Advanced Web Technology

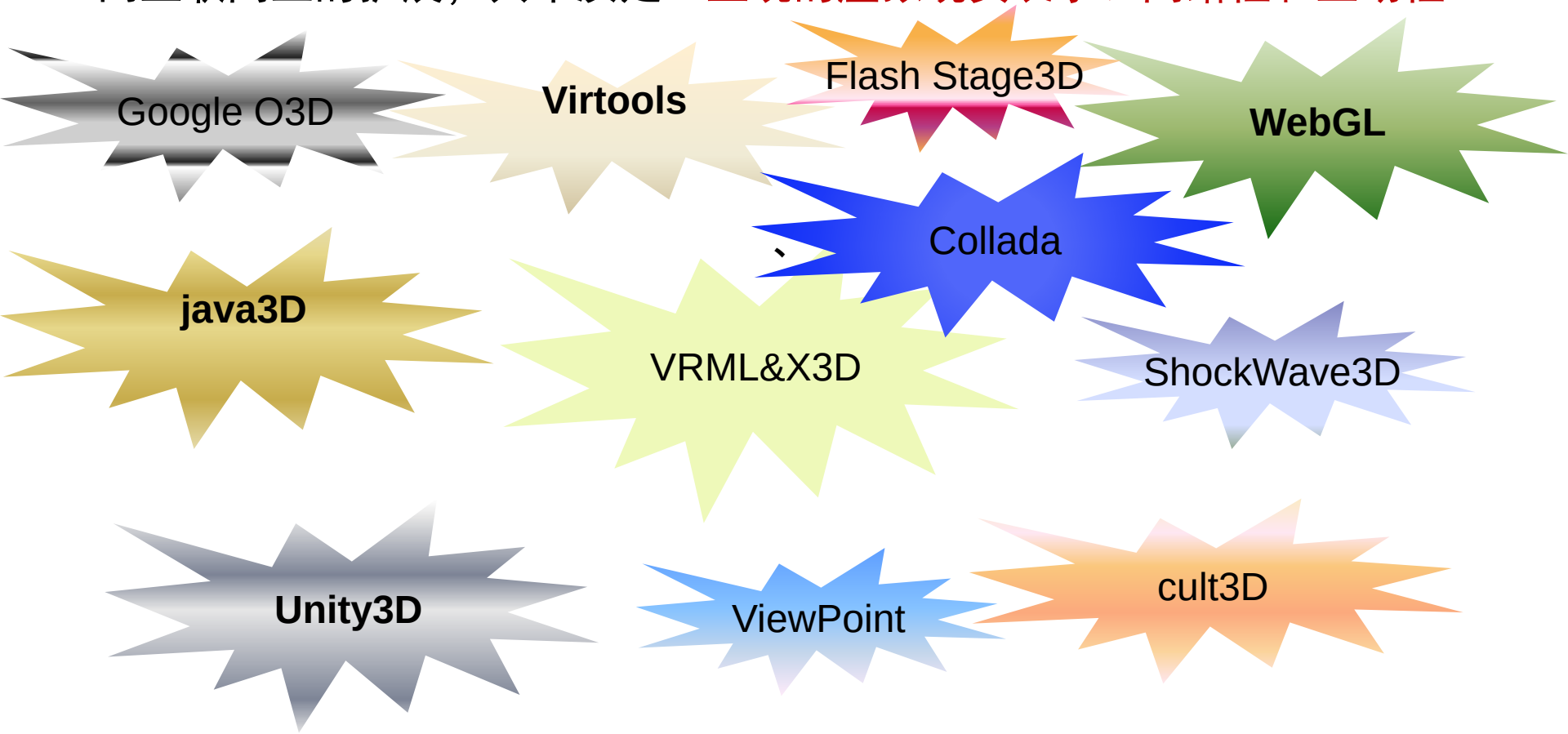
高级Web技术

Web3D

WebGL&Three.js

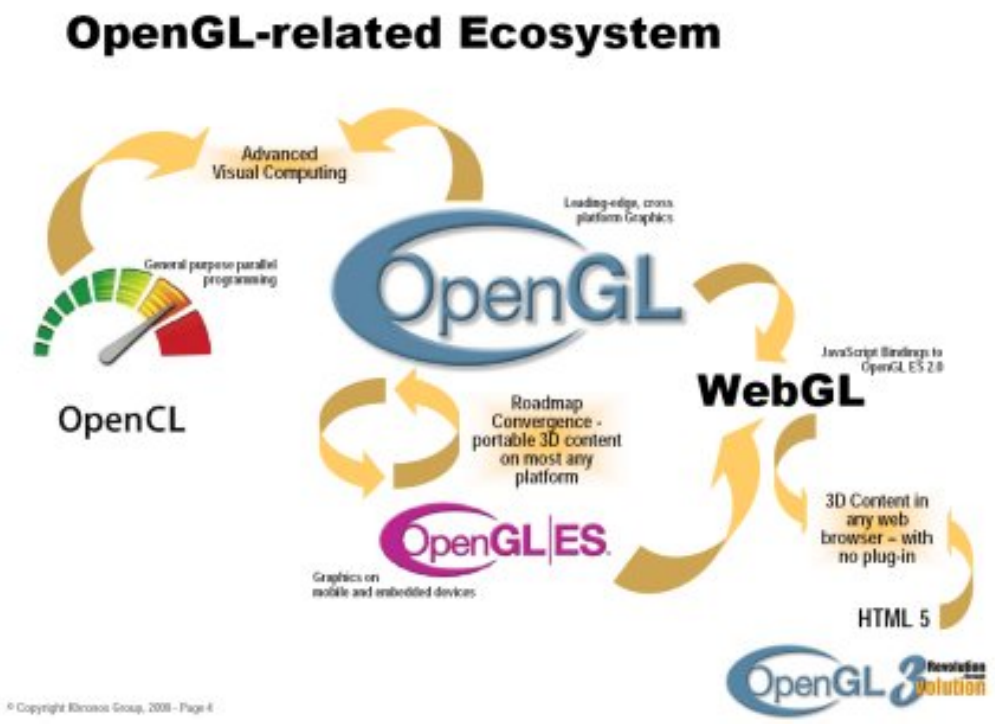
Web3D相关技术

Web3D技术可以看作是Web技术和虚拟现实技术的结合，是虚拟现实技术向互联网上的扩展，其本质是：直观的虚拟现实表示、网络性和互动性。



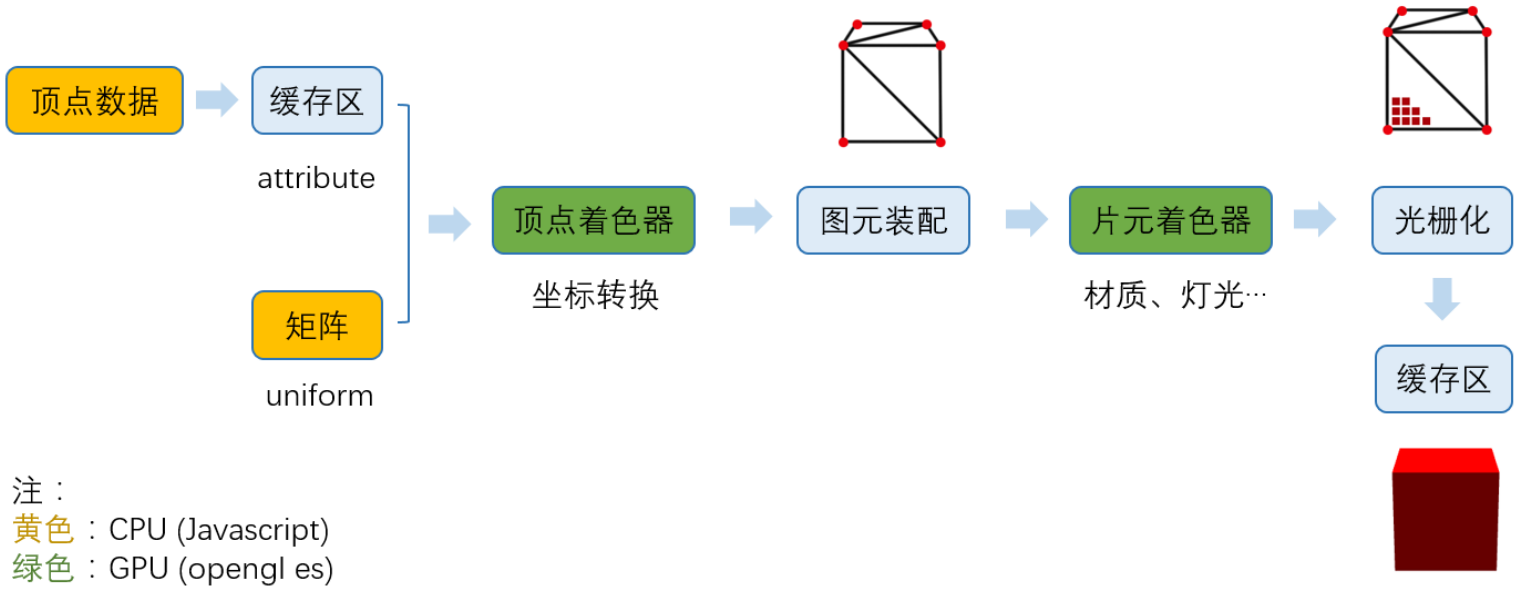
WebGL

- **WebGL**
  - 无插件Web3D技术，实现OpenGL ES 2.0 的JavaScript绑定，可以为HTML5 Canvas提供硬件3D加速渲染



