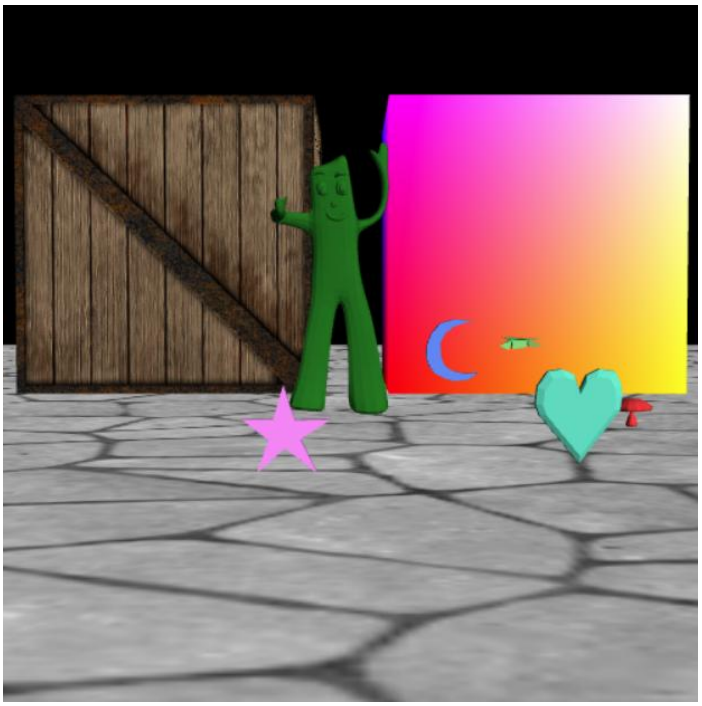

计算机图形学

PROJECT 3

三维场景漫游

时间节点：

发布日期：2024-5-14

截止日期：2024-6-14

TA： NA

Project3 三维场景漫游

目录

1. 项目说明

1.1 项目简介

不同于以往两个 PJ 是让大家从头写代码，本次 PJ3 是将基础代码（见项目根目录下“2-项目基础代码/BaseCode”文件夹）给到了大家，需要大家参照基础代码，在其基础上完成第 2 节任务说明中所列出的任务。所以大家要尽可能的理解基础代码，才能更好的完成本次的 PJ。

本次 PJ 各任务（详情见第 2 节任务说明）相对独立，大家可以根据建议的步骤（详情见 3 实现步骤）逐步骤完成，对于每个任务，都有在各个的任务说明中（见任务说明第 2.1-2.8 小节）标识完成此任务需要学习的《WebGL 编程指南》相关章节及此任务在基础代码中涉及的主要 JS 代码文件，大家可以在学习《WebGL 编程指南》相关章节的同时，在基础代码中找到对应位置，加以理解，完成对应任务。

1.2 项目目的

理解 WebGL 绘制 3D 图形的原理，利用基础代码最终实现 3d 场景效果的正确显示。

1.3 项目知识点

三维相机模型的建立，三维模型的绘制，纹理的使用，添加光照模型，从文件中载入和显示 3D 复杂模型，3 维动画的实现。详情见第 4 节知识点介绍。

1.4 项目使用的语言

Html5, Javascript, WebGL。

1.5 项目开发与测试环境

1) **硬件**：带有支持 OpenGL ES2.0 显卡的计算机。
2) **软件**：支持 HTML5, JavaScript 和 WebGL 的浏览器。
3) **Web 服务器**：WAMP 等集成环境安装包。（可在项目根目录下的“4-浏览器跨域请求解决方案/ Web 服务器”文件夹中找到）

2. 任务说明

本次 PJ 的任务需要在充分理解基础代码（见 2-项目基础代码/BaseCode 文件夹）的基础上，对基础代码进行修改或在其之上添加新的代码，以完成下面第 2.1-2.8 小节列出的任务。

【注】 以下提到的“**初始场景**”为运行“基础代码”中的 3DWalker.html 文件（见 2-项目基础代码/BaseCode/3DWalker.html）可看到的初始的场景效果。

