -控制工程整体页面。

img - 网站 UI 的素材。

resources - 可设置和替换的素材。

assets – 待使用模型（.FBX 格式）。

index.html - 工程主页。

开发手册

本文档旨在介绍如何开发一个可运行于 web 前端的 MorphoMR® MRH5 项目，实现以下功能：

(一) 预加载资源

(二) 调用摄像机

(三) 设置 AR-3D 场景

(四) 加载 AR 视频

(五) 加载 AR 模型

(六) IMU 控制

（一）预加载资源

MRH5 通过 create.js 实现预先加载资源。create.js 实现了预先加载图片，视频和音频及加载进度更新的功能。

1. 在 html 代码中引入 create.js 库，示例如下：

```
<script src="../a/libs/createjs.min.js"></script>
```

2. 在 index.js 初始化 create.js，示例如下：

```
this.preload = new createjs.LoadQueue(!1),
```

3. 初始化 preloader 函数，示例如下：

```
key: "preloader",value: function () {
var progressBar = $("#progressBar")

this.preload.installPlugin(createjs.Sound),

this.preload.on("complete", function () {
    // 加载完成后的代码
}, this),

this.preload.on("progress", function () { // 更新加载进度
```

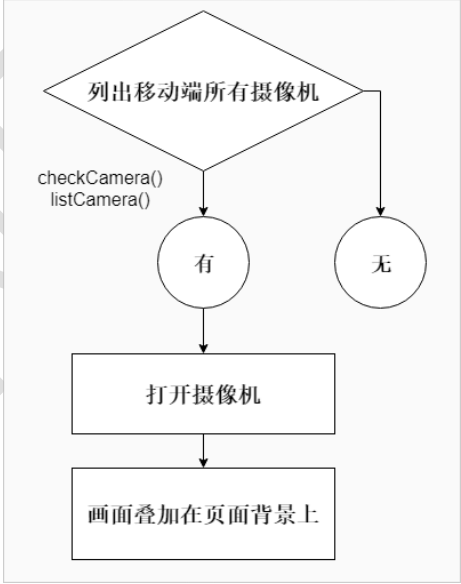
```
    var e = Math.floor(100 * this.preload.progress);
    $("div", progressBar).css("width", ((e / 100) + "%");

    document.getElementById("openLoadingPercent").innerHTML =
((e / 100) + "%";
}, this),

this.preload.loadManifest([ // 待预加载的资源（图片，视频，音频）
    {src: "../a/media/btn_ar.png"},
    {src: "../a/media/openanim-landscape2.jpg"},
    {src: "resources/1.mp4"}
])
}
```

（二）调用手机 / WebCam 摄像机

1. MRH5 实现相机调用可直接通过 index.js 实现，示例如下：




```
//列出移动端所有摄像机
navigator.mediaDevices.enumerateDevices().then(function(devices){
    console.log(devices),
        devices.find(function(device){
            if ("videoinput" === device.kind) {
                var videoDevices = {};
```

```
                videoDevices.name = device.label || "camera",
                    videoDevices.deviceId = device.deviceId,
                    devices.push(videoDevices)
            }
        }),
        r.devices.length === 0 ? n("获取摄像头失败") :
(canvasElement = document.createElement("canvas"),
            canvasContext = canvasElement.getContext("2d"),
            console.log(devices),
            resolve(devices))
}).catch(function (error) {
    reject(error)
});

var deviceSettings = {
     audio: !1,
     video: {
             deviceId: {
                     exact: a
             }
     }
};

//打开摄像机
let video = $("#video-box")[0],
navigator.mediaDevices.getUserMedia(deviceSettings).then(function
(e) {
     video.srcObject = e,
     video.style.display = "block",
     video.play(),
     resolve(!0)
}).catch(function (error) {
     n(error)
});
```

2. 在 HTML 的 <body> 分区里加入以下代码

```
// 把摄像机画面叠加在页面背景上
<video id="video-box" playsinline=""></video>
```

（三）设置 AR-3D 场景

1. 在 index.htmL 中引入 Three.js，示例如下：

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/94/three.min
.js"></script>
```

或者

```
<script src="js/three.min.js"></script>
```

2. 在 index.js 上初始化 ThreeJs 元素，示例如下：

```
// 初始化 ThreeJs 基本场景
var scene = new THREE.Scene()

// Camera 相机
var camera = new THREE.PerspectiveCamera( // 透视相机 - 近大远小
    45, // 可视角度
    this.canvas.offsetWidth / this.canvas.offsetHeight, // 长宽比
例
    0.1, // 近深度剪切面
    1000 // 远深度剪切面
)

