项目二 2D 世界-八分音符

利用 Unity 设计完成一款通过音量控制角色的 2D 小游戏。当音量较小时角色向前移动，当音量较大时角色跳起，角色跳起的高度由音量大小控制。

课堂学习目标

- 创建工程及脚本
- 创建主要 UI
- 通过代码控制 Bird 移动
- 设置游戏失败机制
- 设置游戏重新开始机制
- 实现相机跟随功能
- 制作障碍物

任务 2.1　创建工程及脚本

任务要求

小贝已经完成软件的安装和配置，开始学习开发第一个 2D 项目，需要完成如下工作内容：

（1）创建 2D 工程

（2）创建脚本文件

任务实现

步骤 2.1.1 新建 2D 工程项目

（1） 打开 Unity，点击【新建】按钮创建新的项目。如图 2-1-1 所示：

图 2-1-1

（2）　在弹出窗口中点击【2D】模板、输入项目名称“bafenyinfu”（unity 文件名称不能出现中文）以及选择项目文件存储路径，点击【创建】按钮完成操作。如图 2-1-2 所示：

图 2-1-2

步骤 2.1.2 创建脚本

1. 创建脚本

（1）　首先新建一个脚本文件夹并命名为 script，在 Assets 面板中点击【鼠标右键】，弹出弹框后点击【Creat】,接着再点击【Folder】创建新文件夹。如图 2-1-2 所示：

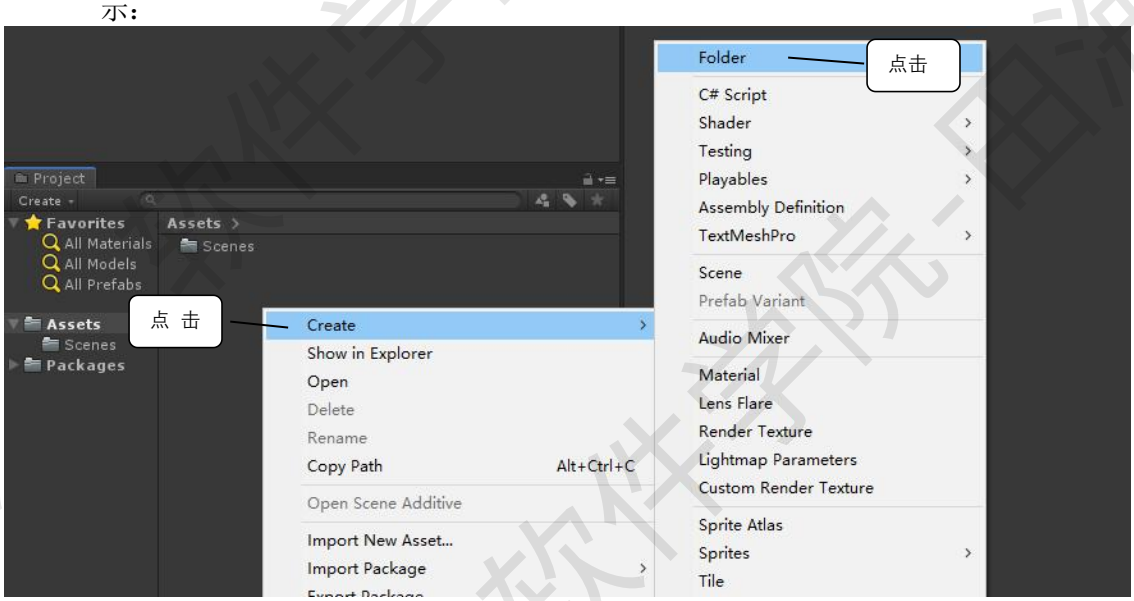


图 2-1-4

（2）　将它命名为“script”。如图 2-1-5 所示：

图 2-1-5

（3） 创建一个 C#脚本文件并命名为“MicInput.cs”，通过此脚本来获取音量大小。在 script 文件夹里再次点击【鼠标右键】，弹出弹框后点击【Creat】,接着再点击【C# Script】。如图 2-1-6 所示：