【注】 以下提到的“**配置文件**”即 scene.js 文件（见 2-项目基础代码/BaseCode/scene.js）为最终场景效果的数据配置文件，大家要根据此配置文件中各个模型的数据参数，调整初始场景，或在初始场景中添加新物体和新效果，各数据参数在配置文件中有详细注释说明。

【注】以下提到的所有 JS 文件均放置在基础代码文件夹即项目根目录下的“2-项目基础代码/BaseCode 文件夹”中。

2.1 校准相机姿态

在初始场景中，相机初始是以一种右斜的角度观察场景，效果见图 2.1.1，要求根据场景配置文件中给

到的相机模型参数（CameraPara），进行相机初始姿态的矫正,效果见图 2.1.2 。

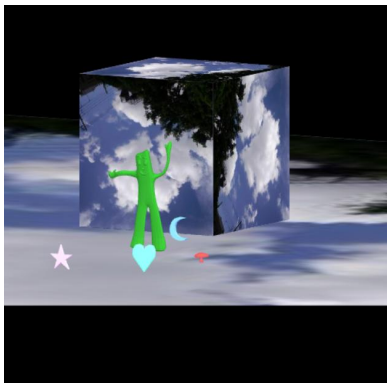

图 2.1.1

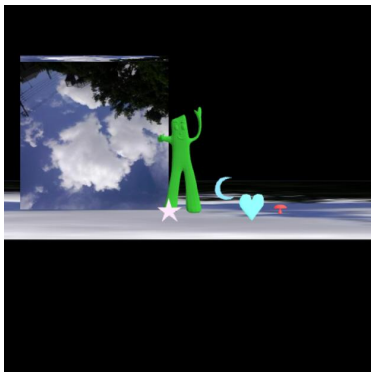

图 2.1.2

此小节任务对应学习资料为《WebGL 编程指南》第七章第 1-10 节，对应基础代码中的 scene.js 和 Camera.js 文件。

2.2 变换投影方式

在初始场景中，相机采用的是正交投影，效果见图 2.2.1，要求根据场景配置文件中给到的相机模型参数（CameraPara），将相机模型改为透视投影，效果见图 2.2.2。

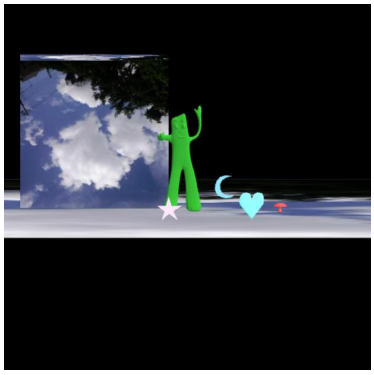

图 2.2.1

图 2.2.2

此小节任务对应学习资料为《WebGL 编程指南》第七章第 11-26 节，对应基础代码中的 scene.js 和 Camera.js 文件。

2.3 绘制简单三维模型

在初始场景中，根据场景配置文件中给到的颜色渐变箱体模型参数（cubeRes），在场景中添加颜色渐变的三维箱体，任务完成前后效果图分别见图 2.3.1 和图 2.3.2。

图 2.3.1

图 2.3.2

此小节任务对应学习资料为《WebGL 编程指南》第七章第 33-37 节内容，**此任务需要同学们自己新建 JS 文件**，从配置文件中读取颜色渐变箱体模型参数（cubeRes），并添加绘制颜色渐变的三维箱体的代码，对应基础代码中的 scene.js 和 3DWalker.js 文件。

2.4 更换纹理

初始场景中对已有的一个箱体模型和地面模型初始化了一个相同的纹理，效果见图 2.4.1。请根据配置文件中给到的纹理箱体的模型参数（boxRes）和地面的模型参数(floorRes)更换相应正确的纹理，效果见图 2.4.2。