WebGL

- **WebGL**
  - **用于在任何兼容的Web浏览器中呈现交互式3D和2D图形，而无需使用插件。**

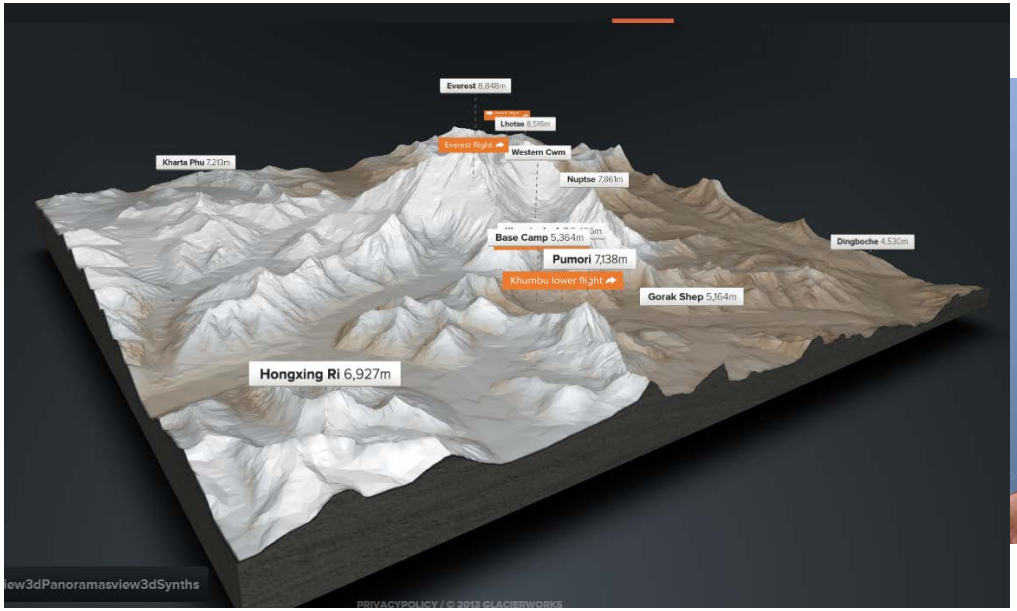


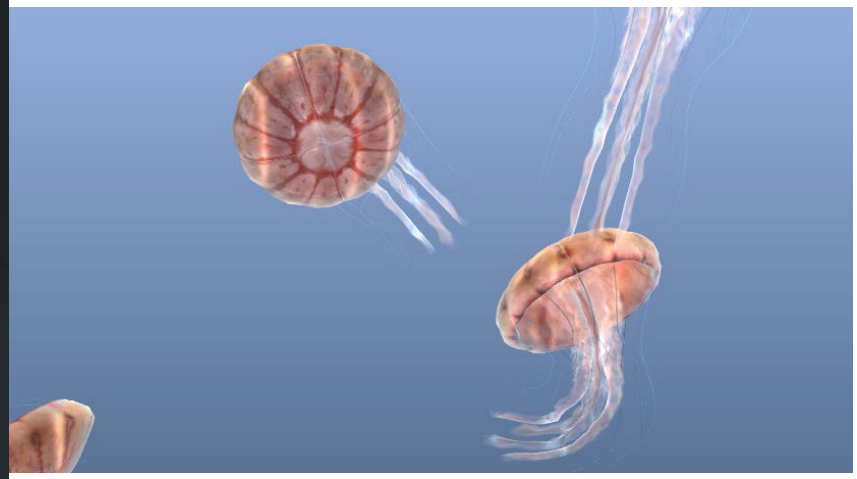
WebGL

- **WebGL展示**

WebGL

- **WebGL展示**





WebGL

- **WebGL编写**

```
▼<html>
  ▼<head>
      <title>WebGL Up And Running - Example 1</title>
      <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
    ▶<script type="text/javascript">…</script>
    </head>
  ▼<body onload="onLoad();">
      <canvas id="webglcanvas" style="border: none;" width="500" height="500">
    </body>
  </html>
```

WebGL

- **WebGL框架**

Name	Cost	Popularity	Rating	Tags	Last Release
Construct 2	varies		★★★★½	game-maker free 2d 3d webgl sounds collisions physics	Nov 12th 2013
pixi.js	free (MIT)		★★★★★	2d webgl free	Oct 17th 2013
PlayCanvas	free		★★★★☆	3d cloud-based free webgl sounds	Jul 31st 2013
Three.js	free (MIT)		★★★★☆	3d webgl free	Oct 21st 2013
Turbulenz	free (MIT)		★★★★½	2d 3d webgl sounds collisions physics debug networking	Oct 30th 2013
CAAT	free (MIT)		★★★★☆	2d free webgl	Jul 2nd 2013
Phaser	free (MIT)		★★★★★	flash-like 2d sounds collisions physics typescript webgl free	Nov 1st 2013
enchant.js	free (MIT)		★★★★☆	2d sounds collisions physics webgl free	Sep 17th 2013
voxel.js	free (BSD)		★★★½☆	webgl 3d voxel sounds physics networking	Jul 24th 2013
PlayCanvas	free		no ratings	3d cloud-based free webgl sounds	unknown

WebGL

- **WebGL框架：Three.js**
  - **封装底层的WebGL，兼顾灵活性和易用性**
  - **遵循CSS-3D规范的新渲染器，兼容性更好**
  - **Three.js官网地址：http://threejs.org/**
  - **Three.js库：https://github.com/mrdoob/three.js/**

WebGL

- **Three.js**
  - **程序基本框架**

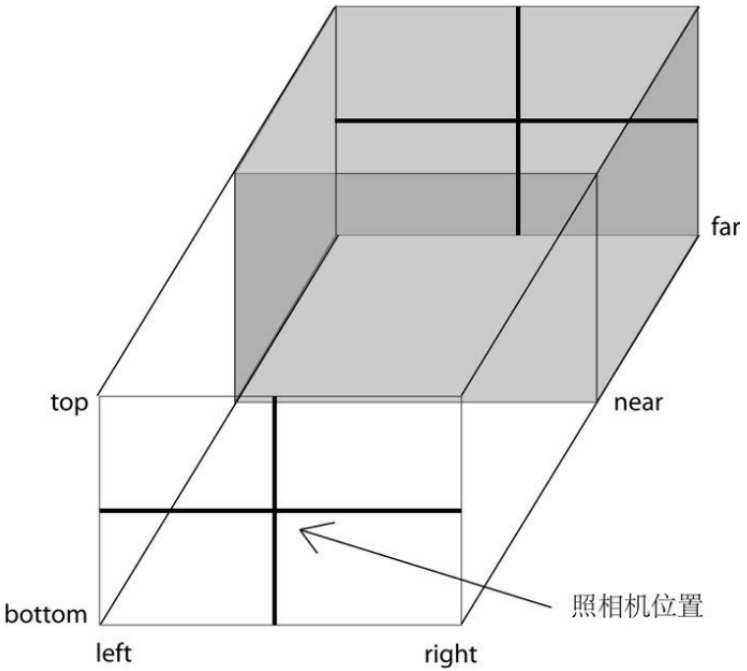
基本流程：
创建渲染器(renderer)
创建场景(scene)
添加相机(camera)
添加网格模型(mesh)
添加灯光(light)
渲染(render)

```
<script src="../libs/Three.js"></script>
<script>
function onLoad()
{
    // Grab our container div
    var container = document.getElementById("container");

    // Create the Three.js renderer, add it to our div
    var renderer = new THREE.WebGLRenderer();
    renderer.setSize(container.offsetWidth, container.offsetHeight);
    container.appendChild( renderer.domElement );

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

    // Create a camera and add it to the scene
    var camera = new THREE.PerspectiveCamera( 45,
        container.offsetWidth / container.offsetHeight, 1, 4000 );
    camera.position.set( 0, 0, 3.3333 );
    scene.add( camera );

    // Now, create a rectangle and add it to the scene
    var geometry = new THREE.PlaneGeometry(1, 1);
    var mesh = new THREE.Mesh( geometry,
        new THREE.MeshBasicMaterial( ) );
    scene.add( mesh );

    // Render it
    renderer.render( scene, camera );
}
```

WebGL

- **Three.js**
  - **相机**
    - 正交投影:对于制图、建模软件通常使用正交投影

```
THREE.OrthographicCamera(left, right, top, bottom, near, far)
```

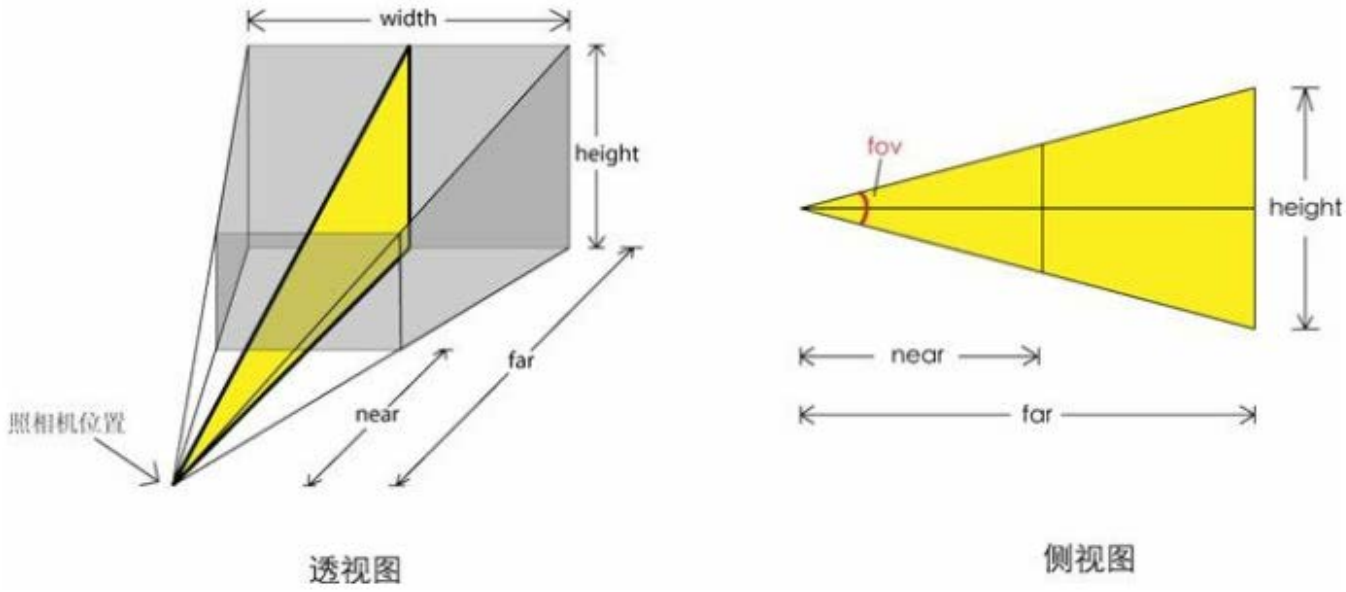


WebGL

- **Three.js**
  - **相机**
    - 透视投影：更接近人眼的观察效果，大多数应用通常采用

```
THREE.PerspectiveCamera(fov, aspect, near, far)
```