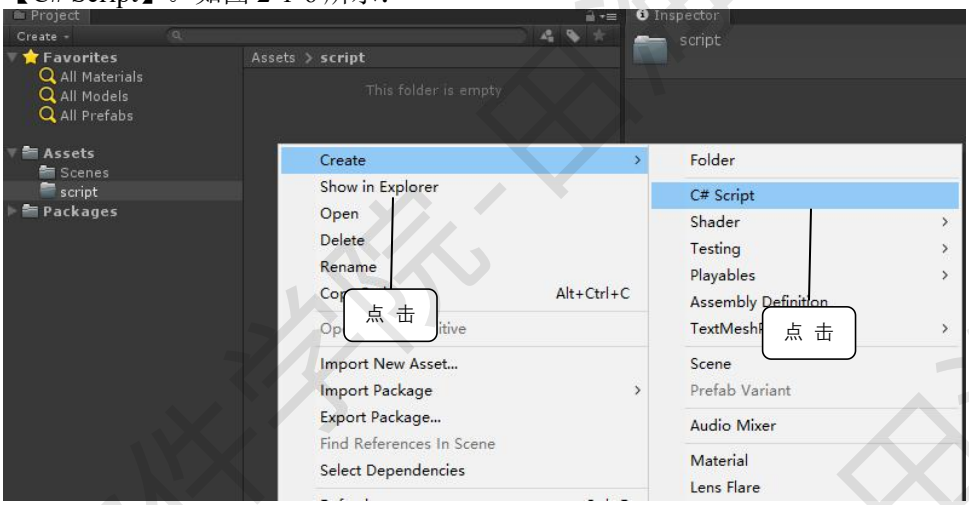


图 2-1-6

（4）命名为“MicInput”，如图 2-1-7 所示：

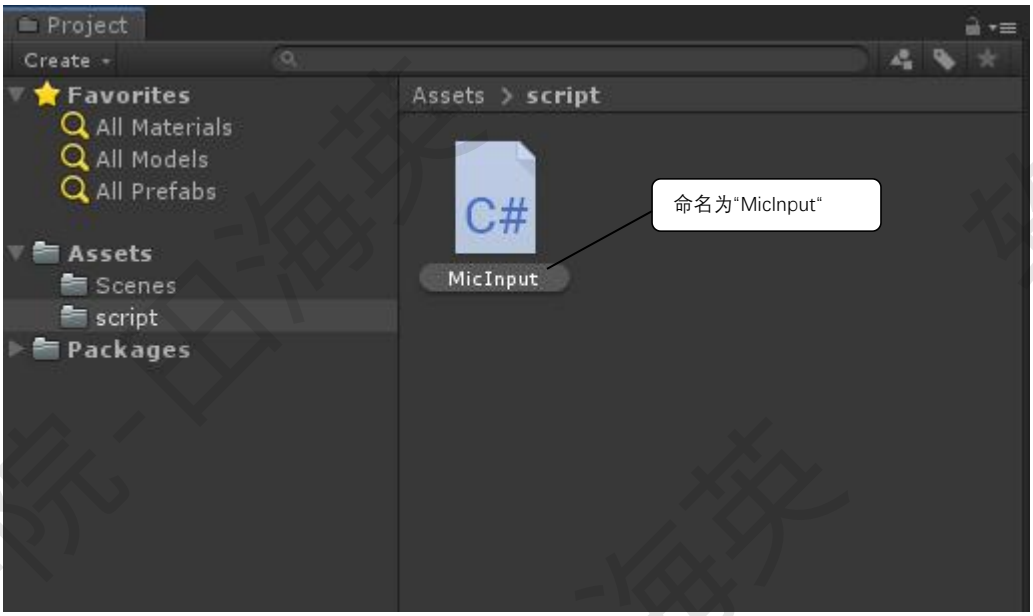


图 2-1-7

2. 定义变量

（1）在 script 页面上双击打开【MicInput】。如图 2-1-8 所示：

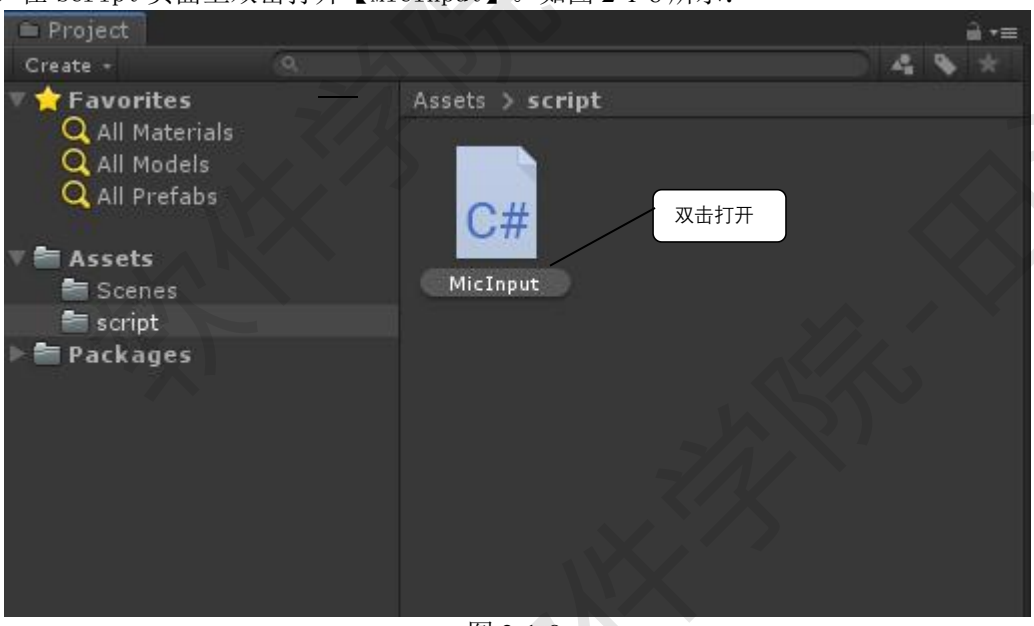


图 2-1-8

（2）定义音量大小、声音信息以及设备的名字，脚本界面如下图 2-1-9 所示：

```
MicInput.cs*
NewBehaviourScript
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour
{
    public float volume;    //储存音里大小
    AudioClip micRecord;    //储存声音信息
    string device;          //设备的名字
```

图 2-1-9

2. 获取录制设备的名称

（1）获取默认设备、声音，脚本界面如下图 2-1-10 所示：

```
void Start()
{
    device = Microphone.devices[0];     //获取默认设备
    micRecord = Microphone.Start(device, true, 999, 44100);    //获取声音
}
```

图 2-1-10

4. 截取声音音量

（1）截取录取声音中的一小段获取其音量最大值，如图 2-1-11 所示：

```
void Update()
{
    volume = GetMaxVolume();
}
float GetMaxVolume()
{
    float maxVolume = 0f;
    float[] volumeData = new float[128];
    int offset = Microphone.GetPosition(device) - 128 + 1;
    if (offset < 0)
    {
        return 0;
    }
    micRecord.GetData(volumeData, offset);

    for (int i = 0; i < 128; i++)
    {
        float tempMax = volumeData[i];
        if (maxVolume < tempMax)
        {
            maxVolume = tempMax;
        }
    }
    return maxVolume;
}
```

图 2-1-11

相关知识

Unity 引擎开发项目使用 C#语言，C#的编程开发工具-Visual Studio 或者 Visual Studio code，Unity Hub 在安装过程中会默认安装 Visual Studio 。

2.1 Visual Studio 开发工具介绍

Visual Studio 是微软公司开发的一款集中平台应用开发工具，通过 C#控制台项目中编写功能程序代码，可以在 windows 平台下完成对项目功能的实现。Visual Studio 的下载和安装可以参考官方网站 https://visualstudio.microsoft.com/zh-hans/downloads/，网站有不同的版本以及不同开发系统的选择供用户选择。

2.2 Unity C# 编程基础

C# 作为 Unity 引擎的编程语言，跟其他语言类似，开发者需要掌握数据类型及类型转换，熟练使用运算符，掌握面向对象的程序设计理念等等，具体语法可以参考 C#语言的教程，因篇幅有限不再重复赘述。

