2.6 原生模块

如果 React VR 满足不了你的需求，你可以自定义的

有时在 React VR 没相关的组件,应用需要接入平台的 API，也许你要复用已经存在的但在 react 中没有

声明的 js 代码，如想写一些体验更好的代码、图片处理的多线程代码、数据库或者更高级的功能。

立方体例子

其中的一个用到原生模块的例子就是在 React VR UI 和从 three.js 添加到场景中的物体之间的交互，

比如说在 Three.js README 里面的几何立方体。

设置场景

React VR 框架能处理相机和渲染设置，你只需关注在场景中添加物体，关注物体的运动即可，首先

在开始项目的一个文件 `vr/client.js` 的 `init` 函数中添加场景，并把场景做为 `VRInstance` 构造函数

的一个参数，原生模块稍后再讲。

```
const scene = new  THREE.Scene();
const cubeModule = new  CubeModule();
```

```
const vr = new  VRInstance(bundle, 'CubeSample', parent, {
  cursorVisibility: 'visible',
  nativeModules: [ cubeModule ],
  scene: scene,
});
```

下一步我们创建立方体 mesh 并添加到场景中，这里依旧用 three.js，我们修改几个地方让 three.js 物

体正确的出现在 React VR 中。

- React VR 采用米作为单位，立方体的长度我们用 1 来替代 100；
- VRInstance 相机在原点，我们只改变立方体 Z 轴的距离就可以看到了。

```
const cube = new  THREE.Mesh(
  new THREE.BoxGeometry(1, 1, 1),
  new THREE.MeshBasicMaterial(),
);
cube.position.z = -4;
scene.add(cube);
cubeModule.init(cube);
```

上面我们用 cube 对 `cubeModule` 进行初始化了，最后我们在 `cubeModule` 方法里面写帧更新的逻辑。

```
vr.render = function(timestamp) {
  const seconds = timestamp / 1000;
  cube.position.x = 0 + (1 * (Math.cos(seconds)));
  cube.position.y = 0.2 + (1 * Math.abs(Math.sin(seconds)));
```

```
};
```

使用原生模块

假设我想通过点击按钮改变立方体的颜色，就可以通过 React Native 桥，异步调用原生模块的 `changeCubeColor` 函数，上面的例子中在 client.js 中已经有了构造函数和 init 函数了，另外还有一个就是原生模块的。

```
export default class CubeModule extends Module {
  constructor() {
    super('CubeModule );
  }
  init(cube) {
    this.cube = cube;
  }
  changeCubeColor(color) {
    this.cube.material.color = new THREE.Color(color);
  }
};
```

然后我们在 `index.vr.js` 的 `onClick` 函数里调用 `changeCubeColor`

```
import NativeModules from 'react-vr';
...
const CubeModule = NativeModules.CubeModule;
...
render() {
...
  <VrButton
```

```
    onClick={()=>CubeModule.changeCubeColor(hexColor)}
  ...
  </VrButton>
...
```

在 `CubeSample` 中可以查看全部代码。