



河南理工大学
Henan Polytechnic University

教学上机实验报告

课 程 名 称: _____

任课教师姓名: _____

学 生 学 号: _____

学 生 姓 名: _____

学生专业班级: _____

_____ ~ _____ 学年第 学期

河南理工大学

教学上机实验报告评价分值标准

序号	评价指标	分值	评价等级及参考分值					评价分
			优	良	中	合格	差	
1	实验报告书写规范、字迹工整认真，内容完整充实	20	20	17	15	13	6	
2	实验过程叙述详细、概念正确，语言表达准确，结构严谨，条理清楚，逻辑性强，自己努力完成，没有抄袭	30	30	26	23	20	10	
3	对实验过程中存在的问题分析详细透彻、全面、规范，结合实验内容，有自己的个人见解和想法，给出解决方法	30	30	26	23	20	10	
4	实验结果、分析和结论正确	20	20	17	15	13	6	
总得分								

签名（签章）：

日期： 年 月 日

说明：

学生应按要求按时参加上机，完成任课教师布置的上机任务。学生每次上机结束后，应该及时按照实验要求和上机操作情况认真填写上机实验报告，并在课程结束后以班级为单位按照学号从小到大顺序排序后上交给任课教师。

河南理工大学上机实验报告

2020—2021 学年

第 一 学期

上机时间_____

专业班级 计实验 1801

学号 311809000608

姓名 王荣胜

实验课程名称：虚拟现实技术

实验目的和要求：

本章要求学生熟练掌握三维建模工具 3ds Max 的常用操作,包括内置几何体建模、常用二维图形建模、常用复合建模方法,材质与贴图的设置方法,了解灯光与摄影机的简单操作及 3ds Max 动画制作的基本流程。(以 Windows 7 系统下安装的 3ds Max 2010 32bit 版本为软件环境。)

实验项目名称：

实验过程及代码：

一：制作沙发靠背

操作步骤如下。

- (1) 创建新文件,单击选择“创建”面板标签。
- (2) 单击“几何体”按钮。
- (3)在下拉列表中选择“扩展基本体”选项。
- (4)单击“切角长方体”按钮。
- (5)在顶视图中采用鼠标左键拖曳→放开在键向上 D 明平向上移可→单击的步骤,创建切角长方体。
- (6)在“创建”或“修改”面板中调整模型的参数:长度——300,宽度 300,高度——8,圆角——30,长度分段——5,宽度分段——6,高度分段——1,圆角分段——3。
- (7)在“修改”面板的“修改器列表”中选择“FFD(长方体)”选项。
- (8)单击“FFD(长方体)4×4×4”右侧的加只地知 T 版水曰城选择“控制点”层级。
- (9)在左视图中,用 Ctrl 键配合鼠标选择 4 个角的点(实际上是选择了 16 个控制点)。单击“选择并均匀缩放”按钮,光标指向 y 轴,向下拖动鼠标。

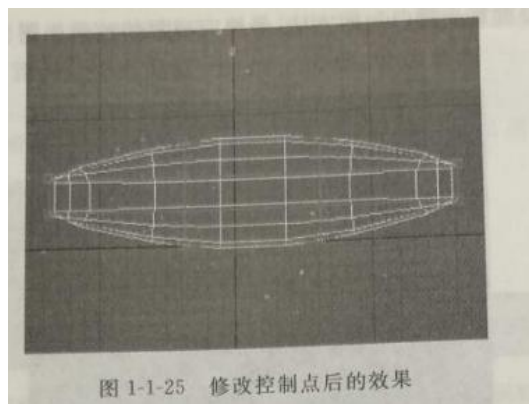


图 1-1-25 修改控制点后的效果

- (10)在顶视图中,用 Ctrl 键配合鼠标选择 4 个角的点(实际上是选择了 8 个控制点)。单击“选择

并均匀缩放”按钮,光标指向工 y 缩放区域,向上拖动鼠标。

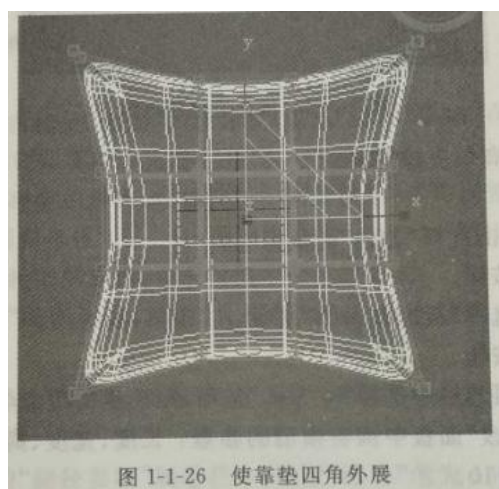


图 1-1-26 使靠垫四角外展

- (11)在前视图中拖动鼠标,选择上边界中间两个控制点,单击“移动工具”按钮,光标指向 y 轴,向上拖动鼠标,使上边线圆滑。使用同样的方法,使下边界向下凸出,边界圆滑。调整后前视图中效果如下所示。

