

# three.js essentials - Jos Dirksen 2014

- Get Up and Running with Three.js (\*\*介绍three.js开发前期准备工作\*\*)

+ Introducing Three.js(\*\*介绍一下three.js是什么\*\*)

+ Looking at the requirements for Three.js(\*\*支持three.js的桌面浏览器和移动端浏览器\*\*)

+ Setting up a local development environment(\*\*配置three.js的本地环境\*\*)

\* Getting the source code(\*\*获取源码\*\*)

\* Setting up a local web server(\*\*配置一个本地服务器\*\*)

- Using Python to run a web server(\*\*用python启动服务器\*\*)

- Using the npm command from Node.js to run a web server(\*\*用Node.js中的npm命令启动服务器\*\*)

- Running a portable version of Mongoose(\*\*启一个轻便版的数据库Mongoose\*\*)

\* Creating a minimal Three.js web application(\*\*创建一个Three.js的小demo\*\*)

- Creating a scene to contain all the objects(\*\*创建一个的场景\*\*)

- Adding a mesh created from geometry(\*\*添加网格\*\*)

+ What are vertices?(\*\*介绍顶点\*\*)

+ Combining vertices into faces(\*\*将顶点组合成面\*\*)

\* Enhancing the basic scene(\*\*介绍两个组件\*\*)

- Adding easy controls with the dat.GUI library(\*\*dat.GUI\dat.gui.min.js\*\*)

- Add a statistics element to show the frame rate(\*\*stats.min.js\*\*)

\* Debugging the examples in this book(\*\*介绍怎么调试示例\*\*)

- Using console logging for debugging(\*\*console的方法\*\*)

- Looking at objects with breakpoints in Chrome(\*\*加断点\*\*)

\* Summary

- Creating a 3D World Globe and Visualizing Open Data(\*\*创建一个3D地球仪和开放性数据可视化\*\*)

+ Setting up the globe and camera controls(\*\*将相机控件加入globe\*\*)

+ Adding basic textures to the globe(\*\*将基础纹理加入globe\*\*)

+ Adding directional and ambient lighting(\*\*将方向光和环境光加入globe\*\*)

+ Combining with a starry background(\*\*调整相机，完成globe示例\*\*)

+ Improving the look with more advanced textures(\*\*改进纹理\*\*)

\* Using a normal map to simulate elevations(\*\*使用法线贴图来模拟海拔\*\*)

\* Using a specular map to define the reflectivity of an area(\*\*使用高光贴图来定义区域的反射率\*\*)

+ Adding 2D information using HTML canvas as a texture(\*\*添加2D信息--使用HTML画布作为纹理\*\*)

+ Summary

- Navigate around a Randomly Generated Maze(\*\*在随机生成的迷宫中导航\*\*)

+ The result we're aiming for in this chapter(\*\*将demo成果先展示一下\*\*)

+ Creating the maze(\*\*创建迷宫\*\*)

\* Generating a maze layout(\*\*生成迷宫布局\*\*)

\* Converting the layout to a 3D set of objects(\*\*将布局转换为一组3D对象\*\*)

+ Animating the cube(\*\*给cube添加动画\*\*)

\* The standard Three.js rotation behavior(\*\*使用Three.js自带的旋转方法rotation\*\*)

\* Creating an edge rotation using matrix-based transformation (\*\*使用基于矩阵的转换创建边缘旋转\*\*)

\* Using Tween.js to add an animation(\*\*使用Tween.js插件添加动画\*\*)

+ Setting up collision detection(\*\*碰撞检测装置\*\*)

\* Selecting objects(\*\*选中物体\*\*)

\* Detecting collisions(\*\*检测碰撞\*\*)

+ Adding textures and improving lighting(\*\*增加纹理和光源\*\*)

\* Adding a repeating texture(\*\*添加反复使用的纹理\*\*)

\* Setting up the light sources(\*\*设置光源\*\*)

+ Adding trackball and keyboard controls(\*\*添加轨迹球和键盘控件\*\*)

\* Adding trackball controls to the camera(\*\*添加轨迹球\*\*)

\* Configuring keyboard controls(\*\*键盘控件\*\*)

+ Summary

- Visualizing Audio Data with a Particle System(\*\*用粒子系统可视化音频数据\*\*)

+ Visualizing the audio volume(\*\*可视化音频音量\*\*)

\* Setting up the HTML5 Web Audio API (\*\*设置HTML5 Web音频API\*\*)

\* Creating a particle system (\*\*创建粒子系统\*\*)

\* Playing a sound and animating the particle system (\*\*播放声音并使粒子系统动画化\*\*)

+ Creating a particle system by hand (\*\*手工创建粒子系统\*\*)

\* Web Audio's configuration and the render loop (\*\*Web音频的配置和渲染循环\*\*)

\* Creating waves with a custom geometry (\*\*创建具有自定义几何形状的波\*\*)