图 2.4.1

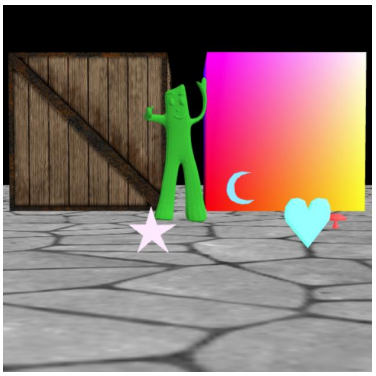

图 2.4.2

此小节任务对应学习资料为《WebGL 编程指南》第五章第 14-29 节内容，对应基础代码中的 scene.js 和 Texture.js 文件。

2.5 读取并绘制复杂模型

初始场景中添加四个复杂模型，效果见图 2.6.1，要求根据配置文件中给到的复杂模型参数列表（ObjectList）中给到的数据，读取 bird.obj 模型文件（见“2-项目基础代码/BaseCode/model/bird.obj”），并将此模型绘制到场景中，效果见图 2.6.2。同学们可**参考已添加的四个复杂模型的代码**实现复杂模型的读取与绘制。

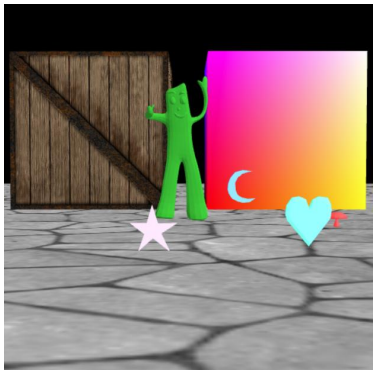

图 2.6.1

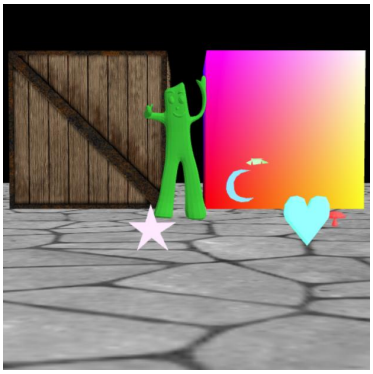

图 2.6.2

此小节任务对应学习资料为《WebGL 编程指南》第十章第 49-54 节内容，对应基础代码中的 3DWalker.js、Object.js 和 objLoader.js 文件。

2.6 实现简单动画

给任务 2.5 新添加的 bird 模型实现一个简单的动画，比如绕着海宝模型不断飞行，高度按照旋转角的正弦值不断变化,具体实现效果不做要求，同学们可以自由发挥，并在文档中描述清楚。动画效果可在最终效果演示视频中进行观看。

此小节任务对应学习资料为《WebGL 编程指南》第四章第 7-14 节内容，动画相关代码需要同学们自行添加。

2.7 实现光照

初始场景中已初始化了一个平行光和环境光照，效果见图 2.7.1，要求将其根据配置文件中提供的平行光的方向（平行光颜色值已固定，大家不需要考虑）和环境光照的颜色值做出修改，效果见图 2.7.2。除此之外需要同学们添加一个**跟随相机移动的点光源**来随时照亮前方的复杂模型，点光源颜色同样在配置文件中读取，这个光源可以通过键盘上的 F 键来开启，开启后效果见图 2.7.3。

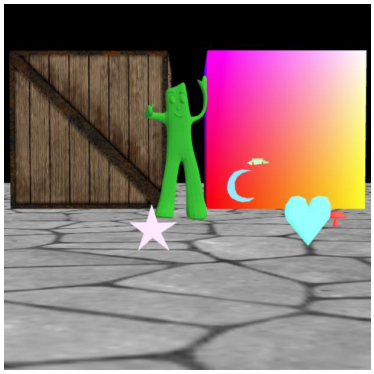

图 2.7.1

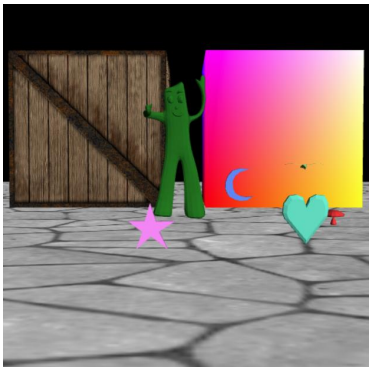

图 2.7.2

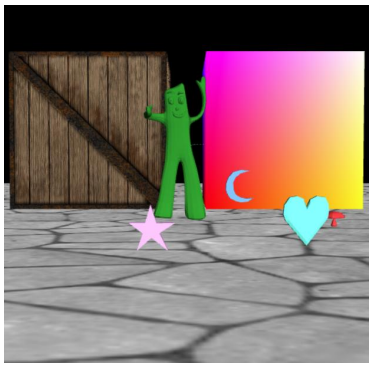

图 2.7.3

此小节任务对应学习资料为《WebGL 编程指南》第八章 1-17 节内容，对应基础代码中的 scene.js、3DWalker.js、Object.js 和 Keyboard.js 文件。