Aspect=width / height



WebGL

- **Three.js**
  - **几何形状**
    - 储存了一个物体的顶点信息。
    - WebGL需要程序员指定每个顶点的位置，而在Three.js中，可以通过指定一些特征来创建几何形状，

CircleGeometry	PlaneGeometry
ConvexGeometry	PolyhedronGeometry
CubeGeometry	ShapeGeometry
CylinderGeometry	SphereGeometry
ExtrudeGeometry	TetrahedronGeometry
IcosahedronGeometry	TextGeometry
LatheGeometry	TorusGeometry
OctahedronGeometry	TorusKnotGeometry
ParametricGeometry	TubeGeometry

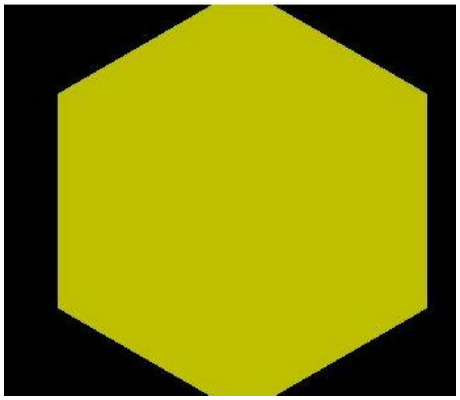
使用文字形状需要下载和引用额外的字体库，可以在http://typeface.neocracy.org/下载。

WebGL

- **Three.js**
  - **基本材质**
    - 是独立于物体顶点信息之外的与渲染效果相关的属性。通过设置材质可以改变物体的颜色、纹理贴图、光照模式等。

```
THREE.MeshBasicMaterial(opt)
```

- visible：是否可见，默认为true
- opacity ：透明度
- side：渲染面片正面或是反面，默认为正面THREE.FrontSide，可设置为反面THREE.BackSide，或双面THREE.DoubleSide
- wireframe：是否渲染线而非面，默认为false
- color：十六进制RGB颜色
- map：使用纹理贴图

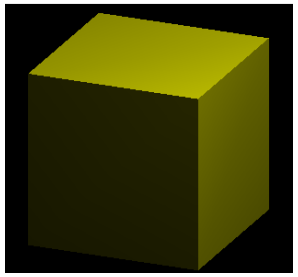


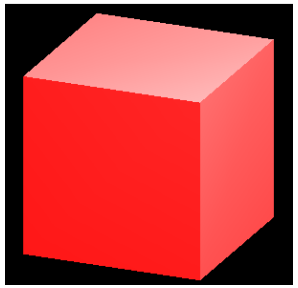
WebGL

- **Three.js**
  - **Lambert材质**
    - 符合Lambert光照模型的材质。只考虑漫反射而不考虑镜面反射的效果，因而对于金属、镜子等需要镜面反射效果的物体就不适应。

new THREE.MeshLambertMaterial (opt)

- color： 是用来表现材质对散射光的反射能力，也是最常用来设置材质颜色的属性。
- Ambient：表示对环境光的反射能力，只有当设置了AmbientLight后，该值才是有效的。
- Emissive：材质的自发光颜色。



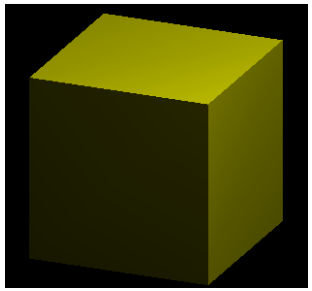


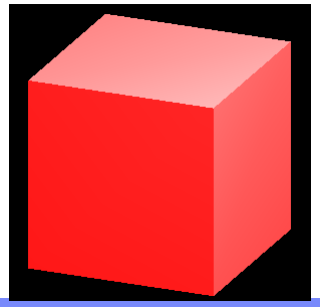
WebGL

- **Three.js**
  - **Phong材质**
    - Phong模型考虑了镜面反射的效果，因此对于金属、镜面的表现尤为适合。
    - 漫反射部分和Lambert光照模型是相同的。

new THREE.MeshPhongMaterial (opt)

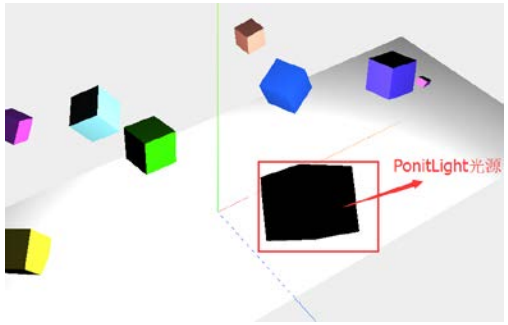
- specular:镜面反射系数
- shininess;高光指数，越大则高光光斑越小。

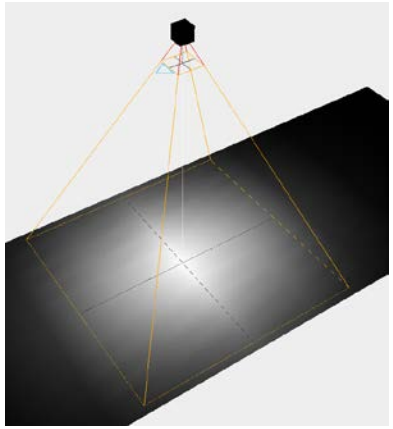




WebGL

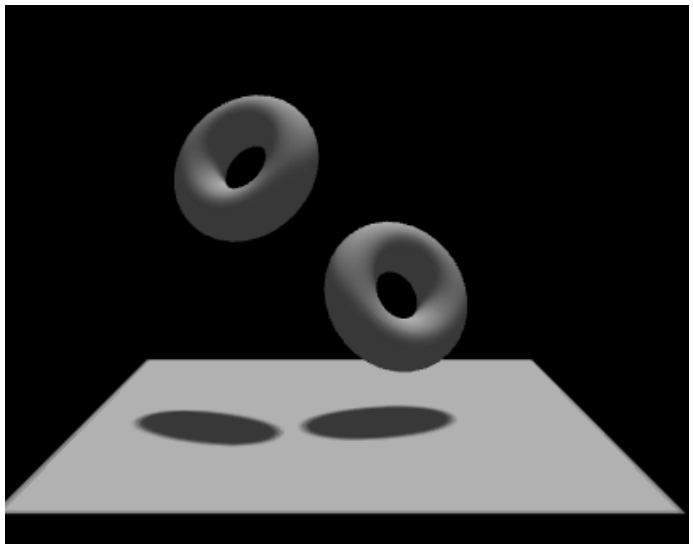
- Three.js
  - 光照：Light
    - AmbientLight:
      - 环境光，基础光源，它的颜色会被加载到整个场景和所有对象的当前颜色上。
    - PointLight：
      - 点光源，朝着所有方向都发射光线
    - SpotLight ：
      - 聚光灯光源：类型台灯，天花板上的吊灯，手电筒等
    - DirectionalLight：
      - 方向光，又称无限光，从这个发出的光源可以看做是平行光.

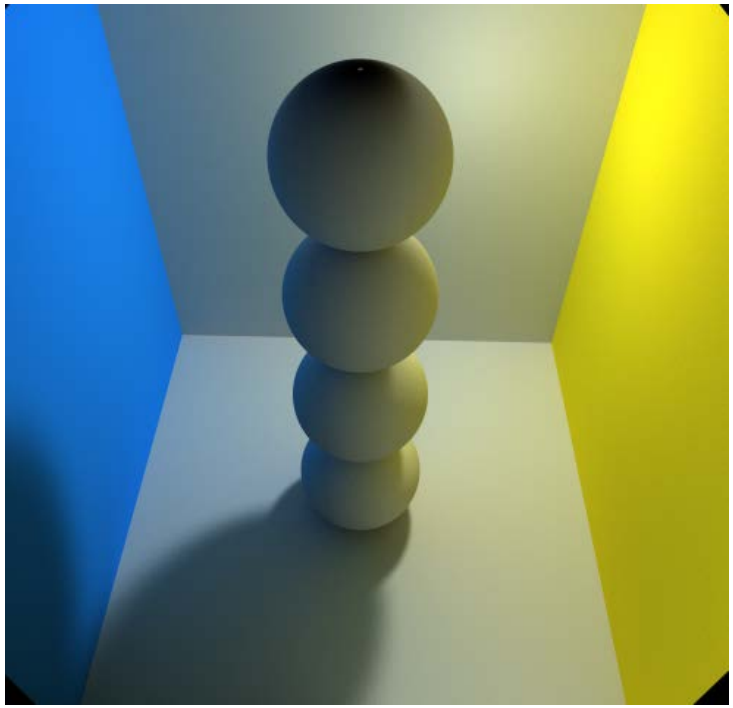




WebGL

- **Three.js**
  - 光照与阴影
    - **光线追踪**

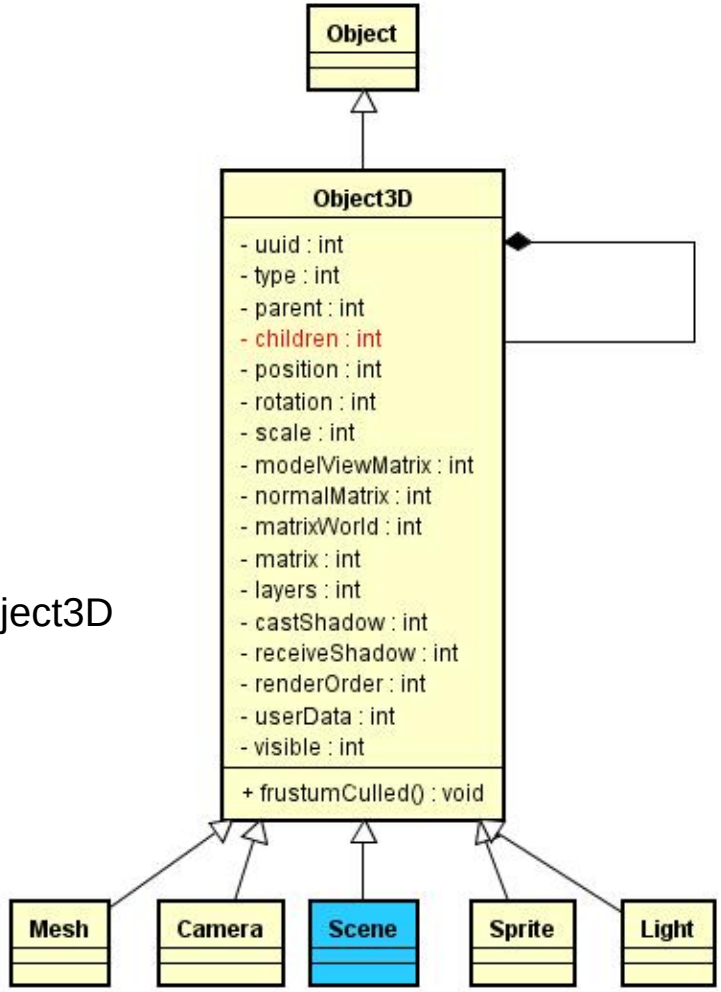




WebGL

- **Three.js**
  - **Object3D：各种3D物体的基类**
    - **3D物体、光源等都是它的子类**
    - **重要属性：**
      - **position、rotation、scale**
      - **通过改变它们来实现各种交互效果**