任务 2.2　创建主要 UI 并控制 Bird 移动

任务要求

小贝已经在完成 2D 工程的创建和变量的定义，需要搭建 2D 场景和代码约束，需要完成如下工作内容：

（1）UI 搭建
（2）Bird 移动

任务实现

步骤 2.2.1　UI 搭建

1. 新建空物体

（1）新建一个空物体并命名为 GM。在 Hierarchy 面板上点击【鼠标右键】，选择点击【Creat Empty】并命名为“GM”，如图 2-2-1 所示，将 MicInput.cs 脚本绑定在 GM 上，如图 2-2-2 所示。

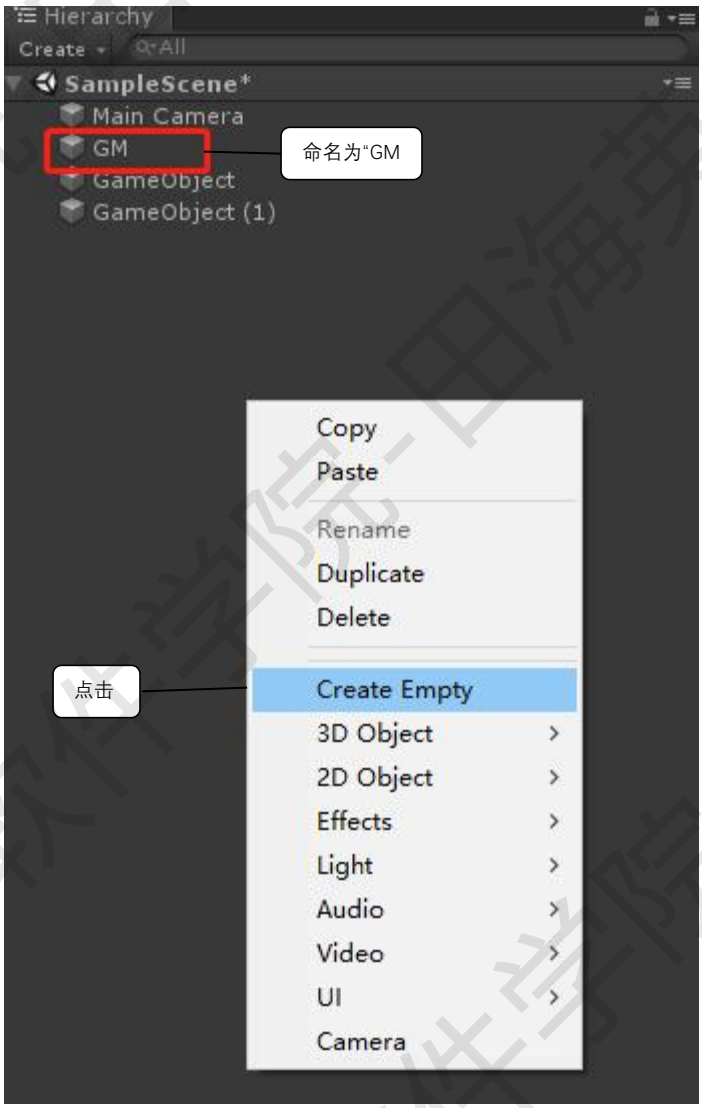


图 2-2-1

图 2-2-2

（2）测试 Volume 值，如图 2-2-2（1）所示。并将 Scene 窗口改成 2D 模式，点击【2D】，如图 2-2-2（2）所示

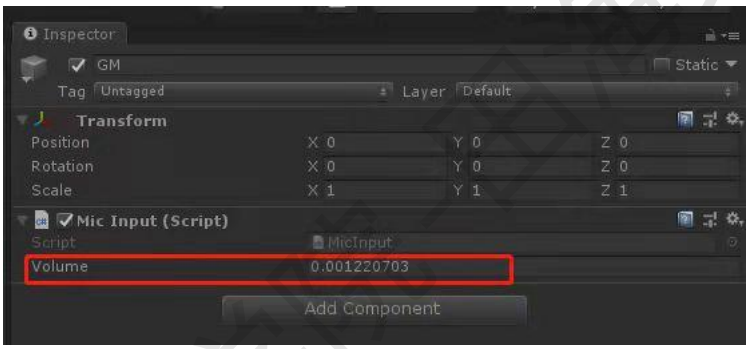


图 2-2-2（1）

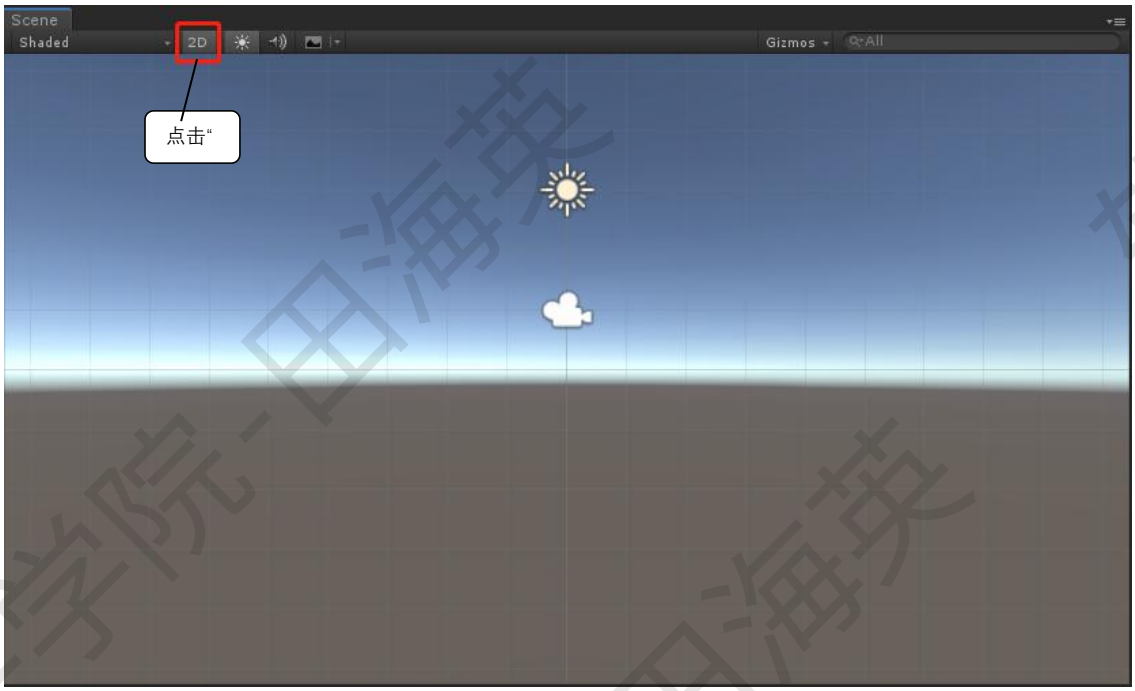


图 2-2-2（2）

2. 创建 Bird

（1）在 Hierarchy 面板中点击【鼠标右键】，顺序选择点击【2D Object --> Sprite】，命名为“Bird”如图 2-2-4，图 2-2-5 所示：