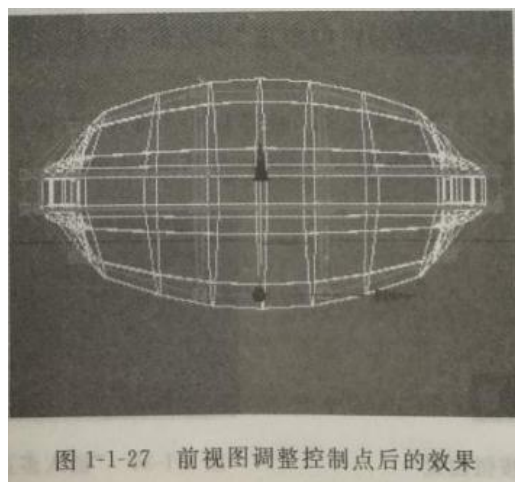


图 1-1-27 前视图调整控制点后的效果

- (12)在左视图中重复步骤(11)的方法最终靠垫完成的效果图如下所示。
- (13)保存文件。

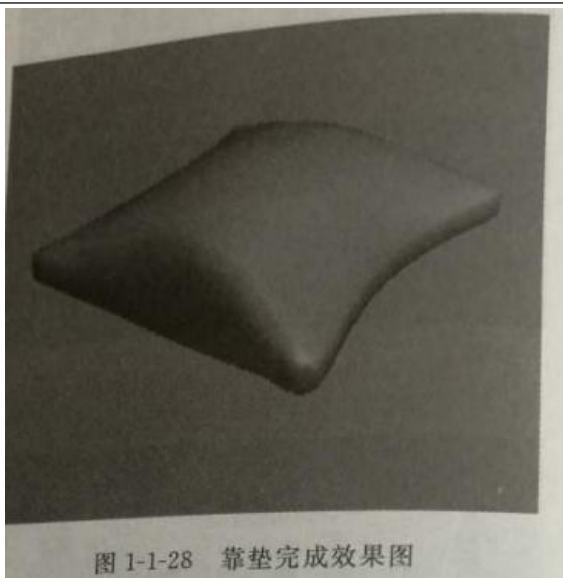


图 1-1-28 靠垫完成效果图

二：制作高尔夫球模型

操作步骤如下。

- (1)创建新文件,单击选择“创建”面板标签。
- (2)单击“几何体”按钮。
- (3)在下拉列表中选择“标准基本体”选项。
- (4)单击“长方体”按钮。
- (5)在“创建方法”展卷栏中选择“立方体”选项,在任意视图中单击创建立方体。
- (6)在“创建”或“修改”面板中调整模型的参数:长度、宽度、高度均为 100,长度分段宽度分段、高度分段均为 10。
- (7)在“创建”或“修改”面板中单击模型名字右侧的色块按钮。弹出“对象颜色”对话框,将对象的颜色修改为白色。
- (8)为立方体添加“球形化”修改器,在球形化的立方体上右击,选择快捷菜单命令转换为”|“转换为可编辑多边形”。
- (9)在“修改”面板的“选择”展卷栏中单击“多边形”按钮,在任意视图中拖动鼠标框选中所有的多边形。
- (10)单击“修改”面板中的“编辑多边形”展卷栏中“插入”右侧的折叠按钮,弹出“插入多边形”对话框。参数设置为:插入类型—按多边形,插入量—0.5。
- (11)单击“修改”面板中“编辑多边形”展卷栏中“挤出”右侧的折叠按钮,弹出“挤出多边形”对话框。参数设置为:挤出类型——按多边形,挤出高度——0.5。
- (12)添加“涡轮平滑”修改器,将“迭代次数”设为 2。

三：制作“圆桌布”模型

操作步骤如下。

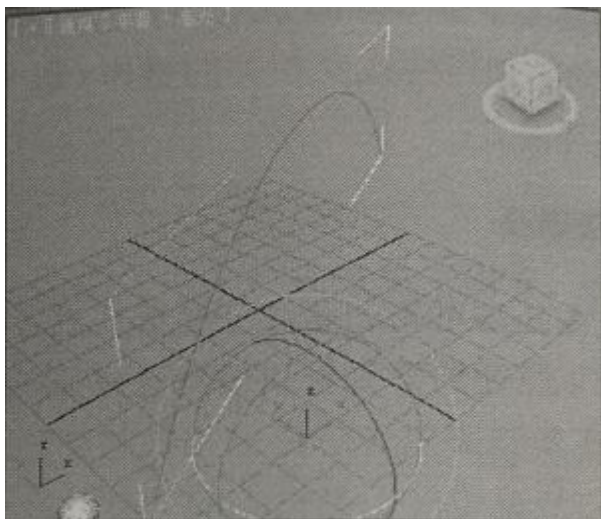
- (1)创建新文件,单击选择“创建”面板标签,单击“图形”按钮,在下拉列表中选择“样条线”选项,单击“圆”按钮,在顶视图中拖动创建圆,半径为 100。
- (2)使用与步骤(1)相似的步骤,在顶视图中创建星形,参数设置为:半径 1——100,半径 2——85,点——14,圆角半径 1——10,圆角半径 2——10。
- (3)使用与步骤(1)相似的步骤,在前视图中创建样条线。
- (4)单击选择“创建”面板标签,单击“几何体”按钮,在下拉列表中选择“复合对象”选项,单击“放样”按钮。
- (5)此时样条线处于选中状态,单击“创建方法”展卷栏中的“获取图形”按钮,选择视图中的圆。
- (6)在“修改”面板的“路径参数”展卷栏中,设置“路径”值为 100。
- (7)再次单击“创建方法”展卷栏中的“获取图形”按钮,在任意视图中选择星形。
- (8)保存文件。

五：约束动画

约束动画是将对象的运动变化限制在某个特定的范围内。3ds Max 提供了多种约束方式,本节以路径约束为例讲解约束动画的制作方法。

制作高尔夫球沿抛物线移动动画。

- (1)单击快速访问工具栏中的“打开文件”按钮,在弹出的“打开文件”对话框中找到文件“实验 2.8max”,选中该文件,单击“打开”按钮。
- (2)单击“创建”按钮,打开“创建”面板,单击“图形”按钮,在下拉列表中选择“样条线”选项,在“对象类型”卷展栏中单击“弧”按钮,在前视图中拖动鼠标指针创建弧线,在“参数”卷展栏中设置“半径”为 3200,“从”为 36,“到”为 145。
- (3)使用“选择并旋转”工具,在上视图中沿 Z 轴方向适当旋转弧线,使其大致符合高尔夫球抛出轨迹的角度。
- (4)单击选中高尔夫球。
- (5)选择“动画”→“约束”→“路径约束”命令,单击弧线,指定高尔夫球的约束路径为该弧线。
- (6)单击“播放动画”按钮,查看动画效果。
- (7)保存文件。