官方文档：https://threejs.org/docs/index.html#api/core/Object3D



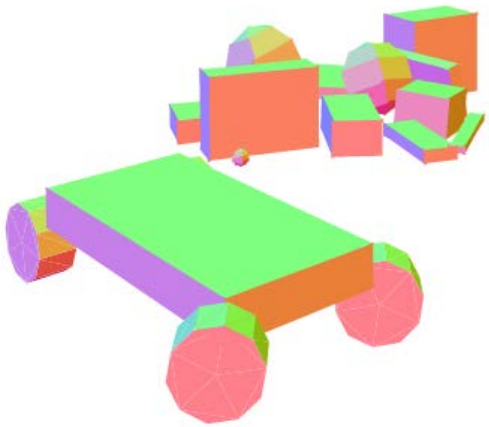
WebGL

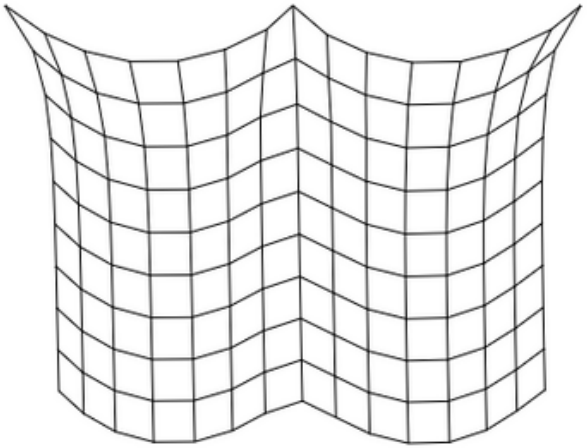
- **Three.js**
  - **天空盒 Skybox：**
    - **实际上是一个立方体对象。用户视角只在盒子内部活动，所以只需要渲染盒子内部表面。**
    - **天空盒子应当足够大，使得摄像机在移动时看天空仍然觉得足够远。但是，天空盒子不能超出摄像机最远可视范围。**


```
var skyBoxGeometry = new THREE.BoxGeometry(500, 500, 500);
var skyBoxMaterial = new THREE.MeshBasicMaterial({
      color: 0x9999ff,
      side: THREE.BackSide
    });
var skyBox = new THREE.Mesh(skyBoxGeometry, skyBoxMaterial);
scene.add(skyBox);
```

WebGL

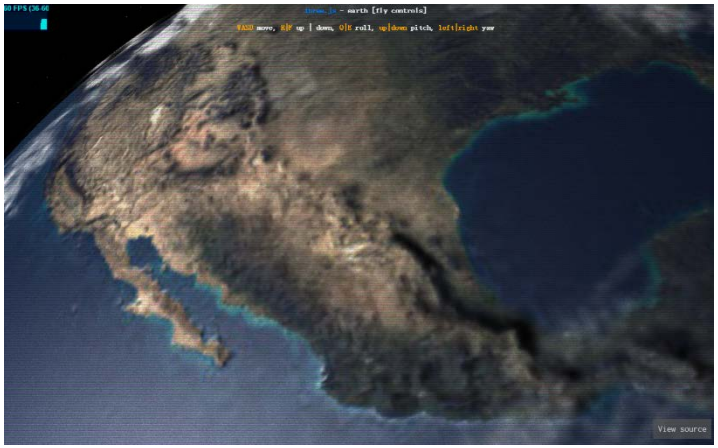
- Three.js
  - 碰撞检测
    - Raycaster：检测射线与物体相交
    - 可用于鼠标选择物体、简单的两物体间碰撞检测等
    - https://threejs.org/docs/index.html#api/core/Raycaster
  - 物理引擎
    - Physi.js、Cannon.js

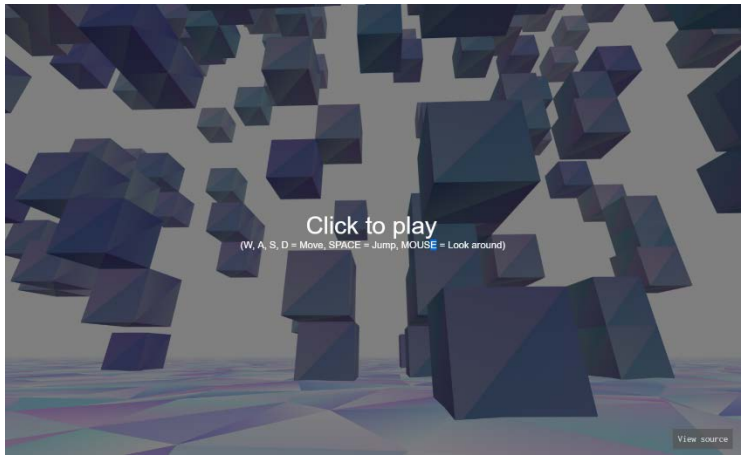




WebGL

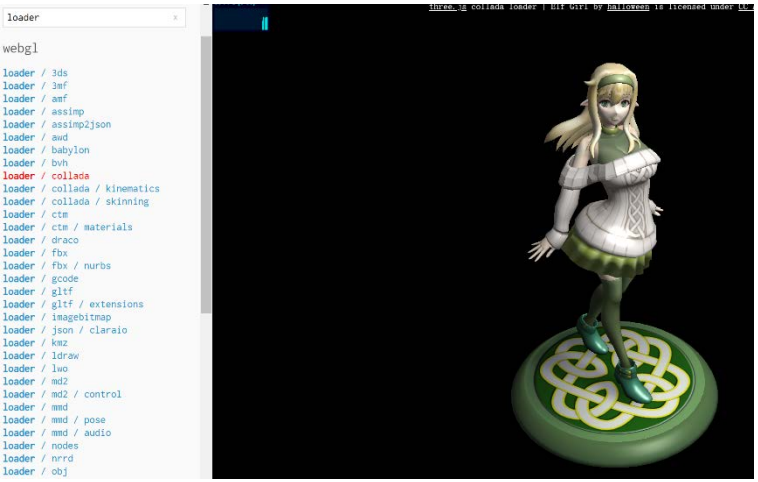
- Three.js
  - 输入控制
    - 参考：https://threejs.org/examples/下 misc/controls
    - 模拟飞行： controls/fly
    - 第一人称控制器： controls/pointerlock





WebGL

- **Three.js**
  - **加载外部模型：**
    - **参考 https://threejs.org/examples/ 下 loader 相关例子的源代码。**




```
// loading manager

var loadingManager = new THREE.LoadingManager( function () {

  scene.add( elf );

} );

// collada

var loader = new THREE.ColladaLoader( loadingManager );
loader.load( './models/collada/elf/elf.dae', function ( collada ) {

  elf = collada.scene;

} );
```

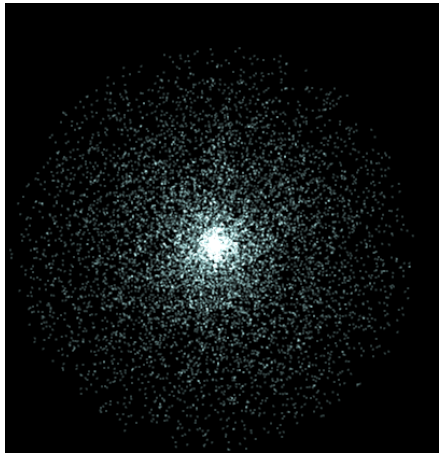
WebGL

- **Three.js**
  - **加载音频**
    - **参考 https://threejs.org/docs/index.html#api/audio/Audio**

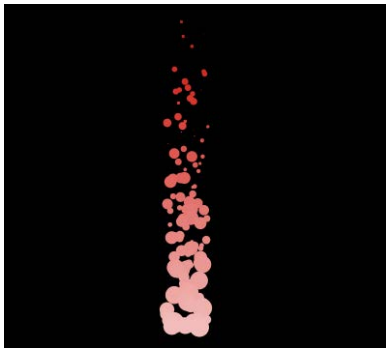
```
// create an AudioListener and add it to the camera
var listener = new THREE.AudioListener();
camera.add( listener );

// create a global audio source
var sound = new THREE.Audio( listener );

// load a sound and set it as the Audio object's buffer
var audioLoader = new THREE.AudioLoader();
audioLoader.load( 'sounds/ambient.ogg', function( buffer ) {
    sound.setBuffer( buffer );
    sound.setLoop( true );
    sound.setVolume( 0.5 );
    sound.play();
});
```

WebGL

- **Three.js**
  - **粒子系统:**三维计算机图形学中模拟一些特定的模糊现象的技术
  - 使用粒子系统模拟的现象有火、爆炸、烟、水流、火花、落叶、云、雾、雪等效果。