2.8 实现场景漫游

初始场景中，通过键盘交互的方式已实现：按 A/D 键来控制摄像机左右移动（移动的速度为 30 个单位/秒），按 J/L 键来使摄像机以 y 轴为轴左右旋转（旋转速度为 60 角度/秒）。要求在此基础上，实现按 W/S 键来控制摄像机前后移动，按 I/K 键来使摄像机以 x 轴为上下旋转。

此任务前后效果演示请观看初始场景演示视频和最终效果演示视频。

此小节任务对应学习资料为《WebGL 编程指南》第七章第 1-10 节和 27-30 节内容，对应基础代码中的 scene.js、Camera.js 和 Keyboard.js 文件。

3. 实现步骤（建议）

3.1 了解项目资料

1) 观看**项目文件介绍视频**(见项目根目录下“0-项目文件介绍视频”)，了解项目资料中所包含的文件及其作用。
2) 阅读项目说明文档（即此文档），配合项目说明文档的讲解视频（见项目根目录下“1-项目说明资料/CG_Project3 文档讲解视频”）了解项目目的，项目任务，实现步骤及所需知识点。
3) 阅读 PJ3 给分点文档，了解项目完成每部分的给分点。

3.2 准备项目开发与测试环境

1) 下载并安装支持 HTML5 和 WebGL 的浏览器（最新版本的 Firefox，chrome 和 Safari 等）
2) 使用安装好的浏览器，运行 testBrowser.html（见项目根目录下“5-浏览器测试文件/testBrowser.html”）确认浏览器支持 HTML5 以及是否已经开启了对 WebGL 的支持，如果没有，请在浏览器设置选项中开启对 WebGL 的支持。

3.3 解决浏览器跨域请求问题

观看 **Web 服务器使用教程**视频(见项目根目录下“4-浏览器跨域请求解决方案/Web 服务器使用教程”), 并配置可做跨域请求的浏览器运行环境。

【注】由于浏览器的安全策略，不允许 JS 跨域请求，因此需要大家使用 Web 服务器来支持本次 Project。具体为下载并安装 WAMP 集成环境，在其 www 目录中放置本次 Project 的 html，css，js 和其它各种需要用到的库和文件。在本地浏览器中通过 localhost 来访问 Web 页面。Windows 用户建议直接使用 Web 服务器的。Mac OS 用户浏览项目提供的解决方法的网站进行学习并操作。

3.4 理解基础代码

1） 观看**基础代码讲解视频**(见项目根目录下“2-项目基础代码/基础代码讲解视频”)。
2） 结合**基础代码讲解视频**和**第 5 节**的项目完整逻辑图，阅读并理解基础代码。
3） 结合知识点第 **4.7** 小节，阅读《WebGL 编程指南》**第十章第 27-29 节**内容，了解如何优雅的切换 shader，有助于帮助理解基础代码中的初始代码。

3.5 结合教材完成项目任务

1) 结合知识点**第 4.1.1 和 4.1.2** 小节，阅读《WebGL 编程指南》**第七章第 1-10 节**内容，学习如何根据相机姿态参数在场景中调整相机姿态， 并完成**任务 2.1** 校准相机姿态。
2) 结合知识点第 **4.1.3** 小节，阅读《WebGL 编程指南》**第七章第 11-26 节**内容，学习相机两种主要的投影模型，学习如何根据相机模型参数创建投影矩阵，并完成**任务 2.2** 变换投影方式。
3) 结合知识点第 **4.1.4** 小节，阅读《WebGL 编程指南》**第七章第 33-37 节**内容，学习如何构建模