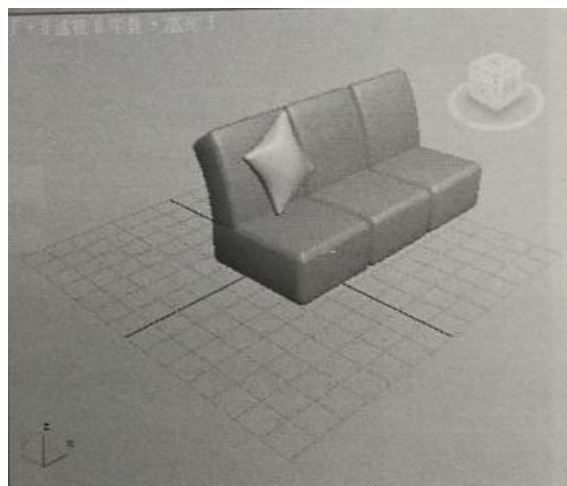


六：制作沙发摆放靠垫动画

- (1)创建新文件,单击“创建”按钮,打开“创建”面板。
- (2)单击“几何体”按钮。
- (3)在下拉列表中选择“扩展基本体”选项。
- (4)在“对象类型”卷展栏中单击“切角长方体”按钮。
- (5)在上视图中创建切角长方体,按住鼠标左键并拖动—释放鼠标—向上移动—单击—向上移动—单击,完成创建。
- (6)在“创建”或“修改”卷展栏中设置“长度”为45,“宽度”为35,“高度”为20,“圆角”为3,该切角长方体的默认名称为“Chamfer-Box01”。
- (7)单击“选择并移动”按钮,按 Shift 键的同时沿上视图正向拖动“Chamfer-Box01”,克隆出它的两个实例,分别命名为“Chamfer-Box02”“Chamfer-Box03”。
- (8)选择“Chamfer-Box02”,单击对齐按钮,选择“Chamfer-Box01”,弹出“对其当前选择”对话框,在“对其位置”选项组中设置参数,使其左侧紧贴“Chamfer-Box01”右侧。
- (9)参照步骤(8)调整“Chamfer-Box03”的位置,使其左侧贴紧“ChamferBox-02”右侧。至此制作完成了沙发的坐垫。
- (10)在上视图中再创建新的切角长方体,在“参数”卷展栏中设置“长度”为15,“宽度”为35,“高度”为35,“圆角”为3,参照步骤(7)~(9)步,制作并调整好沙发的靠背。
- (11)选中作为靠背的3个切角长方体,在“修改”面板中将“高度分段”设置为8。
- (12)单击“修改”按钮,打开“修改”面板,单击“修改器列表”的下拉按钮,选择“弯曲”修改

器。

- (13)在“参数”卷展栏中设置“角度”为35,“方向”为90。
- (14)拖动鼠标指针,选中沙发的所有组件,选择“组”→“成组”命令,弹出“组”对话框。修改组名为“沙发”,单击“确定”按钮。
- (15)单击应用程序按钮,在弹出的下拉菜单中选择“导入”→“合并”命令,弹出“合并文件”对话框,找到靠垫模型文件,选中,单击“打开”按钮。弹出“合并”对话框,单击合并对象“Chamfer-Box01”,点击确定。
- (16)由于当前场景中包含对象名称“Chamfer-Box01”,系统弹出“重复名称”对话框,输入新名称“靠垫”,单击“自动重命名”按钮,该对象被重命名为“靠垫01”。
- (17)使用“选择并缩放”工具,调整靠垫的大小,使其与沙发匹配。
- (18)使用“选择并缩放”工具和“选择并旋转”工具调整靠垫的大小与角度,使其与沙发匹配。



- (19)单击“创建”按钮,打开“创建”面板。
- (20)单击“摄影机”按钮。
- (21)在下拉列表中选择“标准”选项。
- (22)在“对象类型”卷展栏中单击“目标”按钮。
- (23)在上视图中拖动鼠标指针创建目标摄影机,调整摄影机及目标位置,使透视图成为活动视口,按快捷键C,转换为摄影机视图。
- (24)再复制两个靠垫,并适当调整这两个靠垫的角度。将3个靠垫均摆放在摄像机拍摄范围之外。
- (25)将时间滑块移动到第0帧,单击“设置关键点”按钮关时点,启动设置关键点模式。
- (26)调整一个靠垫的位置,然后单击“关键帧”按钮,添加第一个关键帧。