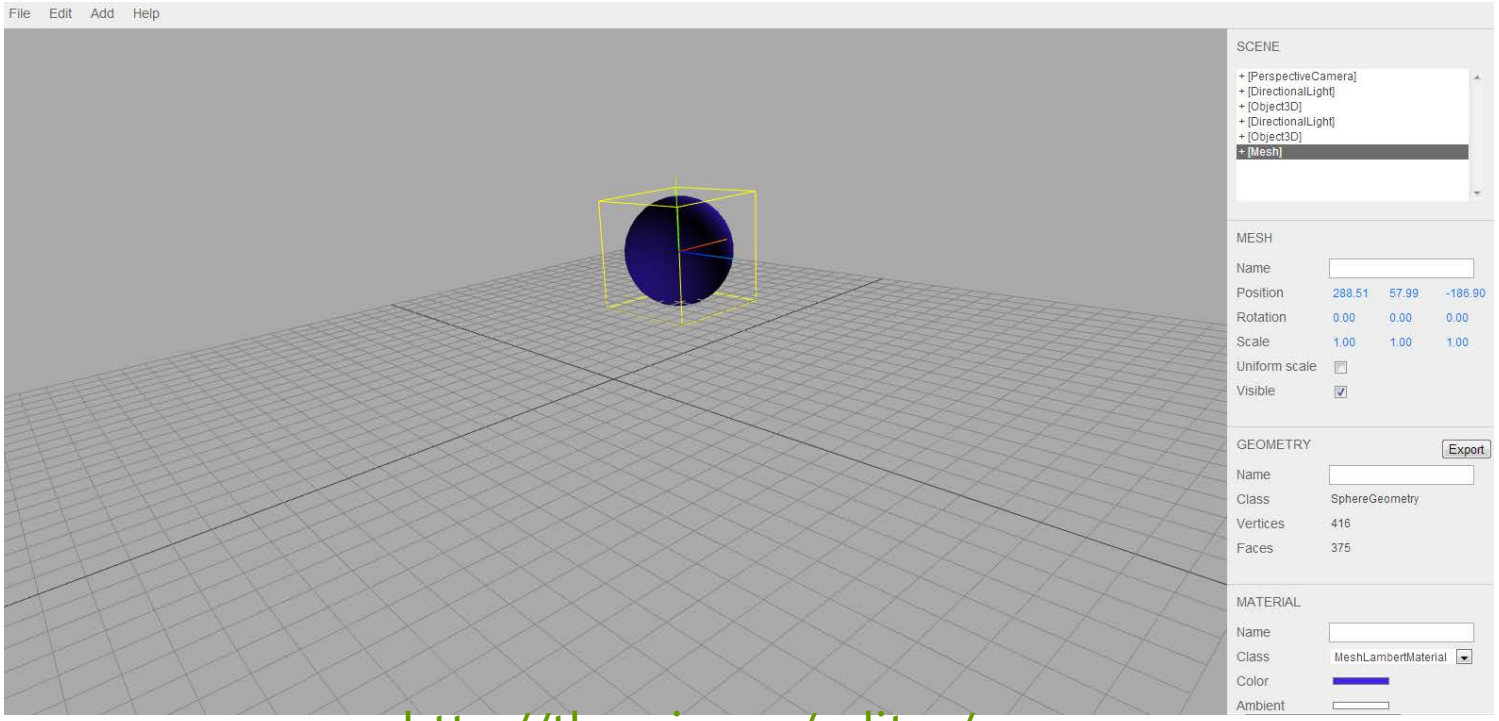
Webgl cloud



Demo netlogo

WebGL

- **Web3D相关技术**
  - **WebGL**
    - 学习站点：http://learningwebgl.com/blog/
    - WebGL可视化编辑网站



http://threejs.org/editor/

WebGL

- **WebGL**
  - WebGL可视化编辑网站

GOO
http://www.gootechnologies.com/
http://www.goocreate.com/learn/

Construct2
http://www.gootechnologies.com/
http://www.goocreate.com/learn/

WebGL

- **WebGL**
  - WebGL可视化编辑网站

http://www.ambiera.com/coppercube/index.html

基于WebGL的NVE

- 网络虚拟环境（Networked Virtual Environment)
  - 利用计算机图形学、计算机网络、人工智能、人机接口等计算机技术构造的一个真实世界的模拟，地理上分布的用户可以通过网络共享该环境，并多通道地与周围的环境以及在相互之间进行交互。

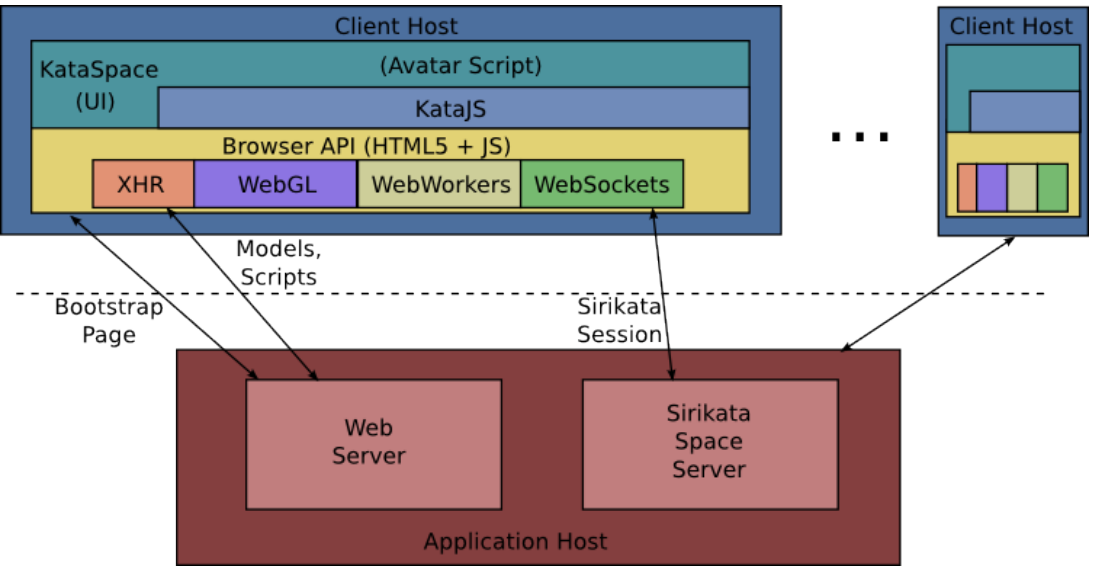
基于WebGL的NVE

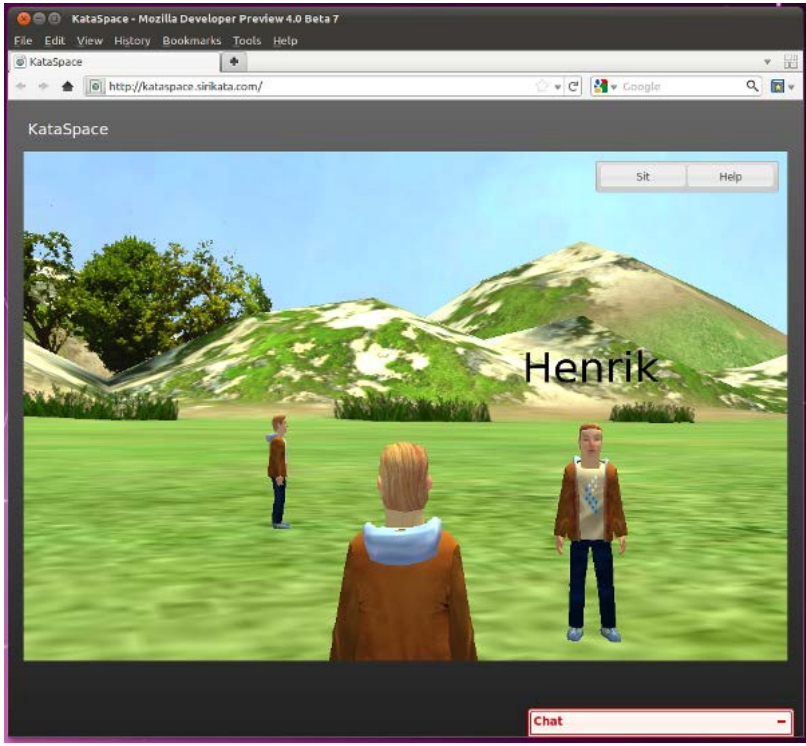
- **分布式虚拟环境必须具有以下功能：**
  - **可视模拟真实世界：**三维模拟构造真实世界；支持多媒体内容；通过硬件给用户触觉等真实感受。
  - **数据共享：** 该环境中的数据可以在一定规则下被进入的用户共享。用户应该看到的是一个统一的视图。
  - **交互性：**用户能以替身(Avatar)形式出现，并通过一定的输入设备与环境和其他用户进行交互。

虚拟现实的3I特征
浸没感（Immersion）
交互性（Interactivity）
构想性 （Imagination）

基于WebGL的NVE

- **WebGL**
  - Kataspace: browser-based virtual worlds built with WebGL and HTML5





基于WebGL的NVE

- 场景与用户的交互更新
  - 典型的游戏循环：

```
while (true)
{
  processInput();
  update();
  render();
}
```

- JavaScript 的游戏循环
  - 使用 requestAnimationFrame 代替 while

```
function loop() {
    processInput();
    update();
    render();
    requestAnimationFrame(loop);
}
```