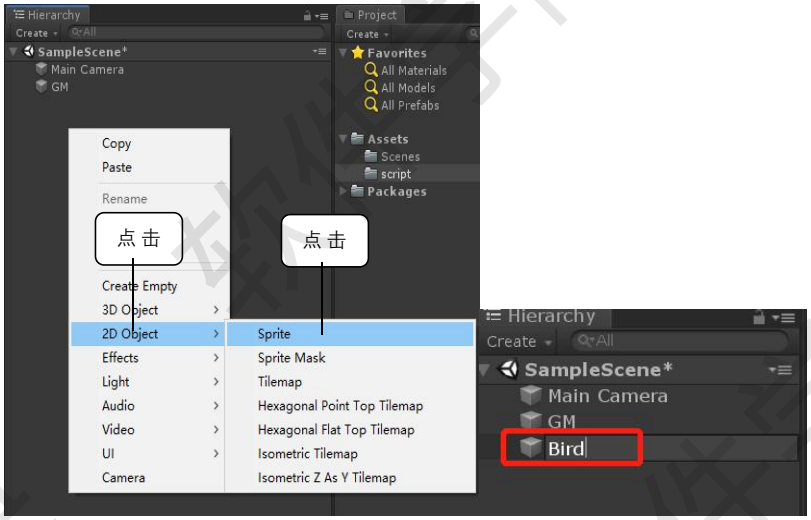


图 2-2-4　　图 2-2-5

（2）添加资源。将“Assets --> picture --> angry bird”目录下的图片 1 拖拽至 Bird 的 Inspector 面板上，如图 2-2-6 所示完成图片拖拽。Scene 界面与 Game 界面的 Bird 效果图，如图 12-2-4 所示。

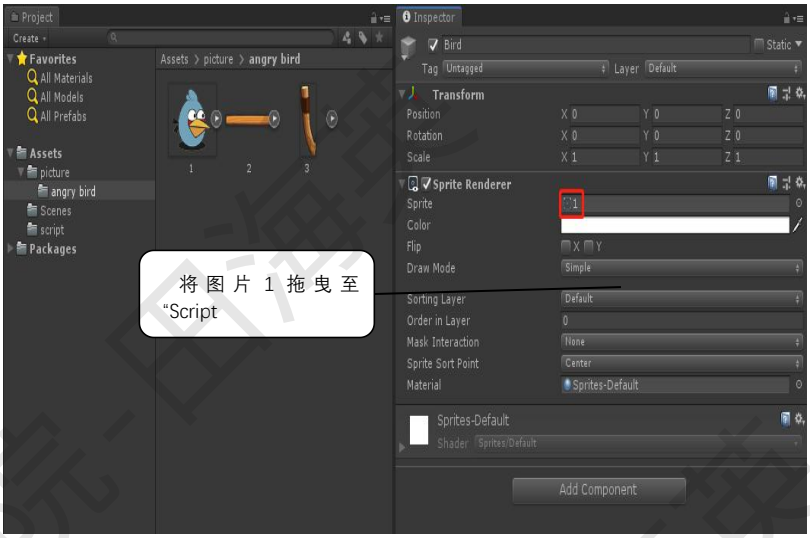


图 2-2-6

：

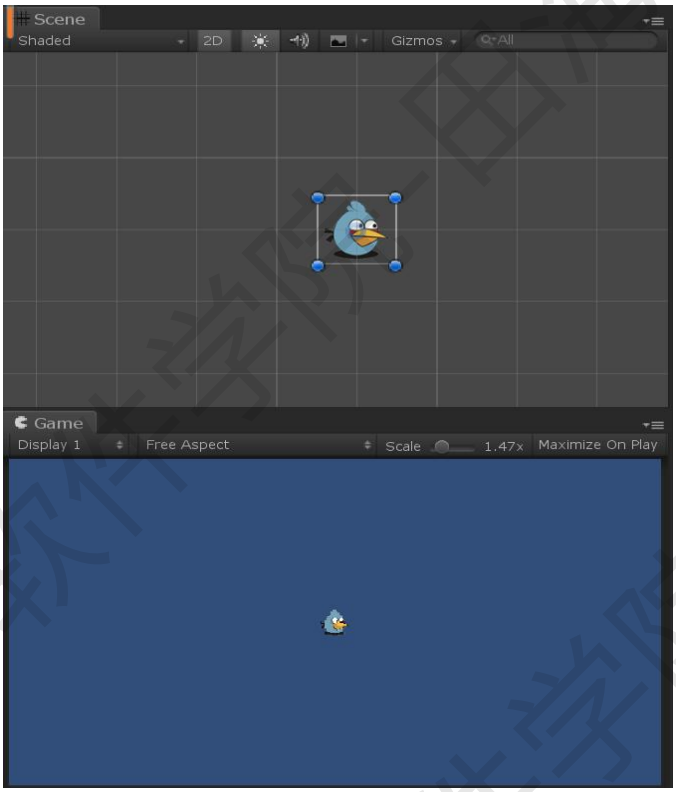


图 2-2-4

2. 添加地形 UI

（1）创建 2D Object 下的 Script 并命名为“zhangaiwu（障碍物）”。在 Hierarchy 面板中点击【鼠标右键】，顺序选择点击【2D Object --> Sprite】，命名为“zhangaiwu”如图 2-2-4 所示：