(27)将时间滑块移动到第 30 帧，调整第一个靠垫的位置到沙发上，然后单击“关键帧”按钮，添加第二个关键帧。

(28)将时间滑块移动到第 60 帧，调整第二个靠垫的位置到沙发上，然后单击“关键帧”按钮，添加第三个关键帧。

(29)重复上述步骤在第 90 帧添加第四个关键帧。

(30)单击“时间配置”按钮，设置“动画”选项组中的“长度”值为 90。

(31)单击“播放动画”按钮，查看动画效果。

(32)单击主工具栏上的“渲染”按钮或按快捷键 F10，弹出“渲染设置(默认扫描线渲染器)”对话框。

(33)在“时间输出”选项组中点选“活动时间段”单选按钮。

(34)在“渲染输出”选项组中设置文件保存位置、名称及类型。

(35)单击“文件”按钮,弹出“渲染输出文件”对话框。指定视频保存的位置，在“保存类型”下拉列表中选择文件保存的类型(如 AVI 文件)，单击保存按钮。

(36)单击“渲染”按钮，渲染生成视频文件。

运行结果：



实验分析：

该实验涉及了 3ds Max 的基本文件操作、软件环境布局控制及主工具栏工具的普遍用法，还涉及了基础建模的方法，如内置几何体建模，修改器建模，多边形建模、复合对象建模等。通过本次实验，是我们体会到 3ds Max 建模的一般过程，达到举一反三的目的。

实验成绩：

河南理工大学上机实验报告

2020—2021 学年

第 一 学期

上机时间_____

专业班级 计实验 1801

学号 311809000608

姓名 王荣胜

实验课程名称：虚拟现实技术

实验目的和要求：

实验目的：

1. 掌握在 Unity 3D 中创建项目、场景、Plane 和 Sphere 等。
2. 掌握在 Unity 3D 中材质的应用。

实验要求：

学会在 Unity 3D 中创建项目、场景、Plane 和 Sphere，学会材质的应用。

实验项目名称：

实验过程及代码：

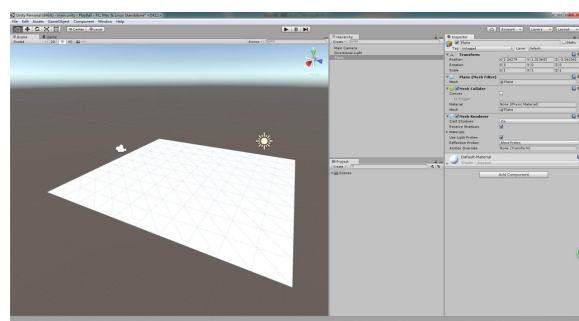
实验一：创建对象

(1)新建项目，依次输入项目名称，选择项目存储路径。

(2)点击“Create project”创建项目进入 Unity 3D 界面，右上角“Layout”可修改布局，Unity 3D 有多种布局，读者可以根据自己的习惯选择，下图为“Layout” | “Tall”布局，本实验截图均为 Tall 布局，如图 2-2 所示。

(3)在“Project”窗口中新建文件夹 Scenes，存储游戏场景，按“Ctrl+S”存储当前游戏场景 main。

(4)在“Hierarchy”窗口中创建游戏地面，右键选择“3D Object” | “Plane”，因为其为游戏地面，将其命名为 Ground。

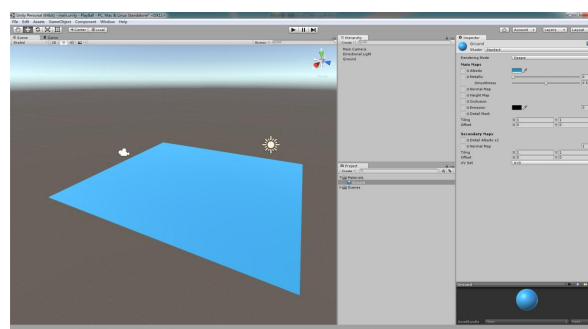


(5)选中 Ground 对象，在“Inspector”窗口的“Transform”右侧的齿轮选择“Reset”，将地面对象放置在坐标系原点

(6)若想修改该地面对象表面颜色，需要为其添加材质。在“Project”窗口中新建文件夹名为 Materials，存储材质，在 Materials 文件夹中新建“Material”材质文件。

(7)将“Material”命名为 Ground，选中 Ground 材质，在“Inspector”窗口中修改“Albedo”颜

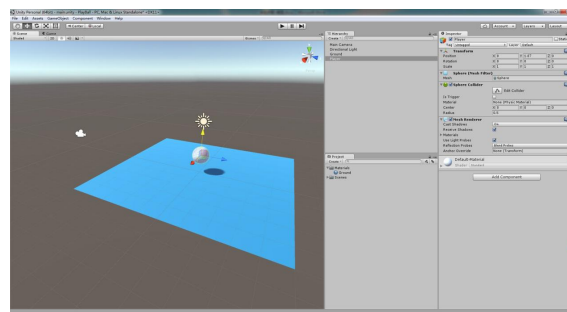
对象上。



(9)在“Hierarchy”窗口创建游戏主角小球对象，右键选择“3D Object” | “Sphere”，并将其命名为 Player。

(10)将小球对象放置在坐标系原点，选中小球 Player 对象，在“Inspector”窗口的“Transform”右侧的齿轮选择“Reset”。

(11)选择移动按钮，，向上拖动“y”轴绿色箭头，将小球拖到地面上方。

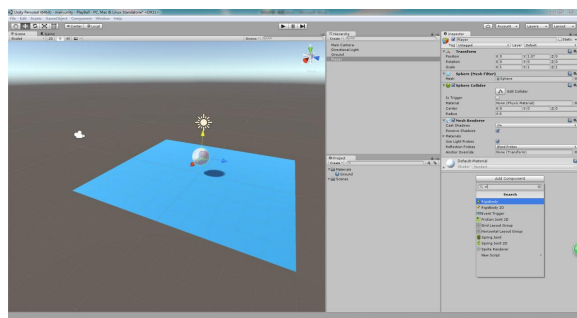


(12)点击播放按钮查看游戏运行效果。

(13)为小球添加重力。在“Hierarchy”窗口选择小球 Player 对象，在“Inspector”窗口点击最小

色为淡蓝色。

(8) 将该材质与 Plane 进行关联, 将 Ground 材质用鼠标左键拖动到 “Hierarchy” 窗口中的 Ground



(14) 点击播放按钮查看与没加刚体组件时的区别。

实验二：添加脚本

(1) 为使小球运动起来, 要小球对象添加脚本文件。在 “Project” 窗口中新建文件夹 Scripts, 存储脚本文件, 在 “Hierarchy” 窗口选择小球 Player 对象, 在 “Inspector” 窗口点击最下面的 “Add component” 添加组件按钮, 输入 Player, 选择 “New Script”, Language 选择默认的 “C#”, 点击 “Create and Add” 创建脚本文件, 如图 2-15 所示。