基于WebGL的NVE

- **网络功能**
  - socket.io封装了websocket ：

client ：

建立连接：

```
<script src="/socket.io
<script>
  var socket = io();
</script>
```

发送/接收消息：

```
socket.emit('chat', 'xxxxxx');
```

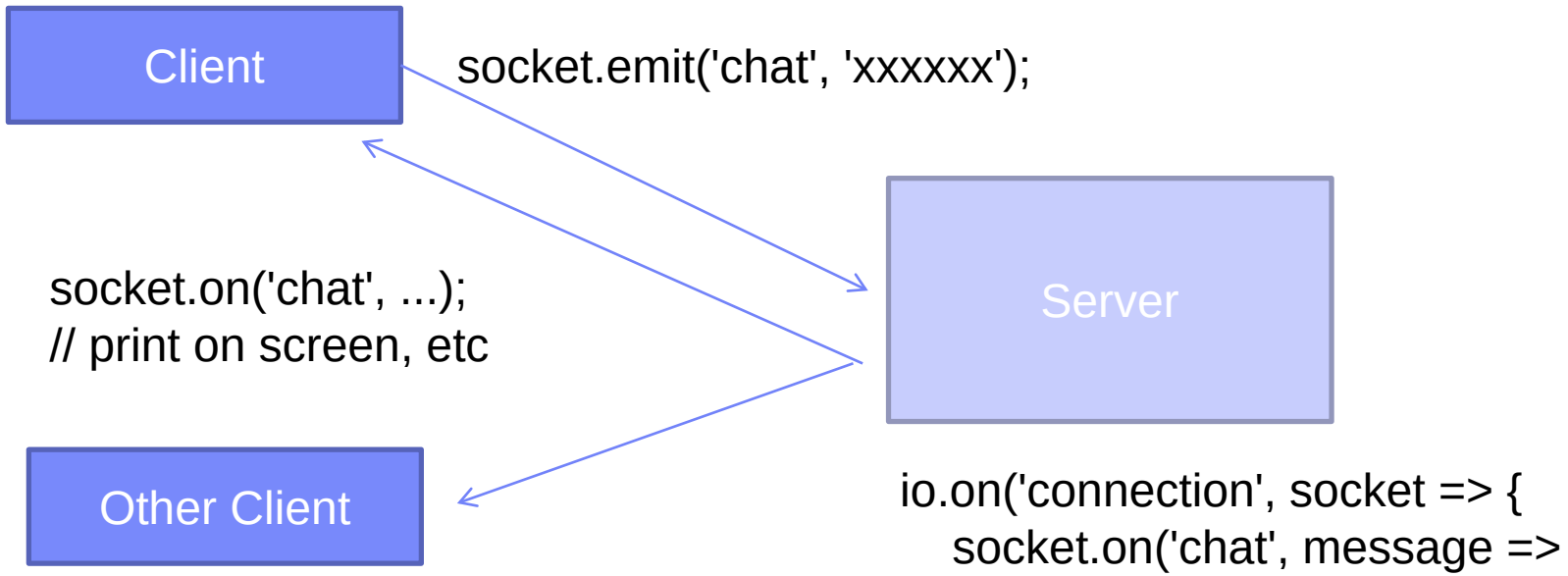
server：

```
io.on('connection', function(socket){
  console.log('a user connected');
  socket.on('disconnect', function(){
    console.log('user disconnected');
  });
});
```

```
socket.on('chat', message => {
      // do something
});
```