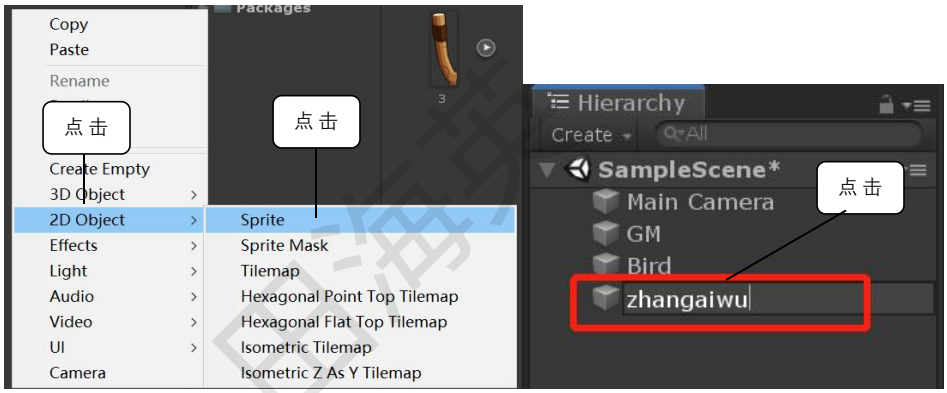


图 2-2-4

（2）将图片 2 拖拽至障碍物的 Inspector 面板上，并调整障碍物的大小，如图 2-2-9 所示：

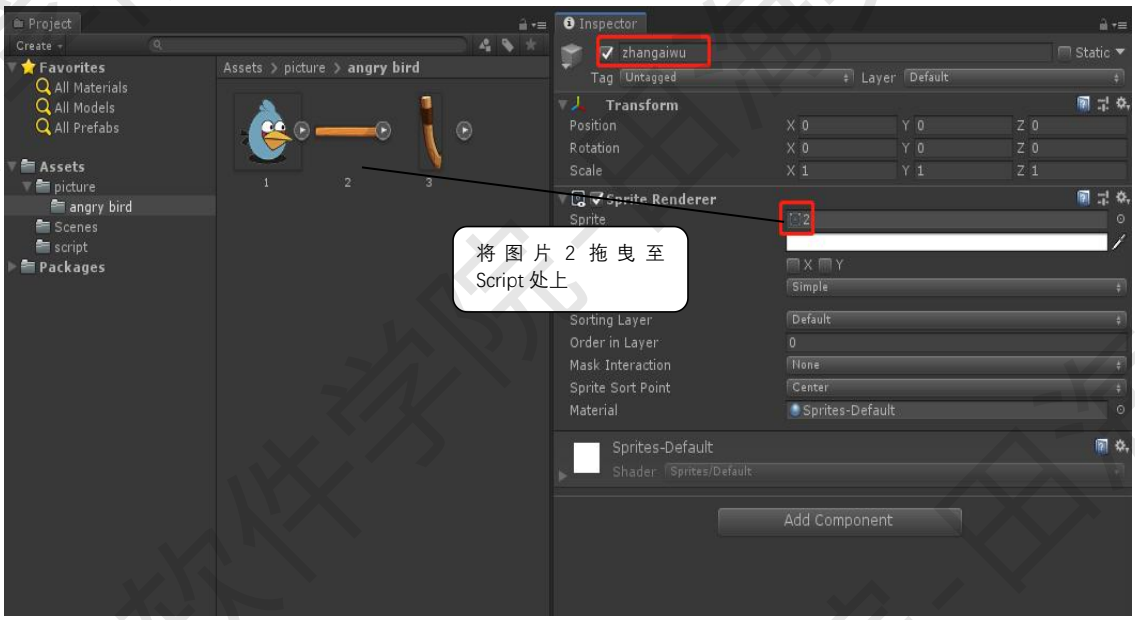


图 2-2-9

（2）Scene 界面与 Game 界面的障碍物效果图如图 2-2-10 所示：

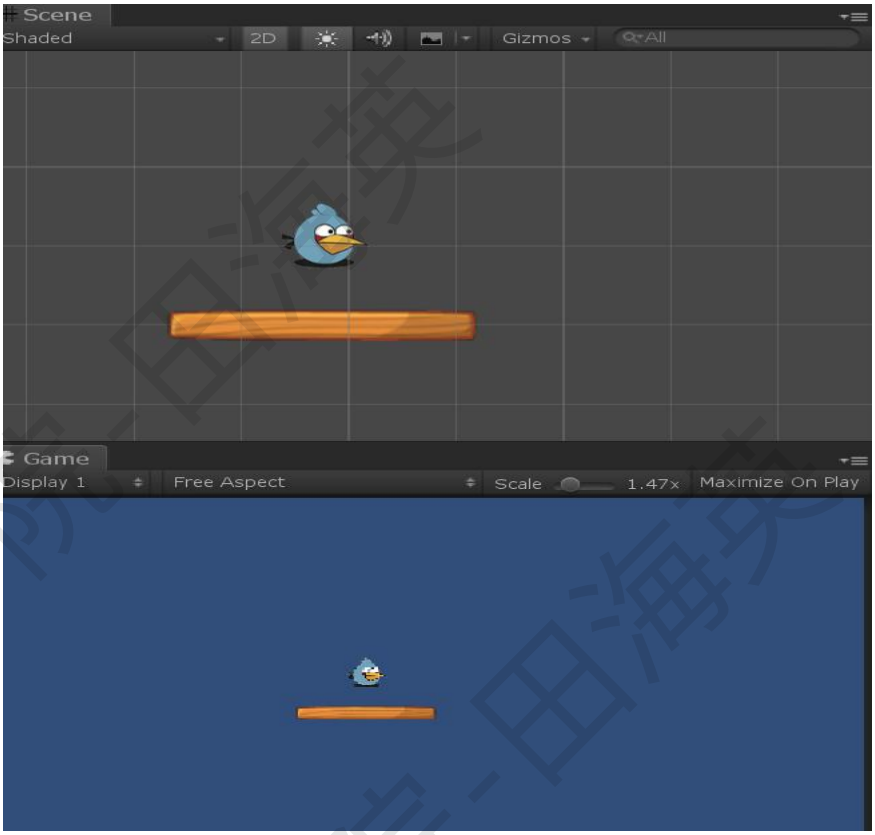


图 2-2-10

4. 添加 Rigidbody 2D 属性

（1）给 Bird 添加 Rigidbody 2D 属性，先点击【Add Component】出现弹窗，再点击【Rigidbody 2D】即属性添加成功，如若没有找到 Rigidbody 2D，则点击【搜索框】进行查找，如图 2-2-11 所示。Rigidbody 2D 添加成功如图 2-2-12 所示。