(2) 将 Player 脚本文件拖入到 Scripts 文件夹中。

(3) 双击 player 脚本文件, 进入编辑器。

(4) 为使小球运动起来, 需要先得到小球刚体, 并对刚体施加一个方向的力, 具体代码如下:

```
using UnityEngine;
using System.Collections;
public class player : MonoBehaviour {
    private Rigidbody rd;    //定义刚体对象
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
        //取得小球的刚体
    }
    // Update is called once per frame
    void Update () {
        rd.AddForce(new Vector3(1, 0, 0));
        //对小球刚体施加一个单位X轴正向的力
    }
}
```

(5) 点击播放, 查看小球运动状态

面的 “Add component” 添加组件按钮, 添加 “Rigidbody” 刚体组件。

(6) 将上例中的 Vector3(1, 0, 0) 修改为 Vector3(-1, 0, 0) 播放查看运动效果, 或者修改为 Vector3(0, 0, 1) 和 Vector3(0, 0, -1) 分别查看效果。

(7) 用键盘控制小球的移动, 需要先取得输入, 再通过输入控制小球的移动, 修改 Update 代码

```
void Update () {
    float h = Input.GetAxis("Horizontal");
    //取得键盘的水平输入
    rd.AddForce(new Vector3(h, 0, 0));
}
```

(8) 点击播放, 试着按键盘的左右方向或者 “A” “D” 键控制小球的水平方向移动, 修改 Update 代码, 实现控制小球四个方向的移动

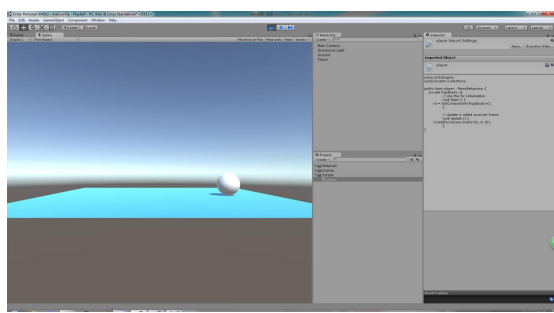
```
void Update () {
    float h = Input.GetAxis("Horizontal");
    //取得键盘的水平输入
    float v = Input.GetAxis("Vertical");
    //取得键盘的垂直输入
    rd.AddForce(new Vector3(h, 0, v));
}
```

(9) 至此可以用键盘实现对小球的方向控制, 若想使小球运动的更快, 可以对施加的力乘以系数, 代码如下:

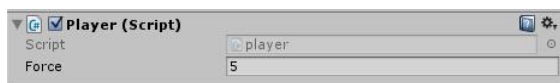
```
rd.AddForce(new Vector3(h, 0, v) * 5);
//对施加的力乘以系数
```

(10) 也可将力的系数设为公共变量, 这样随时可以在 Unity 3D 页面修改系数的值, 代码如下:

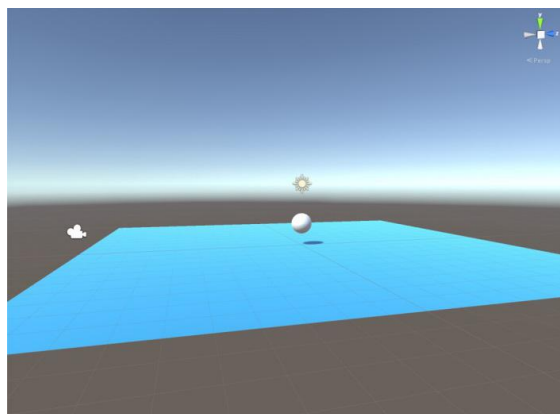
```
using UnityEngine;
using System.Collections;
public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;    //定义公共变量 force, 为施加力的系数
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }
    // Update is called once per frame
    void Update () {
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");
        rd.AddForce(new Vector3(h, 0, v) * force);
        //对施加的力乘以系数
    }
}
```

(11) 设置公共变量之后，在 Player 对象的“Inspector”窗口可以看到，player 脚本中多了 Force 值，这样随时可以在 Unity 3D 页面修改力的系数值。



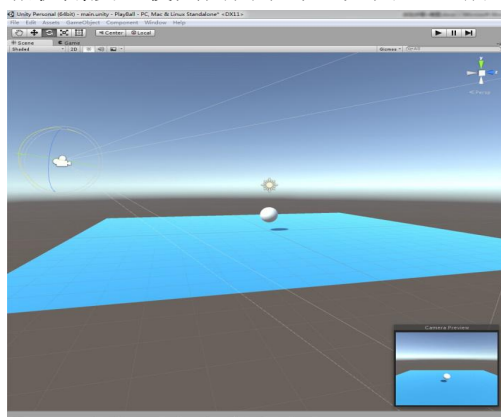
(12) 增大游戏地面面积。在“Hierarchy”窗口选择 Ground 对象，在“Inspector”窗口的“Transform”中修改“Scale”值，将 X、Z 都修改为 2。