+ Customizing colors of individual particles (\*\*定制单个粒子的颜色\*\*)

\* Coloring individual particles (\*\*着色单个粒子\*\*)

\* Coloring the particles based on the amplitude (\*\*根据振幅给粒子上色\*\*)

+ Combining dynamic colors to create advanced visualizations (\*\*结合动态颜色创建高级可视化\*\*)

\* Setting up the initial particle system (\*\*建立初始粒子系统\*\*)

\* Calculating volumes for each range (\*\*计算每个范围的容量\*\*)

\* Determining particles that need to be updated and setting the height and color of an individual particle (\*\*确定需要更新的粒子，并设置单个粒子的高度和颜色\*\*)

+ Summary

- Programmatic Geometries (\*\*程序化的几何图形\*\*)

+ Creating a 3D terrain from scratch (\*\*重新创建一个3D地形\*\*)

\* Generating a terrain with Math.random() (\*\*使用Math.random()生成地形\*\*)

\* Generating a terrain with a perlin noise (\*\*使用柏林杂点perlin.js生成地形\*\*)

\* Adding a texture (\*\*添加材质\*\*)

\* Creating a JavaScript object with constructor (\*\*使用构造函数创建JavaScript对象\*\*)

+ Creating a city from scratch (\*\*从零开始创建城市\*\*)

+ Creating parametric trees (\*\*创建参数化树\*\*)

+ Summary

- Combining HTML and Three.js with CSS3DRenderer

+ Setting up a CSS3DRenderer skeleton (\*\*设置CSS3DRenderer框架\*\*)

+ Creating an interactive 3D Google Maps cube (\*\*创建一个交互式3D谷歌地图立方体\*\*)

\* Displaying a part of Google Maps using HTML (\*\*使用HTML显示部分谷歌地图\*\*)

\* Positioning and rotating the element (\*\*放置和旋转元素\*\*)

+ Animating HTML elements with TweenJS (\*\*使用TweenJS让HTML元素动起来\*\*)

\* Using images as the input (\*\*使用图片作为输出\*\*)

\* Setting up the animations (\*\*设置动画\*\*)

- Determining the target position and rotation(\*\*确定目标位置和旋转\*\*)

- Configuring TweenJS to run the animation(\*\*配置TweenJS以运行动画\*\*)

+ Creating a parametric terrain using CSS sprites(\*\*使用CSS精灵创建参数化地形\*\*)

\* Creating a 3D terrain using sprites(\*\*使用精灵创建3D地形\*\*)

\* Animating the terrain with TweenJS(\*\*使用TweenJS动画地形\*\*)

+ Summary

- Loading and Animating External Models Using Blender(\*\*使用Blender加载和动画外部模型\*\*)

+ Installing Blender and the Three.js plugin(\*\*安装Blender和Three.js插件\*\*)

\* Downloading and installing Blender(\*\*下载和安装Blender\*\*)

\* Installing the Three.js plugin(\*\*安装Three.js插件\*\*)

\* Enabling the Three.js plugin(\*\*授权Three.js插件\*\*)

+ Exporting a model from Blender and showing it in Three.js(\*\*从Blender导出一个模型，用Three.js显示\*\*)

\* Exporting the model(\*\*导出模型\*\*)

\* Loading the model and showing it in Three.js(\*\*加载模型并以Three.js显示\*\*)

+ Using Blender's predefined materials in Three.js(\*\*在Three.js中使用Blender预定义的素材\*\*)

\* Setting up a Blender material(\*\*设置Blender的材质\*\*)

\* Setting up UV mapping in Blender(\*\*在Blender中设置UV映射\*\*)

\* Exporting and rendering in Three.js(\*\*导出和渲染在Three.js\*\*)

+ Working with skeletal-based animations in Three.js(\*\*在Three.js中使用基于骨骼的动画\*\*)

\* Exploring the model and exporting it to Three.js(\*\*探索模型并将其导出到Three.js\*\*)

\* Loading and animating the model in Three.js(\*\*在Three.js中加载和动画模型\*\*)

+ Working with morph-based animations in Three.js(\*\*在Three.js中使用基于morph的动画\*\*)

\* Exploring the model and exporting it to Three.js(\*\*探索模型并将其导出到Three.js\*\*)

\* Loading and animating the model in Three.js(\*\*在Three.js中加载和动画模型\*\*)