型视图投影矩阵，学习如何通过索引方式绘制三维立体图形，并完成**任务 2.3** 绘制简单三维模型。

4) 结合知识点第 **4.2** 小节，阅读《WebGL 编程指南》**第五章第 14-29 节**内容，学习如何载入一张图片，并且将其映射到一个矩形上，并完成**任务 2.4** 更换纹理。
5) 结合知识点第 **4.3** 小节，阅读《WebGL 编程指南》**第十章第 49-54 节**的内容，学习如何从 obj 文件中载入 3D 模型并绘制，完成**任务 2.5** 读取并绘制复杂模型。
6) 结合知识点第 **4.4** 小节，阅读《WebGL 编程指南》**第四章第 17-14 节**的内容，结合 PJ2 中的制作动画的经验，完成**任务 2.6** 实现简单动画。
7) 结合知识点第 **4.5** 小节，阅读《WebGL 编程指南》**第八章 1-17 节**的内容，学会如何向场景中添加不同种类的光源（环境光，方向光，点光源），并完成**任务 2.7** 实现光照
8) 结合知识点第 **4.6** 小节，阅读《WebGL 编程指南》**第七章第 1-10 节和 27-30 节**内容，并结合第 3.5 小节中的步骤 1）和步骤 2）的学习，完成**任务 2.8** 实现场景漫游。
9) 学有余力的同学可以尝试其他高级话题，比如 Phone shading，雾，阴影等。
10) 编写你的项目文档，内容需包括但不限于：你的项目目录及文件说明，开发及运行环境，运行及使用方法，你的项目中的亮点，开发过程中遇到的问题（以及你的解决办法），项目仍然或者可能存在的缺陷（以及你的思考）。
11) 完成后将你的代码和文档放置在同一文件夹中，文件夹名称为“[学号]_[姓名]”，使用 zip 格式压缩后上传至 elearning。

4. 知识点介绍

4.1 3D 场景

4.1.1 三维立体坐标系

本次 PJ 所绘制的物体都是 3 维立体空间中的，因此，我们所使用的坐标系也是空间中的三维坐标系，有三个相互垂直的坐标轴，分别是 X 轴，Y 轴和 Z 轴。相应的，模型上的每一个顶点都有三个坐标分量（x, y, z）。如图 4.1.1 所示：

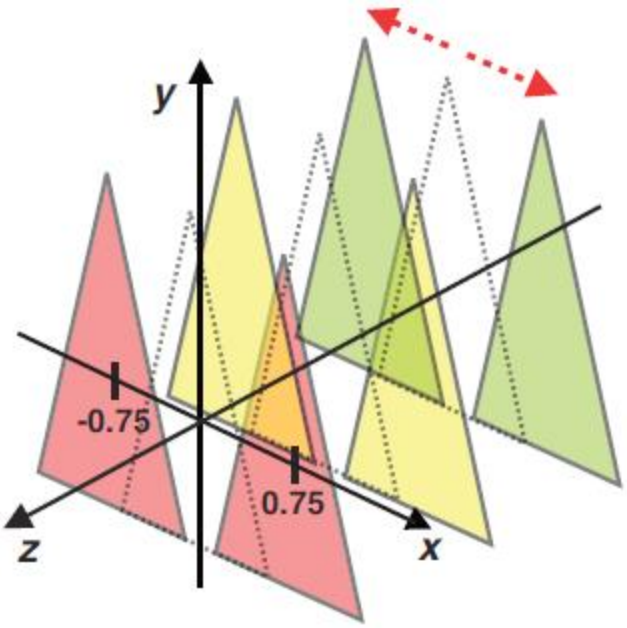

图 4.1.1

【注】此知识点对应《WebGL 编程指南》第七章第 1 节。

4.1.2 3D 相机模型 – 相机位置与姿态

对于 3D 场景的展现，我们都是在场景中的某一位置，对着某个方向，使用不同投影的方式绘制一幅 2D 的画面。因此，我们首先要通过摄像机的位置，朝向和上方向等信息来描述相机的姿态。这里我们也需要一个参数化模型，如图 4.1.2 所示：