实验三：控制相继跟随

(1) 点击移动按钮，选择相机对象，向上拖动“Y”轴，将其拖动到稍高一点的位置，每次调整相机时，在右下角会出现相机预览图，以便观察其视野。

(2) 点击旋转按钮     ，旋转相机角度，使其面向小球，如图 2-22 所示。



```
}}
```

息。代码如下：

```
using UnityEngine;
using System.Collections;
public class follow : MonoBehaviour {
    public Transform playerTransform;
    //定义公共变量记录小球的位置信息
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
    }
}
```

(5) 公共变量定义好后，在“Main Camera”对象的“Inspector”窗口可以看到，follow 脚本中多了 playerTransform，将“Hierarchy”窗口中的小球 Player 对象拖到 playerTransform 上，使 playerTransform 绑定小球的位置信息，如图 2-24 所示。

(6)取得小球的位置信息后，添加代码实现相机跟随小球移动，代码如下：

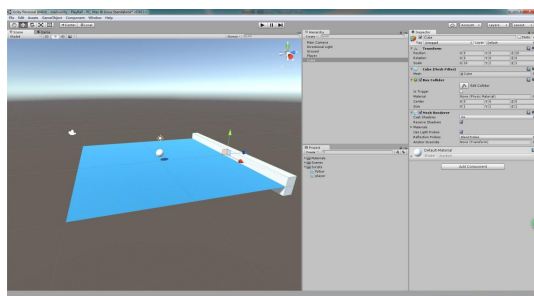
```
using UnityEngine;
using System.Collections;
public class follow : MonoBehaviour {
    public Transform playerTransform;
    //定义公共变量记录小球的位置信息
    private Vector3 offset; //定义相机和小球之前的距离
    // Use this for initialization
    void Start () {
        offset = transform.position -
        playerTransform.position; //计算相机和小球之间距离
    }
    // Update is called once per frame
    void Update () {
        transform.position =
        playerTransform.position + offset; //保持距离
    }
}
```

实验四：旋转对象

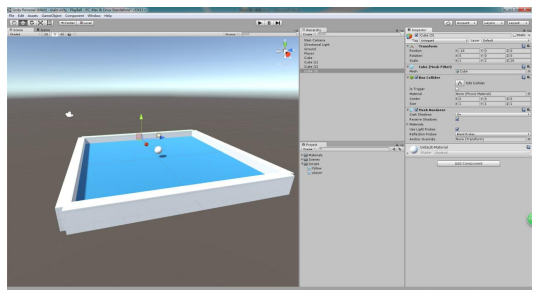
(1)创建地面边界墙，在“Hierarchy”窗口创建 cube 对象，右键选择“3D Object” | “Cube”，选中

(3) 若想相机跟随小球运动，需要为相机添加脚本。在“Hierarchy”窗口选择“Main Camera”对象，在“Inspector”窗口点击最小面的“Add component”添加组件按钮，输入 follow，选择“New Script”，选择默认的“C Shape”语言，点击“Creat and Add”创建脚本文件。

(4)将 follow 脚本拖入到 Scripts 文件夹中。若想相机跟随小球运动，需要相机与小球的位置保持不变。首先，在 follow 脚本中要取得小球的位置信




(2) 在“Hierarchy”窗口选中 cube 对象，按“Ctrl+D”复制 cube 对象，分别创建其他三面墙体，三面墙体的“Transform”参数分别为（Position X、Y、Z 为 0,0,-10，Scale X、Y、Z 为 20,2,1）、（Position X、Y、Z 为 10,0,0，Scale X、Y、Z 为 1,2,20）、（Position X、Y、Z 为-10,0,0，Scale X、Y、Z 为 1,2,20）。



(3) 在“Hierarchy”窗口中，将创建好的四个 cube 墙体对象拖动到 Ground 对象上，形成父子关系。

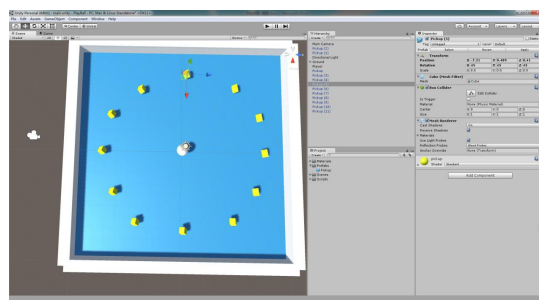
(4) 创建可收集的金币。在“Hierarchy”窗口新建 cube 对象，命名为 Pickup，选中 cube 对象，在“Inspector”窗口的“Transform”中修改“Rotation”的 X、Y、Z 为 45, 45, 45，“Scale”的 X、Y、Z 为 0.5, 0.5, 0.5。

(5) 点击 Local 按钮 ，切换成全局坐标系，调整该 cube 对象的高度，使其立在地面上，如图 2-29 所示。

cube 对象，在“Inspector”窗口的“Transform”中修改“Position”的 X、Y、Z 为 0,0,10，“Scale”的 X、Y、Z 为 20,2,1。

Pickup 对象做成预制体，在“Project”窗口中新建文件夹 Prefabs，存储预制体，将“Hierarchy”窗口中的 Pickup 对象用鼠标左键拖动到该文件夹内。

(8)将 Prefabs 文件夹中的 Pickup 对象拖入到左侧的舞台上，调整高度，使其立于地面上，选中 Pickup 对象，并按“Ctrl+D”复制多个，调整位置。