+ Summary

##three.js essentials大纲总结

- 第一章介绍three.js

+ 什么是Three.js，Three.js开发需要什么

+ 访问本书中讨论的示例的源代码

+ 设置一个本地环境来使用这些示例进行实验

+ 创建一个最小的Three.js场景

+ 使用stats、control、progress辅助库扩展场景

- 第二章用3D渲染地球仪解释一些Three.js核心内容

+ Three.js提供了许多简单的方法来控制相机。展示了如何使用轨道摄像机控件。

+ 通过使用纹理，快速增强渲染的简单网格的外观。展示了如何使用简单和高级的纹理将一个简单的球体转换成一个看起来像地球的物体。

+ 深入研究灯光。向场景添加环境光和方向光。

+ 如果在一个场景中添加很多网格时，效果会受到影响。Three.js可以将多个几何图形与单个几何图形组合在一起，以提高性能。

+ 大多数例子都使用透视相机。除此之外，Three.js还提供了一个正交相机。

+ 渲染场景的一般方法是在渲染器上调用render()函数。然而，也有一种不同的方法，通过它我们可以创建各种渲染或效果步骤，并将它们组合在一起。本章还展示了如何使用这种替代呈现方法进行混淆和使用

- 第三章创建一个简单的游戏来解释Three.js提供的其他一些特性。更具体地说，创建了一个随机的3D迷宫，在迷宫中，使用键盘上的箭头键来导航一个滚动的立方体。

+ 旋转和移动物体周围的场景有两种方法，一是通过标准Three.js属性旋转和平移对象，二是使用矩阵转换。

+ 聚光灯光源。这个光源只发出一束光。

+ 解释了一种Three.js的JS库--Tween.js来创建更高级动画的简单方法。

+ 通过解释如何设置重复纹理来扩展如何在Three.js中使用纹理。

+ 展示了一种使用THREE.Raycaster设置冲突检测的方法。

+ 展示了如何使用THREE.TrackballControls对象轻松移动和平移创建的场景。

+ 解释如何使用键盘来控制场景中的Three.js元素

- 第四章探讨了如何使用THREE.ParticleSystem对象通过显示物体各种方法来可视化声音。

+ THREE.ParticleSystem对象可以以两种不同的方式创建。展示了如何使用空THREE.Geometry对象来创建粒子系统以及如何从现有的粒子系统中创建一个粒子系统。

+ 展示了给你们看如何使粒子系统中的单个粒子产生动画效果。讨论了两种方法:一种是缩放整个几何图形，一个是移动单个顶点。

+ 在创建THREE.ParticleSystem对象时，还需要提供一个材料到系统。这种材料可以消除单个粒子的样子。在本章中，解释了THREE.ParticleSystemMaterial的各种特性。

+ 利用THREE.ParticleSystem，可以对系统中的粒子进行造型;然而默认情况下，所有的粒子都有相同的颜色。展示了如何设置每个粒子的颜色。

+ 本章的最后一部分介绍了Three.js中可用的一些混合模式。混合模式解释了屏幕上像素的颜色如何与它后面像素的颜色进行交互。对于THREE.ParticleSystem对象，混合模式THREE.AdditiveBlending尤其有趣。

- 第五章研究了创建几何图形的不同方法，而不是使用Three.js的内置几何图形。以编程方式创建几何图形。

+ 使用顶点和面从头创建一个几何图形

+ 为生成的几何图形着色;使用了创建的面的颜色属性。

+ 使用纹理时，除了面和顶点，还需要构造一个额外的属性。这叫做UV映射。

+ 使用柏林杂点(perlin.js)，可以创建看起来自然的随机性。使用柏林杂点(perlin.js)创建一个自然的地形。

+ 就像创建几何图形使用随机库一样使用随机库生成纹理。

+ 使用外部库的算法，可以生成创建几何图形所需的信息。使用一个包含生成树的算法的库。

- 第六章 解释了如何使用CSS3DRenderer以及它有哪些特性。之前一直使用THREE.WebGLRenderer对象渲染场景。这个渲染器需要浏览器支持WebGL。大多数桌面浏览器都支持WebGL，但移动端浏览器有点落后，因此Three.js提供了两个不同渲染器使用：CanvasRenderer和CSS3DRenderer，但前者性能不是很好。所以这里介绍后者，后者的缺点是不支持集合形状、材质和灯光

+ 设置一个简单的框架页面，可以将其作为创建基于CSS3DRenderer的可视化的起点。

+ 即使CSS3DRenderer不支持几何图形和网格，仍然可以模拟它。展示了如何创建类几何的对象通过重写THREE.Geometry对象的方法

+ 通过使用three.js和CSS3DRenderer，可以很容易地在3D世界中做动画和移动HTML元素。通过添加动画来扩展前面的示例。

+ 了解如何使用CSS3DRenderer通过使用CSS3DSprite模拟粒子系统。

- 第七章解释了如何将Blender的建模功能与Three.js提供的呈现功能结合起来。

+ 安装Blender并对Three.js插件进行混合，这样模型和动画就可以导出到Three.js。

+ 解释如何在blender完成一个简单的模型,导出使用Three插件,并加载模型和在浏览器中显示它。

+ Three.js支持两种不同的动画:基于骨骼动画和基于形morph动画。Blender也支持这些类型。在本章中，展示了如何在Blender中设置这样的动画，并通过Three.js在浏览器中对其进行动画处理。

+ 对于基于骨骼和基于morph的动画，展示了如何使用Three.js来播放Blender中定义的动画