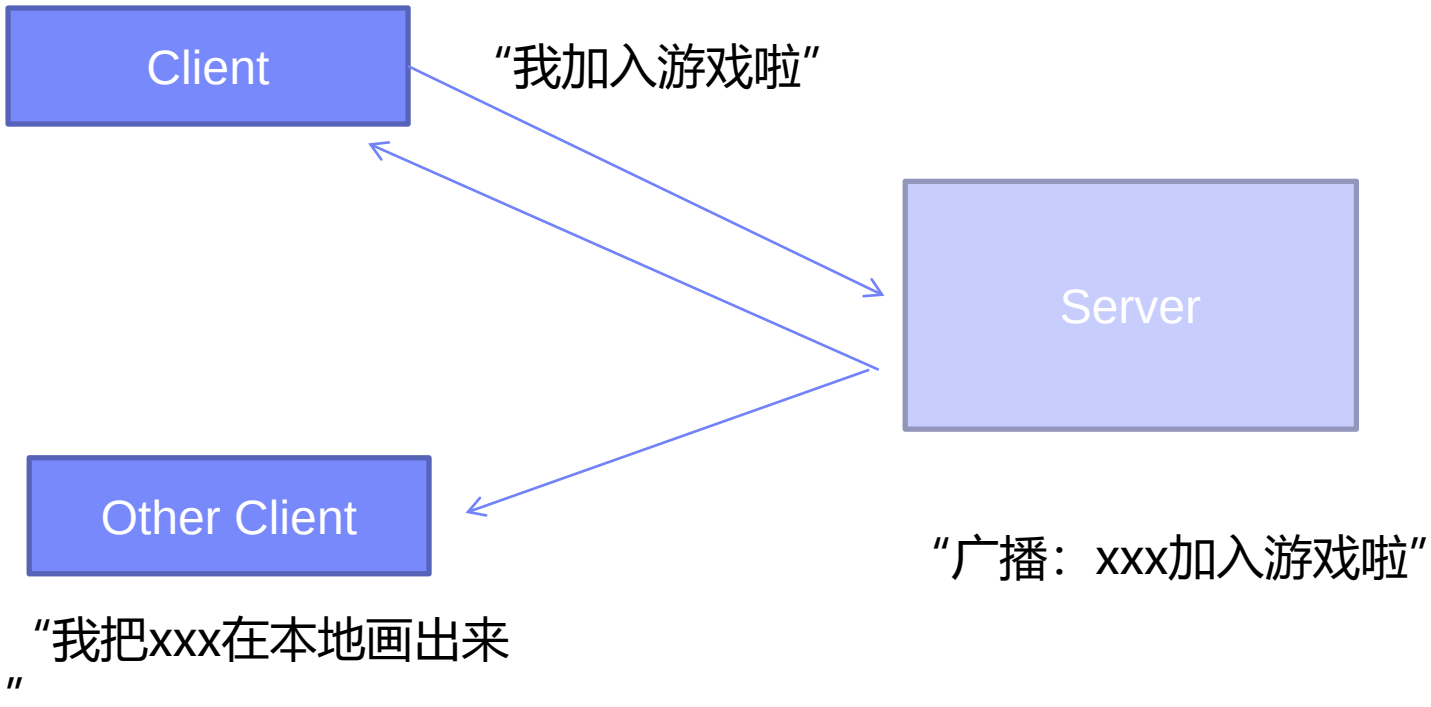
基于WebGL的NVE

- **网络功能**
  - **采用socket.io聊天室例子：**



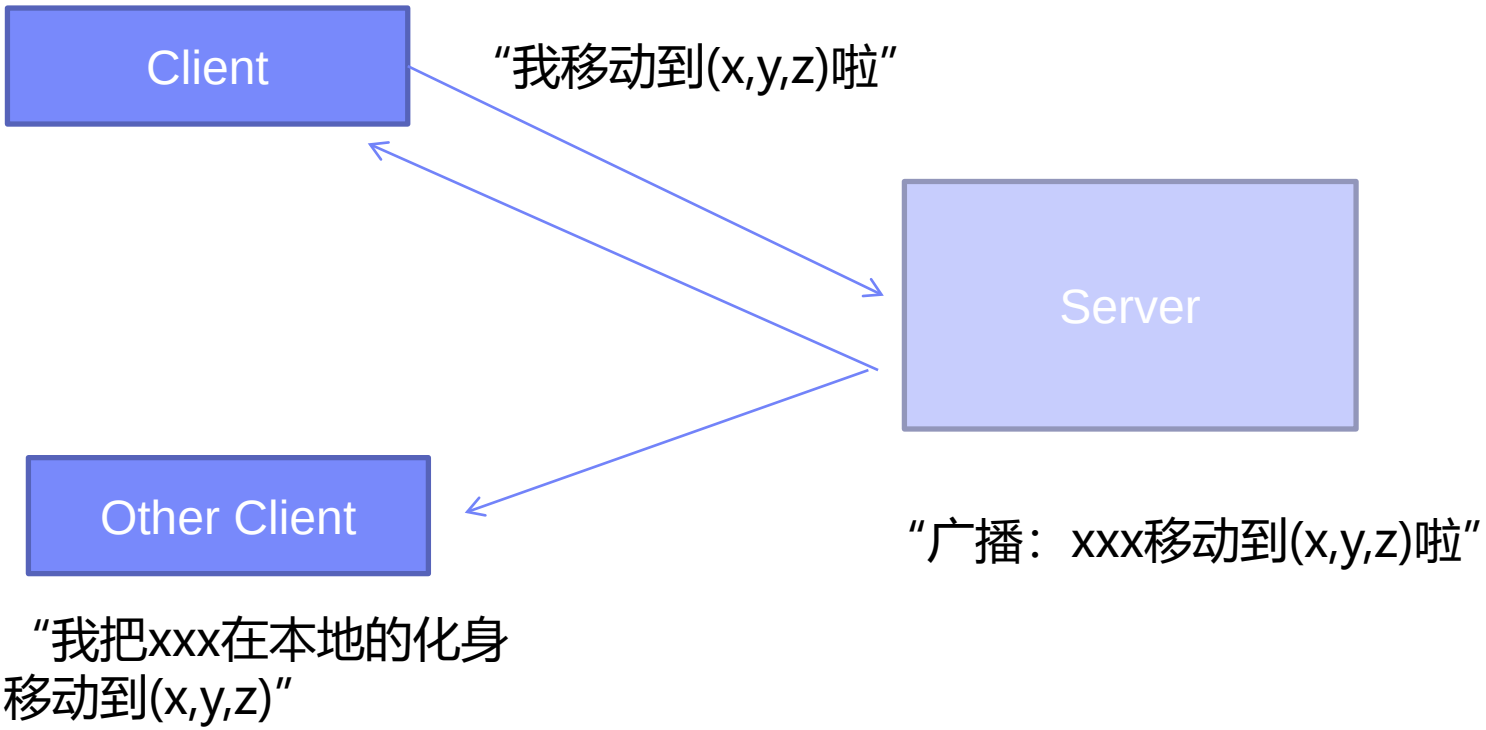
基于WebGL的NVE

- 网络功能
  - 所有玩家相互可见：



基于WebGL的NVE

- 网络功能
  - 行为分享：



基于WebGL的NVE

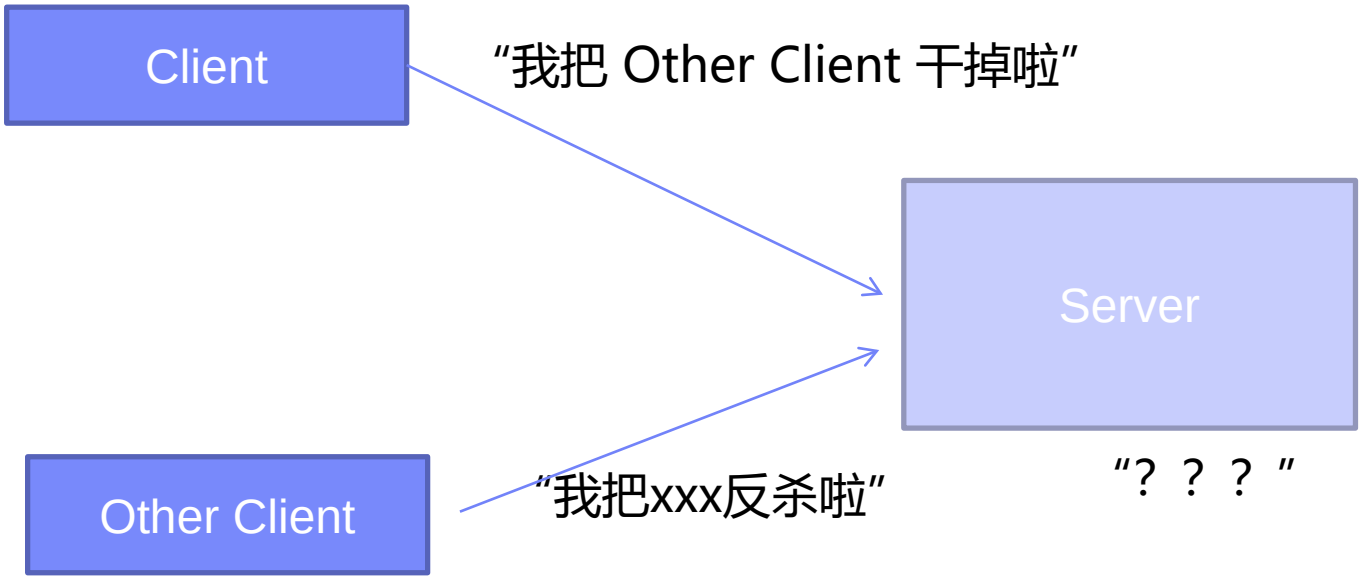
- 网络功能
  - 行为分享：客户端定时上报自身的位置信息

```
function loop() {
    processInput();
    update();
    render();
    requestAnimationFrame(loop);
}
```

```
reportToServer() {
    // 判断上次上报时间
    // 大于一定的时间才上报移动
}
```

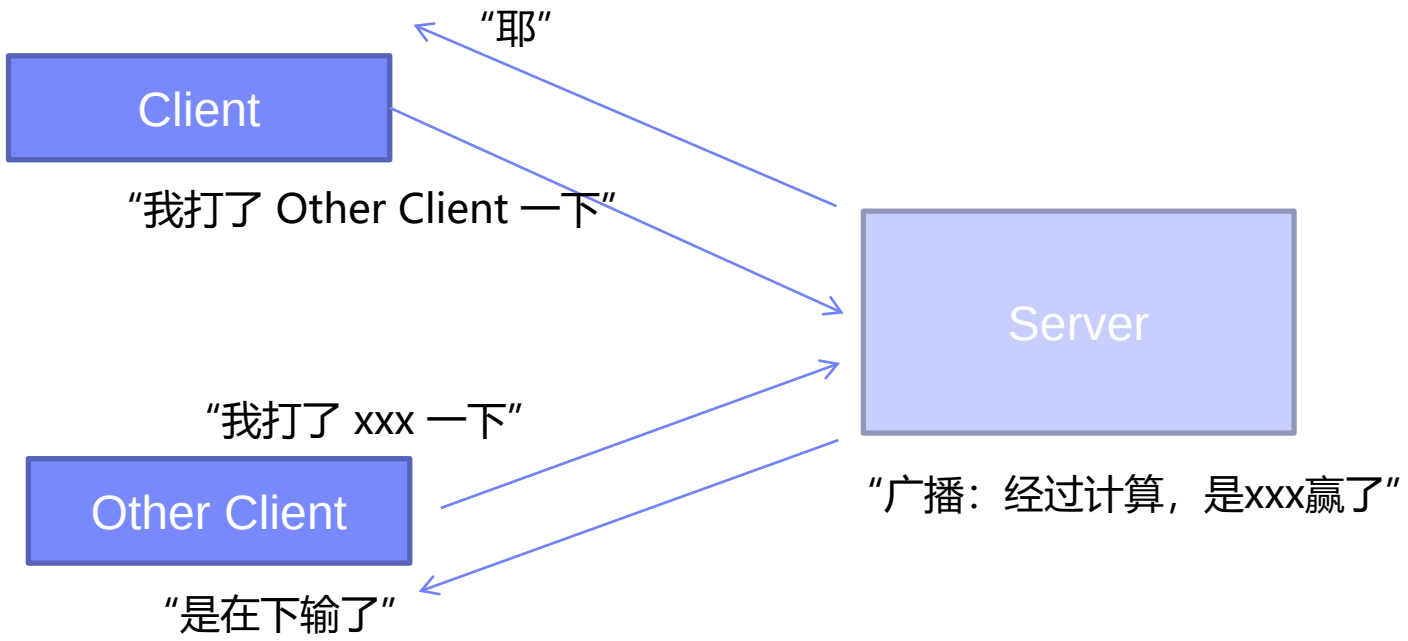
基于WebGL的NVE

- 网络功能
  - 行为分享：同步游戏的状态



基于WebGL的NVE

- 网络功能
  - 行为分享：权威服务器
    - 服务器上也跑一个游戏循环
    - 客户端只上报输入，服务端计算结果并广播



基于WebGL的NVE

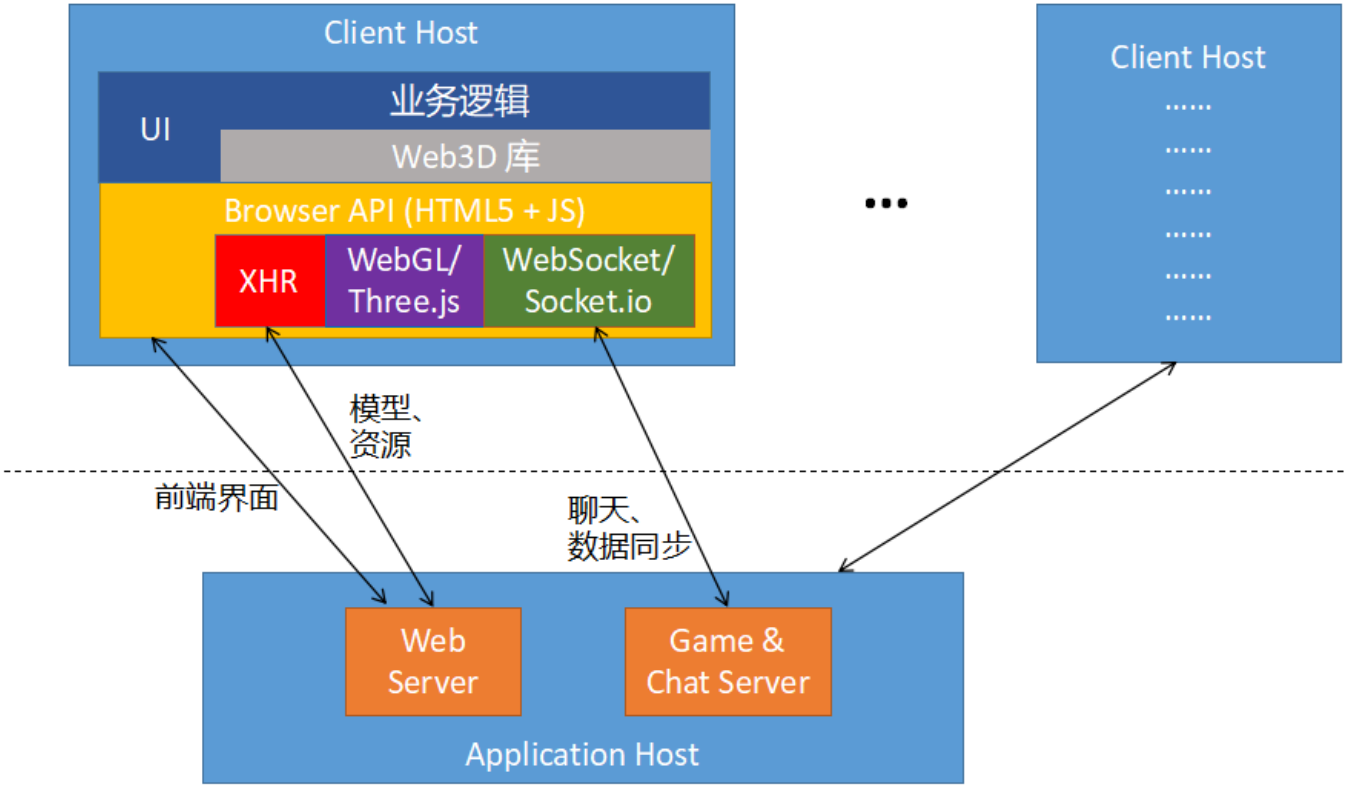
- 网络功能
  - 行为分享：权威服务器
    - 定时向所有客户端发送场景的状态

```
function loop() {
    processInput(); // 从客户端读到的输入
    update();
    render();
    setImmediate(loop);
}
```

```
reportToClient() {
    // 判断上次同步时间
    // 大于一定的时间同步所有客户端
}
```