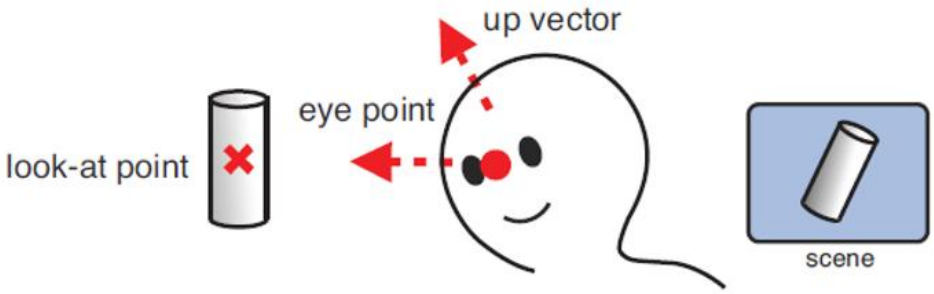

图 4.1.2

将相机姿态参数化后，就会可以根据相机的姿态参数（摄像机的位置，朝向和上方向），构建相机的视图矩阵。在实际的 WebGL 变编程中，我们可以调用 cuon-matrix.js 提供的 Matrix4.setLookAt()函数，并给函数传入相应的参数来进行视图矩阵的构建。

【注】此知识点对应《WebGL 编程指南》第七章第 1-10 节，和任务第 2.1 小节校准相机姿态。

4.1.3 3D 相机模型 – 可视空间

可视空间决定了相机可以观察到的可视范围，WebGL 只显示可视范围内的区域，即只有在可视范围内的物体才会被绘制。

常用的可视空间有两种：

1) 盒状空间，由正交投影产生。正交投影的好处是可以方便的比较场景中物体的大小，这是因为物体看上去的大小与其所在位置没有关系。这种相机模型如下图所示，需要六个参数来描述，分别为近裁面的上、下、左、右边界及近裁面和远裁面的位置。如图 4.1.3.1 所示：

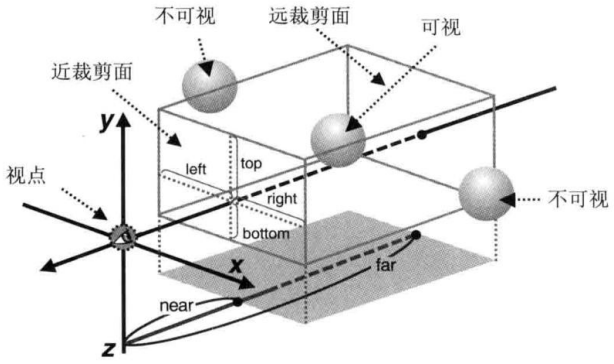

图 4.1.3.1

通过调用 cuon-matrix.js 提供的 Matrix4.setOrtho()函数，并传入相机的正交投影参数来进行正交投影矩阵的构建。

2) 四棱锥可视空间，由透视投影产生。无论是用人眼看物体，还是摄影，写实油画等，其展现的画面中的物体都是近大远小的。在计算机三维立体绘制中，这样的成像方式我们称之为透视投影。它需要一种特定的相机模型，如下图所示。我们需要四个参数来描述一个透视投影相机远/近成像距离，画面宽高比，垂直张角。如图 4.1.3.2 所示：