(9)在“Hierarchy”窗口中，创建空的游戏物体“Create Empty”，并将该游戏物体放到原点，将所有的 Pickup 放到该物体上。

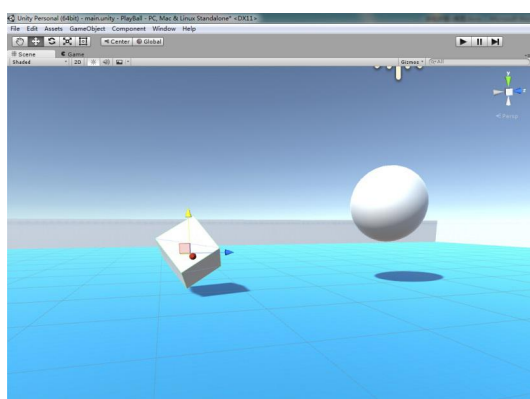
(10)为了使所有的 Pickup 都旋转，选择“Project”窗口下 Prefabs 文件夹中的 Pickup，在“Inspector”窗口点击最小面的“Add component”添加组件按钮，输入 pickup，选择“New Script”，选择默认的“C Shape”语言，点击“Creat and Add”创建脚本文件，并将该脚本放在 Scripts 文件夹下。

(11)双击进入该脚本，添加如下代码，使金币旋转起来。

```
using UnityEngine;
using System.Collections;

public class pickup : MonoBehaviour {
    // Use this for initialization
    void Start () {
    }
    // Update is called once per
    frame
    void Update () {
        transform.Rotate(new
        Vector3(0, 1, 0)); //使对象绕y轴旋转
    }
}
```

(12)点击播放，游戏运行如图



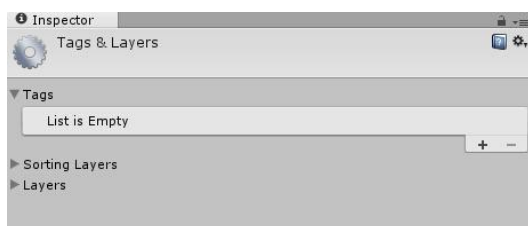
(6)在 Materials 文件夹中新建 Material,将 Material 命名为 Pickup,选中 Pickup 材质,在“Inspector”窗口中修改“Albedo”颜色为金色,并将该材质与 Pickup 对象关联,如图 2-30 所示。

(7)因为还要创建很多个 Pickup 对象,所以把

实验五：碰撞检测

(1) 为了检测小球碰撞到金币还是墙体,为金币 Pickup 对象设置标签,选择“Project”窗口下 Prefabs 文件夹中的 Pickup,在“Inspector”窗口“tag”下拉菜单中选择“Add Tag”,如图 2-36 所示。

(2) 进入如图界面



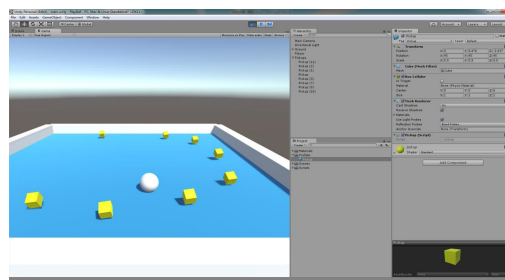
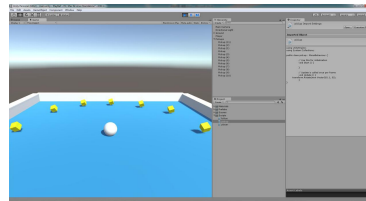
(3) 点击“+”,添加标签 Pickup。

(4) 再次选择“Project”窗口下 Prefabs 文件夹中的 Pickup,在“Inspector”窗口“tag”下拉菜单中选择 Pickup。

(5) 为了实现小球碰撞金币使金币消失,打开 Player 脚本,修改代码如下

```
using UnityEngine;
using System.Collections;

public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }
    // Update is called once per frame
    void Update () {
        float h = Input.GetAxis("Horizontal");
```



(7)游戏运行后发现,在碰撞金币使金币消失时,会有撞到物体的反弹效果,如果不想要这种效果,首先选择“Project”窗口下 Prefabs 文件夹中的 Pickup,在“Inspector”窗口“Box Collider”中,勾选“Is Trigger”。

(8)修改 Player 脚本如下:

```
using UnityEngine;
using System.Collections;

public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }
    // Update is called once per frame
    void Update () {
        float h =
        Input.GetAxis("Horizontal");
        float v =
        Input.GetAxis("Vertical");
        rd.AddForce(new Vector3(h, 0, v)
        * force);
    }
    void OnTriggerEnter(Collider
    collider) //检测接触
    {
        if(collider.tag == "Pickup")
        {
            Destroy(collider.gameObject);
        }
    }
}
```

```

float v = Input.GetAxis("Vertical");
rd.AddForce(new Vector3(h, 0, v) *
force);
}
void OnCollisionEnter(Collision
collision)    //碰撞检测函数
{
    if(collision.collider.tag == "Pickup")
//判断小球是否碰到金币
    {

Destroy(collision.collider.gameObject)
;    //使金币消失
    }
}
}

```

(6) 运行游戏效果如图所示

实验六：显示文本

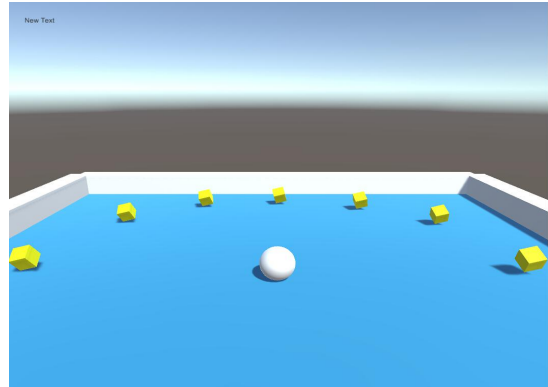
(1) 如果想在小球吃掉金币时加分，并实时显示分数，修改 Player 代码如下

```

using UnityEngine;
using System.Collections;
public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;
    private int score = 0;    //定义分数变量
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }

    // Update is called once per
frame
    void Update () {
        float h =
Input.GetAxis("Horizontal");
        float v =
Input.GetAxis("Vertical");
        rd.AddForce(new Vector3(h, 0, v)
* force);
    }
    void OnTriggerEnter(Collider
collider)
    {
        if(collider.tag == "Pickup")
        {
            score++;    //每次接触金币
对分数加1