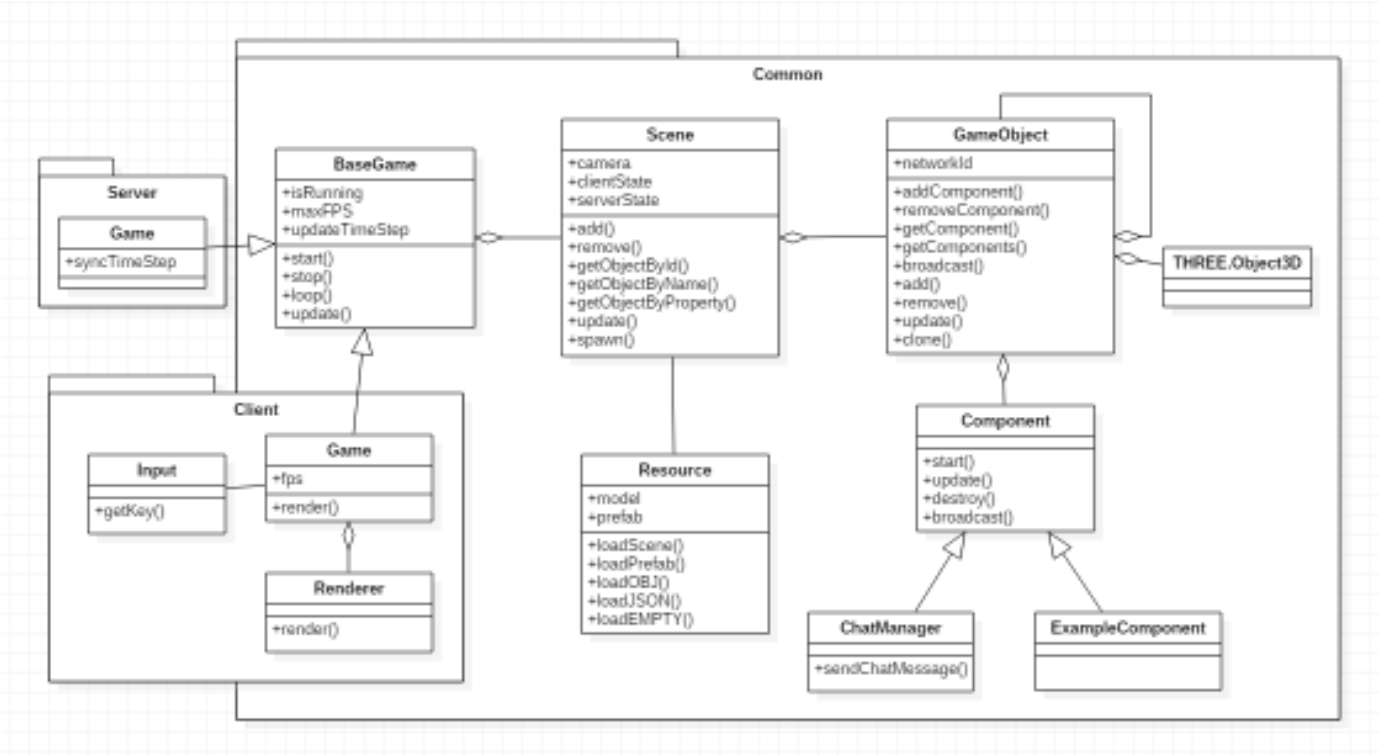
基于WebGL的NVE

- 采用Three.js和Socket.io的NVE工程实例



基于WebGL的NVE

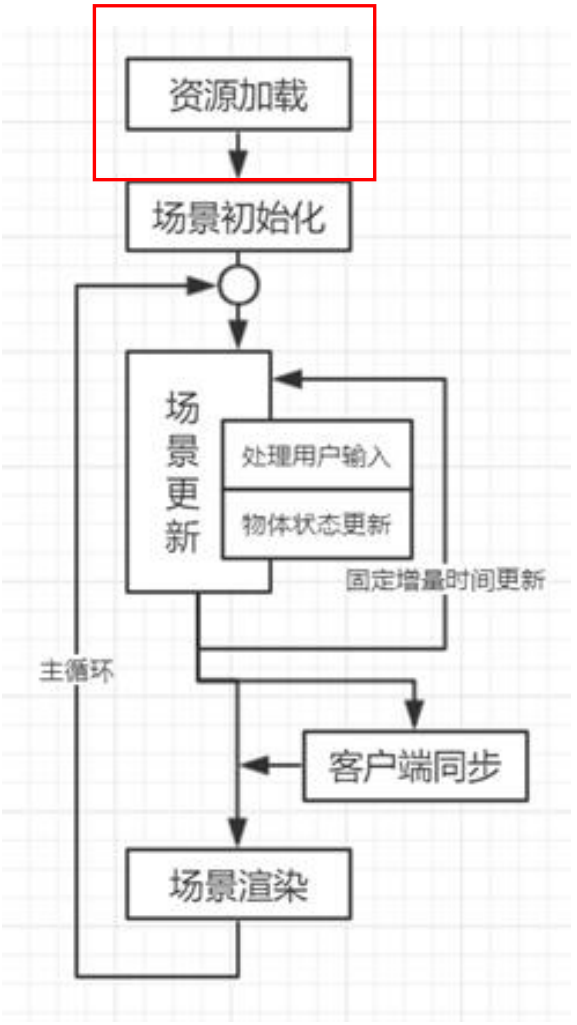
- 采用Three.js和Socket.io的NVE工程实例
  - 框架类图



基于WebGL的NVE

- 采用Three.js和Socket.io的NVE工程实例
  - 资源加载、场景初始化：

```
// 创建 three.js 渲染器，并附加到 div 元素下
var renderer = new THREE.WebGLRenderer();
renderer.setSize(container.offsetWidth, container.offsetHeight);
container.appendChild(renderer.domElement);

// 创建 three.js 场景
var scene = new THREE.Scene();

// 创建相机并加入场景
var camera = new THREE.PerspectiveCamera(45,
    container.offsetWidth / container.offsetHeight, 1, 4000);
camera.position.set(0, 0, 3.3333);
scene.add(camera);

// 创建一个长方体并加入场景
var geometry = new THREE.PlaneGeometry(1, 1);
var mesh = new THREE.Mesh(geometry,
    new THREE.MeshBasicMaterial());
scene.add(mesh);
```

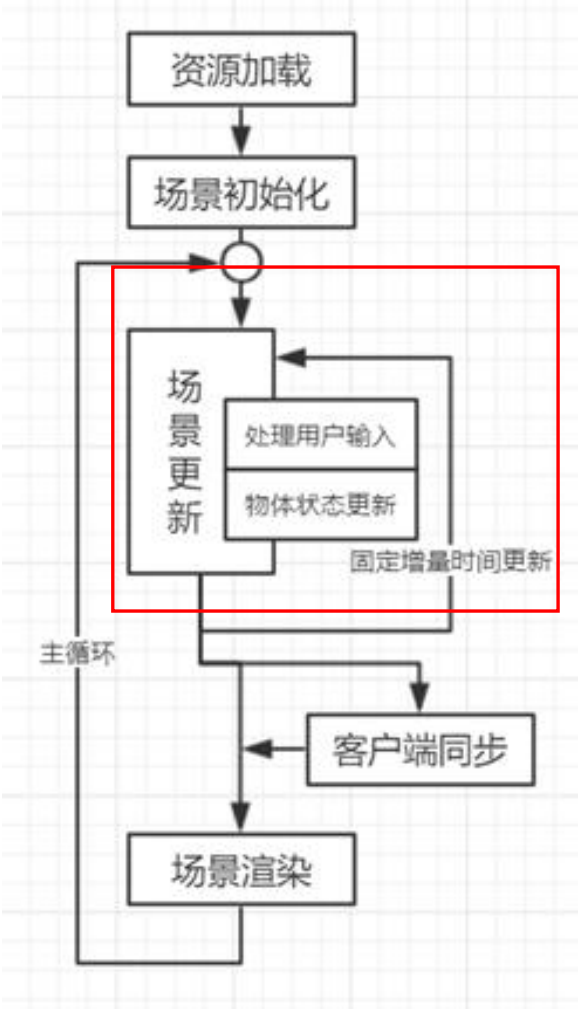


基于WebGL的NVE

- **采用Three.js和Socket.io的NVE工程实例**
  - **场景更新：**

```
function loop() {
    processInput();
    update();
    render();
    requestAnimationFrame(loop);
}
```

```
let frameTime = timestamp - this.lastFrameTime;
this.delta += frameTime;
this.lastFrameTime = timestamp;

while (this.delta >= this.updateTimeStep) {
    this.update(this.updateTimeStep);
    this.delta -= this.updateTimeStep;
}
```



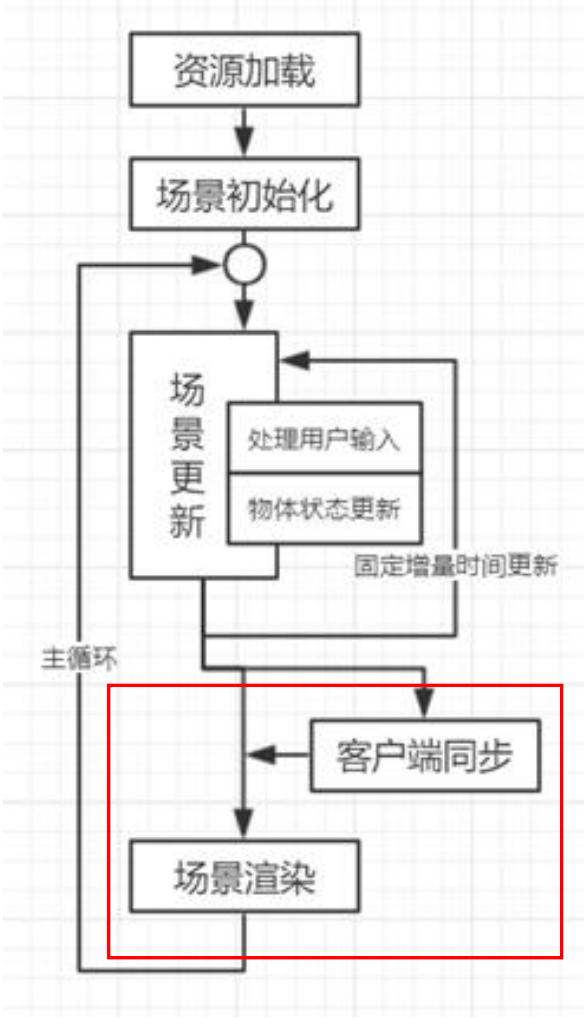
基于WebGL的NVE

- 采用Three.js和Socket.io的NVE工程实例
  - 客户端同步：

```
// sync all clients
if (timestamp - this.lastSyncTime > this.syncTimeStep) {
    io.emit(Event.SERVER_SEND_STATE,
        JSON.stringify(this.scene.serverState));
    this.lastSyncTime = timestamp;
}
```

  - 场景渲染

```
renderer.render( scene, camera );
```



参考资料

- ECMAScript 6 标准：
  - http://es6.ruanyifeng.com/
- 游戏编程模式：
  - http://gameprogrammingpatterns.com/ （en）
  - http://gpp.tkchu.me/ （cn）
- 简单的多人 Web3D Demo：
  - src：https://gitee.com/maliut/aweb-web3d
  - demo： http://maliut.gitee.io/aweb-web3d