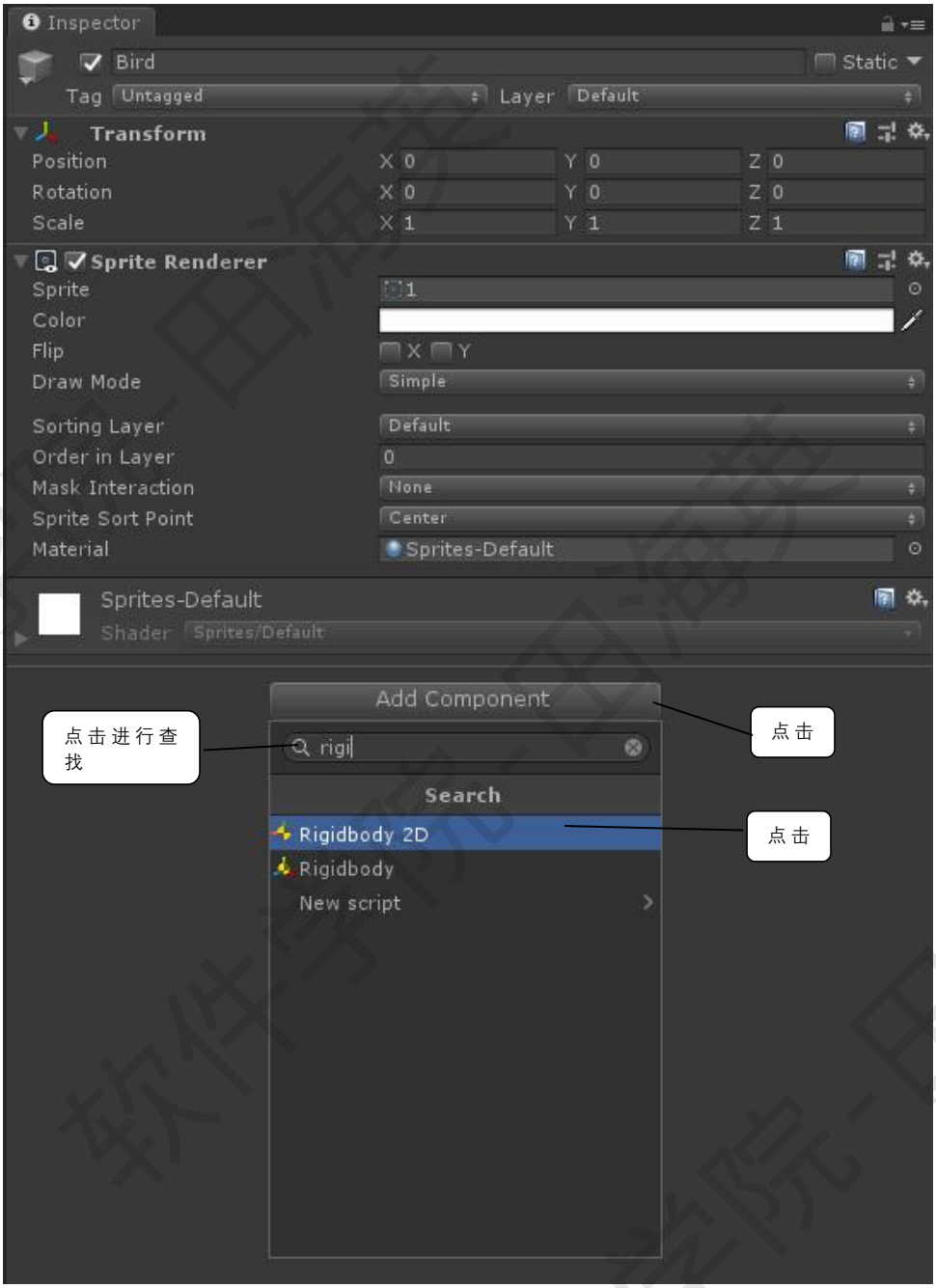

Inspector
Bird
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer
Sprite
Color
Flip X Y
Draw Mode Simple
Sorting Layer Default
Order in Layer 0
Mask Interaction None
Sprite Sort Point Center
Material Sprites-Default
Sprites-Default
Shader Sprites/Default
Add Component
点击进行查找
rigi
Search
Rigidbody 2D
Rigidbody
New script
点击
点击