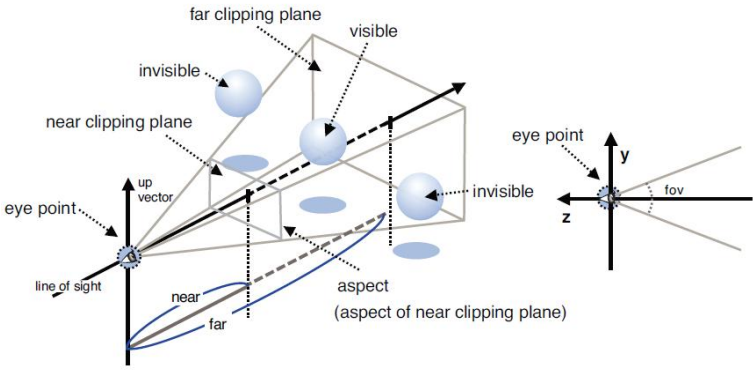

图 4.1.3.2

通过调用 cuon-matrix.js 提供的 Matrix4.setPerspective()函数，并传入相机的透视投影参数来进行透视投影矩阵的构建。

【注】此知识点对应《WebGL 编程指南》第七章第 11-26 节，和任务第 2.2 小节变换投影方式。

4.1.4 从 2D 到 3D

在 Project 2 当中，大家已经能够在 2D 平面上绘制图形了，而且能制作简单的动画。在制作动画的时候，我们用到了矩阵变换来改变物体的坐标。在 3D 场景中绘制物体，其本质同样也是坐标变换，只不过其过程较 2D 变换而言复杂一些。我们需要如下几步：

1) 将物体按照其位置，旋转，大小等描述，将其自身坐标，转换到世界坐标系中
2) 将三维世界中的物体坐标，转换到相机坐标系中
3) 将相机坐标系中的物体坐标，转换到相机成像平面中
4) 将相机成像平面上投影出的画面，映射到最终显示的画布当中

要完成上述坐标变换，除了第 4.1.2 和 4,.1.3 小节提到的视图矩阵和投影矩阵，还需要对模型变换的模型矩阵。之后将物体顶点坐标与这些矩阵依次相乘，绘制，即可得到最终的画面。而我们也可以先将上述模型矩阵(modelMatrix)、视图矩阵(viewMatrix)和投影矩阵(projMatrix)乘起来构成一个模型视图投影矩阵，再统一和模型顶点相乘。在实际 WebGL 编程中通过以下语句来进行模型视图投影矩阵的构建:

```
mvpMatrix.set(projMatrix). multiply (viewMatrix).multiply(modelMatrix);
```

【注】此知识点对应《WebGL 编程指南》第七章第 33-37 节，和任务第 2.3 小节绘制简单三维模型。

4.2 纹理（贴图）

在室内装修的时候，很多人会觉得纯色的墙面过于单调，因此会选择在墙上贴一些墙纸来进行修饰。同样，对于 3D 场景中的物体，我们需要使其具有更逼真的细节，和更丰富的色彩。因此，我们需要使用纹理来赋予物体更多的表现力，如图 4.2 所示。

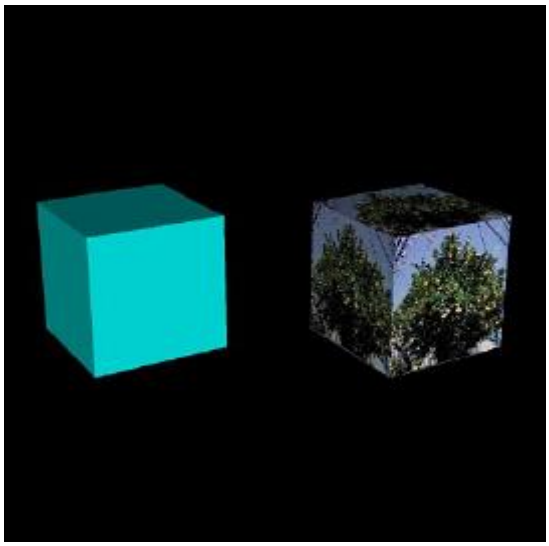

图 4.2

【注】此知识点对应《WebGL 编程指南》第五章第 14-29 节，和任务第 2.4 小节更换纹理。

4.3 从 obj 文件中读取复杂的物体模型

在实际应用（比如游戏）的 3D 场景中，我们不能只绘制一些简单的物体诸如矩形平面，正方体，球体等，而是需要一些更加复杂的立体模型。这些立体模型通常由艺术设计人员使用诸如 Maya，3DMax 等三维建模软件手工制作的，并以一些特定的格式存储。对于这些比较复杂的物体，我们通常需要从文件中读取。本次 PJ 当中就具有这样的物体，比如封面图中的月亮，心形，海宝等。大家可以通过调用 readOBJFile()函数从文件中载入三维模型，拿到模型信息后通过 parse()函数对模型数据进行解析，然后传到 WebGL 系统中实现复杂模型的绘制。