// CanvasRenderer 渲染器
var renderer = new THREE.WebGLRenderer({
    canvas: canvas, // 若未设置，ThreeJs 会在 HTML 上自动生成 Canvas
    antialias: !1,
    alpha: !0,
    premultipliedAlpha: !0
}),

renderer.setClearColor(0, 0), // 设置背景色
renderer.setPixelRatio(window.devicePixelRatio), // 渲染区域大小

//可根据预先设置好的 Canvas 或者根据窗口尺寸设置尺寸
window.innerWidth, window.innerHeight
renderer.setSize(canvas.offsetWidth, canvas.offsetHeight, !1),
// renderer.setSize(canvas.offsetWidth, canvas.offsetHeight, !1),
```

（四）加载 AR 视频

1. 在 index.html 初始化<video>元素

```
<div class="videobox" ontouchmove="return false;" style="display:
none;z-index: -1">
    <video id="myvideo" x5-video-player-type="h5"
style="display:none" height="512"
                    width="256" loop x5-playsinline=""
playsinline=""
                    webkit-playsinline="" poster=""
preload="auto"></video>
</div>
```

2. 在 index.js 初始化 ThreeJs 视频平面

```
// 指向已初始化的 <video> 元素
this.myvideo = $("#myvideo");

// 绑定 AR 视频素材
$("#myvideo").html('<source src="resources/1.mp4"/>'),

// 初始化 ThreeJs VideoTexture，绑着该视频，然后继续通过
MeshBasicMaterial 绑定在 ThreeJs 平面（plane）上：
var InitVideoPlane = function () {
    function {
    this.video = (0, a.default)("#myvideo")[0],
    this.texture = new s.VideoTexture(
            this.video,
            s.UVMapping,
            s.ClampToEdgeWrapping,
            s.ClampToEdgeWrapping,
            s.LinearFilter,
            s.LinearFilter,
            s.RGBFormat,
            s.UnsignedByteType
    ),
    this.material = new s.MeshBasicMaterial({
            map: this.texture,
            overdraw: .5
    }),
```

```
    this.plane = new s.PlaneGeometry(this.width, this.height, 4,
4),
    this.mesh = new s.Mesh(this.plane, this.material),
    this.mesh.scale.x = this.scale
    }
    return i(n, [{
            key: "show",
            value: function (e, t) {
                    var n = this.height * (.5 *
window.innerHeight - (e + .5 * t)) / t,
                    r = -this.height * window.innerHeight / (2 *
t *
                                        Math.tan(s.Math.degToRad(.5
* this.camera.fov)));
                    this.mesh.position.set(0, n, r);
    }
}])
}

// 当预先加载完成后，初始化 AR-3D 场景
this.preload.on("complete", function () {
getVideo().show(e, t);
}, this),

// 设定全屏播放视频
this.myvideo[0].play()
```

（五）加载 AR 模型

在 index.js 中调用 FBXLoader.js 实现预先加载资源

```
{
key: "loadFBXmodel",
value: function (modelName, x, y, z) {
    let ke = this;

    // 初始化
    this.fbxLoader = new THREE.FBXLoader();
    this.fbxLoader.load(
            'assets/fbx/' + modelName + '.fbx',
            function ( object ) {
                    ke.modelObject = object;

                    // 绑定模型动画，初始化 AnimationMixer，指向模型
                    object.mixer = new
THREE.AnimationMixer( object );
                    ke.mixers.push( object.mixer );
                    var action =
object.mixer.clipAction( object.animations[ 0 ] );
                    action.play();

                    ke.modelObject.position.set(x, y, z); // 设置
模型位置
                    ke.scene.add( object );
            },
            function (e) {
                    // 显示模型加载进度
                    var f = Math.floor(e.loaded / e.total *
100);
                    $("div", $("#progress")).css("width", f +
"%");

    document.getElementById("loadingPercent").innerHTML = f +
"%";

                    console.log( f + '% loaded' );
                    if (e.loaded === e.total) {
                            console.log("content all loaded");
                            // 继续进行代码
                    }
            },
```

```
            function (error) {
                    console.log("an error happened: " + error);
            }
    )
}
}
```

（六）IMU 控制

在 index.js 调用 DeviceOrientationControls 类库实现 IMU 控制。

```
{
key: "set",
value: function () {
let ke = this;

// 绑定 3D 场景里的相机到模型上
this.camera.updateMatrixWorld(!0);
this.camera.localToWorld(ke.modelObject.position);
this.camera.getWorldQuaternion(ke.modelObject.quaternion);

console.log("model set");
}
```