图 2-2-11

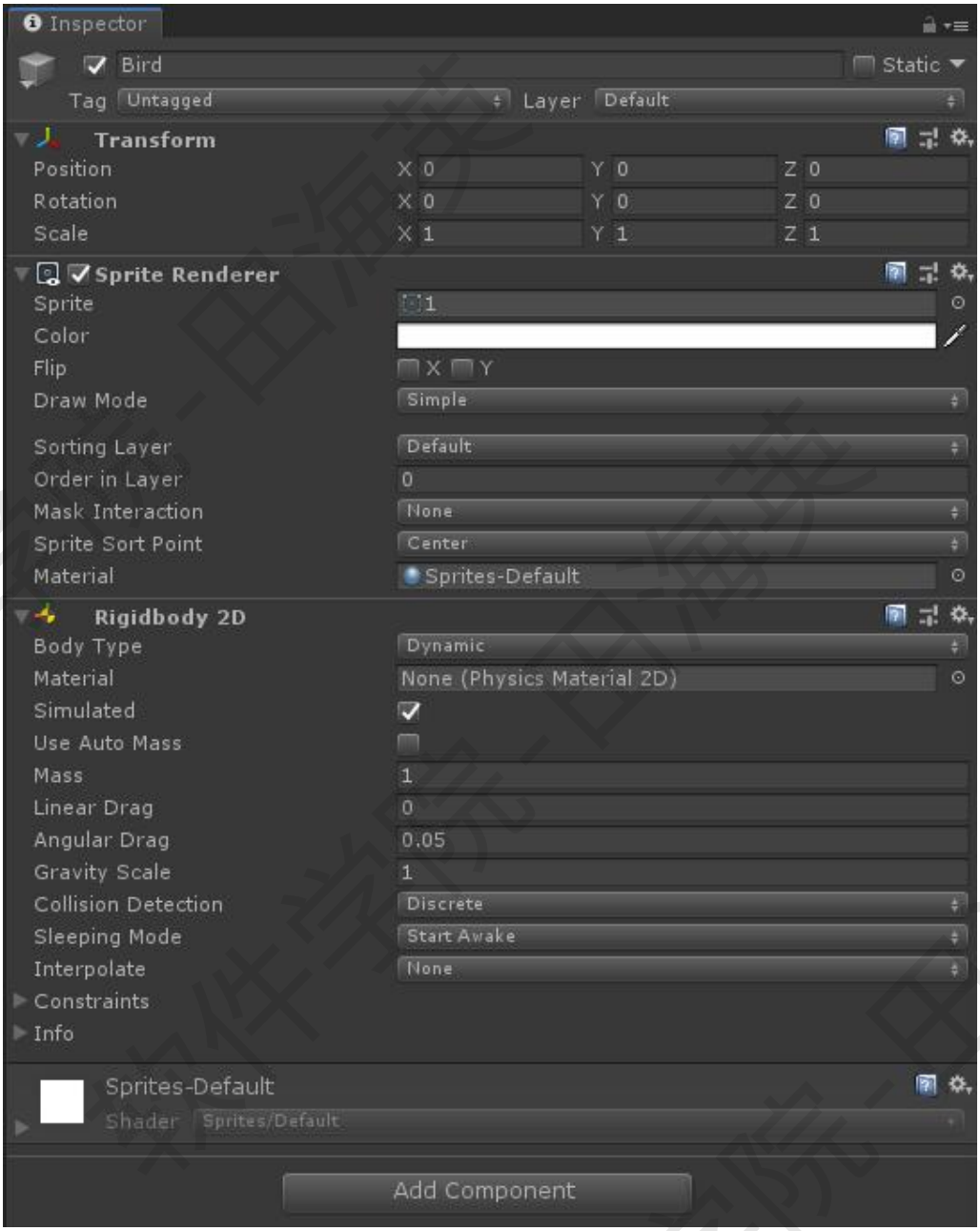


图 2-2-12

（2）为了避免在碰撞到障碍物时，Bird 翻滚行走，需勾选【Freeze Rotation】选项，如图 2-2-12 所示：

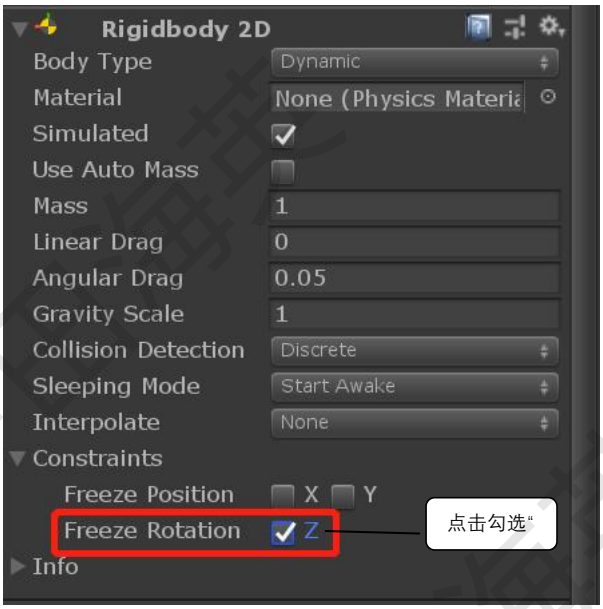


图 2-2-12

5. 添加 Box Collider 2D 属性

（1）分别给 Bird 和障碍物添加 Box Collider 2D 属性。先点击【Add Component】出现弹窗，再点击【Box Collider 2D】即属性添加成功，如若没有找到 Box Collider 2D，则点击【搜索框】进行查找，如图 2-2-14（1）、2-2-14（2）所示：