【注】此知识点对应《WebGL 编程指南》第十章第 49-54 节，和任务第 2.5 小节读取绘制复杂模型。

4.4 动画

在 PJ2 中，同学们已经尝试过在二维场景中实现动画效果。在三维场景中，仍然可以实现三维空间中的动画效果，比如某一个物体绕着另一个物体旋转，其本质和二维动画是一样的，都是通过在间隔时间内不断的对模型的顶点进行变换（如平移，旋转），并重新绘制模型实现的，而重绘方式仍然是通过 requestAnimationFrame 函数循环申请调用场景绘制函数实现。区别在于，三维场景中模型每个顶点多了 z 轴坐标，且在模型运动过程中要考虑模型间的遮挡关系。

【注】此知识点对应《WebGL 编程指南》第四章第 7-14 节，和任务第 2.6 小节实现简单动画。

4.5 光照模型

为了模拟真实世界中的效果，我们还需要在场景中添加光源，来使物体产生明暗等效果。场景中添加了由海宝正面射向海宝背面的**平行光**线。所以，从正面看物体（见图 4.5.1）是明亮且色彩鲜艳的，而从背面看（见图 4.5.2），物体非常暗淡。另一方面，从背面看物体，并非完全漆黑，而是略微有一点色彩，这是**环境光**的影响。

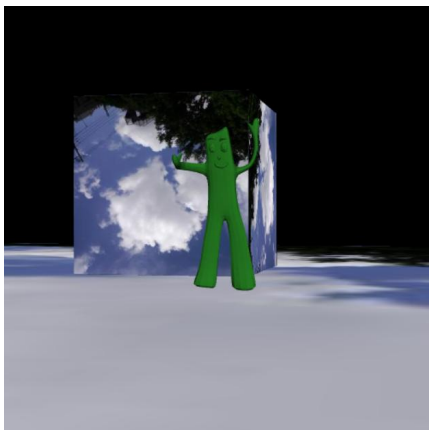

图 4.5.1

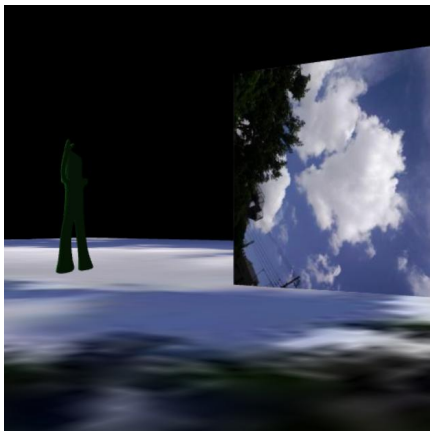

图 4.5.2

【注】此知识点对应《WebGL 编程指南》第八章 1-17 节，和任务第 2.7 小节实现光照。

4.6 场景漫游的实现方法

场景漫游，本质上是通过不断的移动摄像机的位置和朝向来实现的。关于动画和速度控制，方法可以和 Project 2 相同。场景漫游的控制方法建议，通过键盘事件改变某个（些）全局变量 g 的值（表示状态的时候用布尔值或者 0/1），同时通过包含了 requestAnimationFrame 的 tick 函数（Project2 中有提

到）来不断重新绘制场景，在绘制函数中读取全局变量 g，并根据 g 的值（布尔）来决定是否改变摄像机的坐标。

【注】此知识点对应《WebGL 编程指南》第七章第 1-10 节和 27-30 节，和任务第 2.8 小节实现场景漫游。

4.7 优雅的 Shader 切换

本次 PJ 在绘制地面和箱子的时候，使用纹理来为物体着色，而在绘制其它物体的时候，使用光照模型。因此在一轮绘制当中，需要在两个 Shader Program 之间进行切换，先用一个 shader 绘制地面和箱子，再用另一个 Shader 根据光照绘制物体。

【注】此知识点对应《WebGL 编程指南》第十章第 27-29 节

5. 项目完整逻辑图