```



(5) 为了在 Text 对象中显示分数，需要将 Text 对象与 Player 脚本关联，修改 Player 脚本代码如下

```

using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;
    public Text text;    //定义Text公
共变量，准备关联Text对象
    private int score = 0;
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }

    // Update is called once per
frame
    void Update () {
        float h =
Input.GetAxis("Horizontal");
        float v =
Input.GetAxis("Vertical");
        rd.AddForce(new Vector3(h, 0, v)
* force);
    }
    void OnTriggerEnter(Collider
collider)
    {
        if(collider.tag == "Pickup")
        {
            score++;

```

```

        Destroy(collider.gameObject);
    }
}

```

(2) 若想在游戏时显示分数，需要加入 UI，在“Hierarchy”窗口中，右键“UI” | “Text”，切换到 2D 显示。

(3) 选择 Text 对象，在“Inspector”窗口“Rect Transform”中，点击左侧方框，按住“Alt”键，选择左上角的方框。

(4) 调整 Text 位置，将其向右下稍微调整，游戏运行，效果如图。

(6) 在 Player 对象的脚本中会出现 Text，将 Text 对象拖入到 Text 后面的框中。

(7) 在 Player 对象的脚本中会出现 Text，将 Text 对象拖入到 Text 后面的框中。

(8) 通过拖动已经在 Player 中取得了 Text 对象，然后将分数传递给 Text，修改 Player 代码如下

```

using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;
    public Text text;
    private int score = 0;
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }

    // Update is called once per
frame
    void Update () {
        float h =
Input.GetAxis("Horizontal");
        float v =
Input.GetAxis("Vertical");
        rd.AddForce(new Vector3(h, 0, v)
* force);
    }
    void OnTriggerEnter(Collider
collider)
    {
        if(collider.tag == "Pickup")
        {

```

```

        Destroy(collider.gameObject);
    }
}

```

(10) 当把所有金币都吃掉后，显示游戏胜利。在“Hierarchy”窗口中，在“Canvas”上右键“UI” | “Text”，再创建一个 Text 用来显示胜利信息。

(11) 选择调整按钮  调整 Text (1) 对象的大小，在 Text (Script) 中修改 Text (1) 的一些属性。

(12) 在“Inspector”窗口中取消 Text (1) 前面的对号，使其在游戏刚开始运行时不显示。

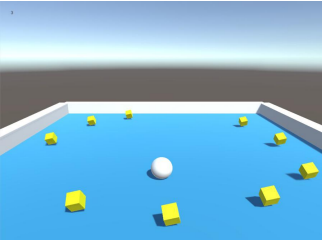
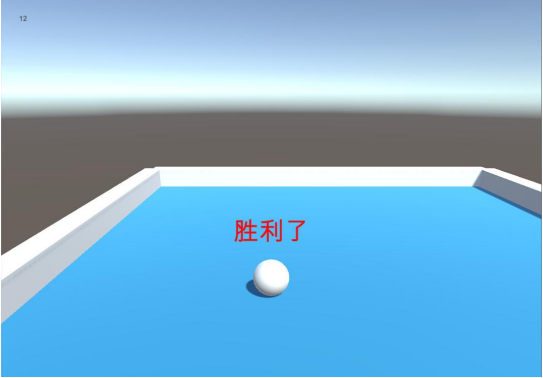
(13) 用同样的方法关联 Player 和 Text (1) 对象，并修改 Player 脚本的代码如下

```

using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class player : MonoBehaviour {
    private Rigidbody rd;
    public int force = 5;
    public Text text;
    private int score = 0;
    public GameObject winText; //定义
存放胜利消息的Text
    // Use this for initialization
    void Start () {
        rd = GetComponent<Rigidbody>();
    }

    // Update is called once per
frame
    void Update () {
        float h =
Input.GetAxis("Horizontal");
        float v =
Input.GetAxis("Vertical");
        rd.AddForce(new Vector3(h, 0, v)
* force);
    }
    void OnTriggerEnter(Collider
collider)
    {
        if(collider.tag == "Pickup")
        {
            score++;

```


<pre>score++; text.text = score.ToString(); //将分数赋值给Text Destroy(collider.gameObject); }}}</pre> <p>(9)运行游戏，效果如图所示。</p>  <pre>winText.SetActive(true); //如果全部金 币（本例为12个）全部吃完，显示胜利消息 }</pre> <pre>Destroy(collider.gameObject); }}}</pre> <p>(14)在 Player 对象的脚本窗口中会出现 winText，将 Text（1） 对象拖入到 winText 后面的框中。</p> <p>(15)运行游戏，当吃光所有金币后效果如图 2-51 所示</p>	<pre>text.text = score.ToString(); if(score == 12) {</pre> 
<p>运行结果：</p> <p>小球吃金币游戏成功运行</p>	
<p>实验分析：</p> <p>小球吃金币实验利用 unity3d 实现了撞击，消失，显示文字，键盘控制方向等功能，unity3d 建模操作采用窗口按键控制，而基本的动画如撞击，消失等需要采用代码实现，这使得 unity3d 建模过程更加方便。同时通过这次实验让我们了解了基础的 unity3d 的操作和一般过程，达到举一反三的目的。</p>	
<p>实验成绩：</p>	