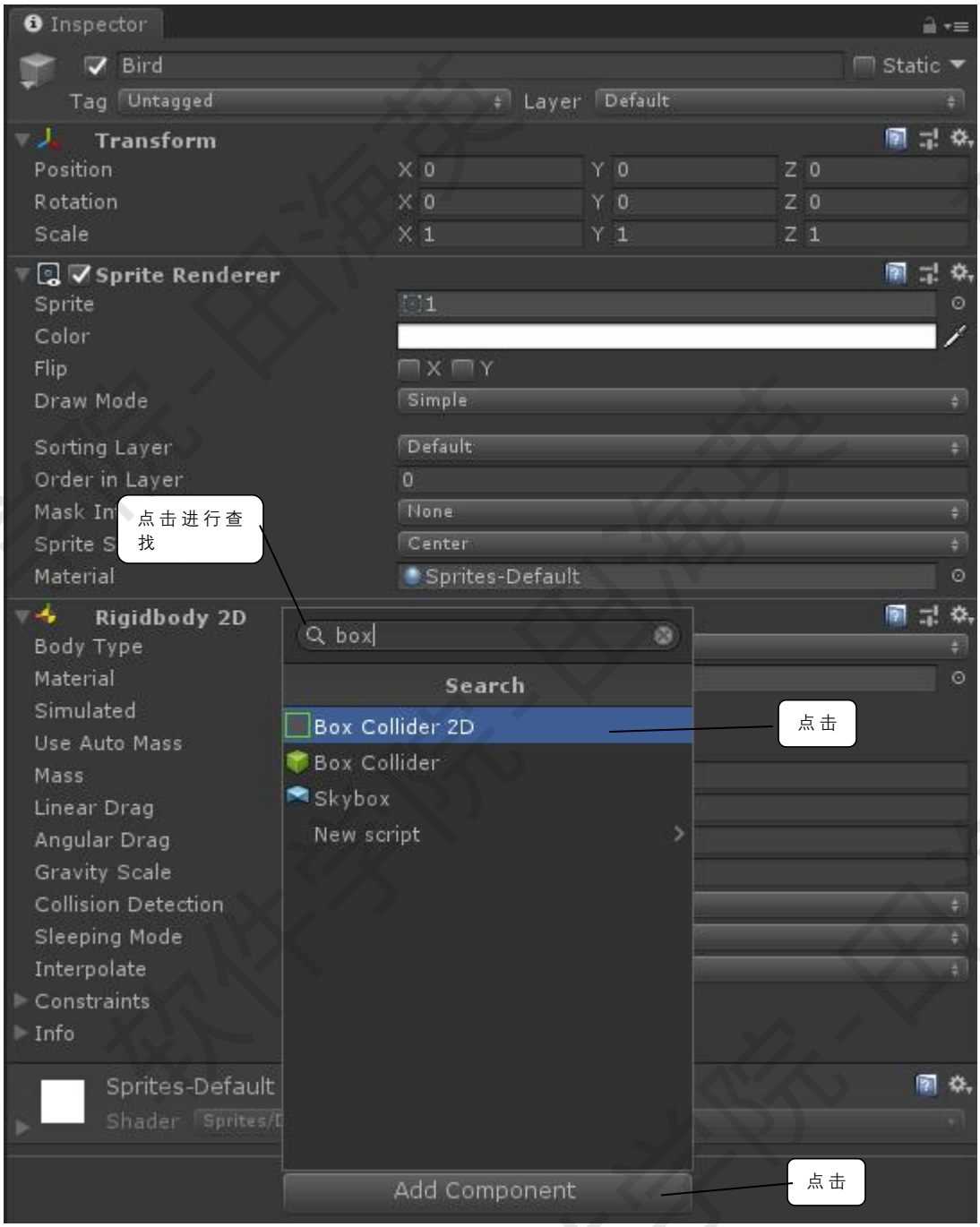


图 2-2-14（1）Bird 添加属性

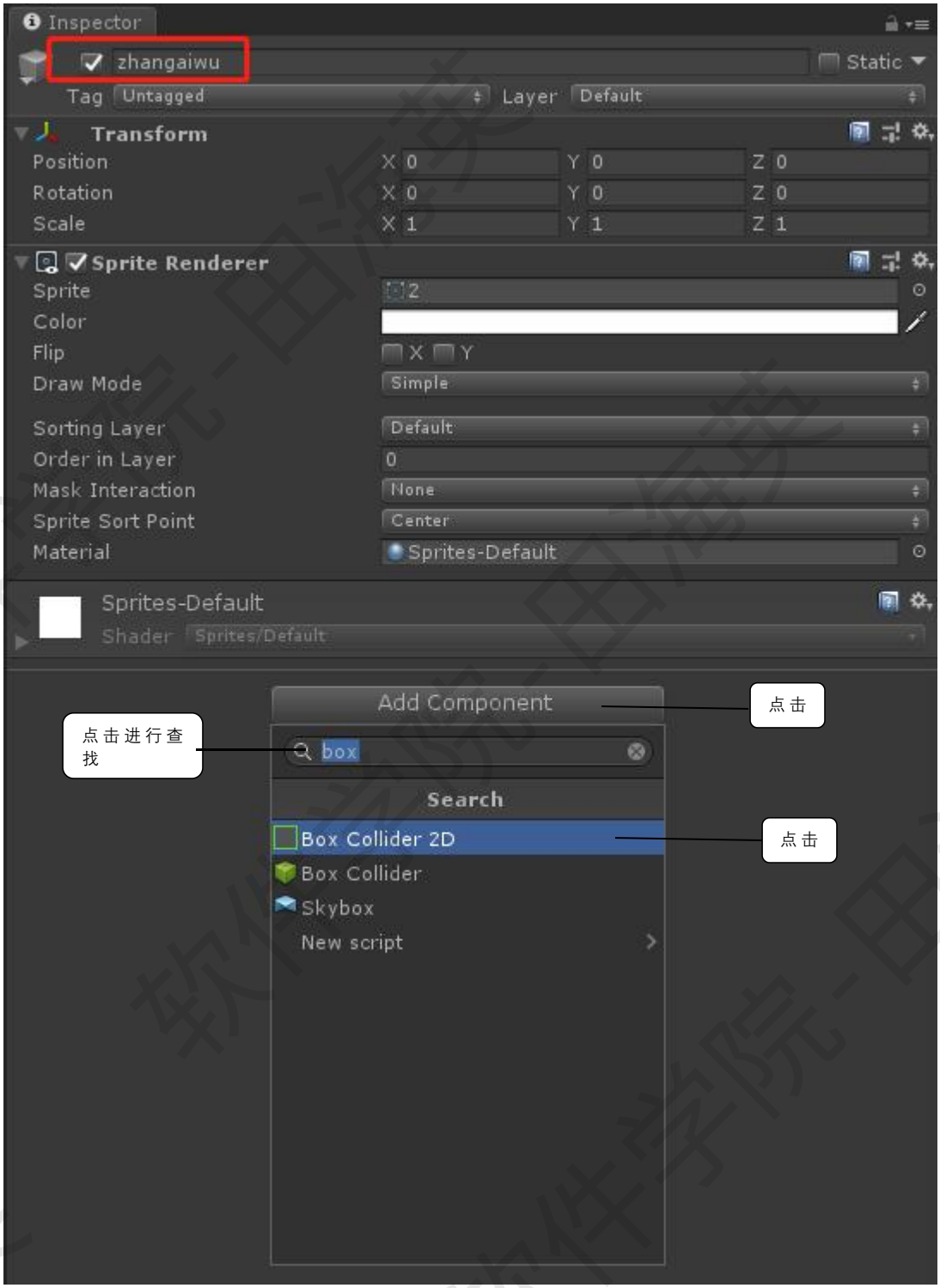


图 2-2-14（2）障碍物添加属性

6. 复制障碍物

（1）通过复制制作多个障碍物，在 Hierarchy 面板中选中点击【zhangaiwu】，键盘按住【Ctrl+C】复制，【Ctrl+V】粘贴，如图 2-2-15 所示：

图 2-2-15

（2）调整障碍物的位置及大小，其效果如图 2-2-16 所示：

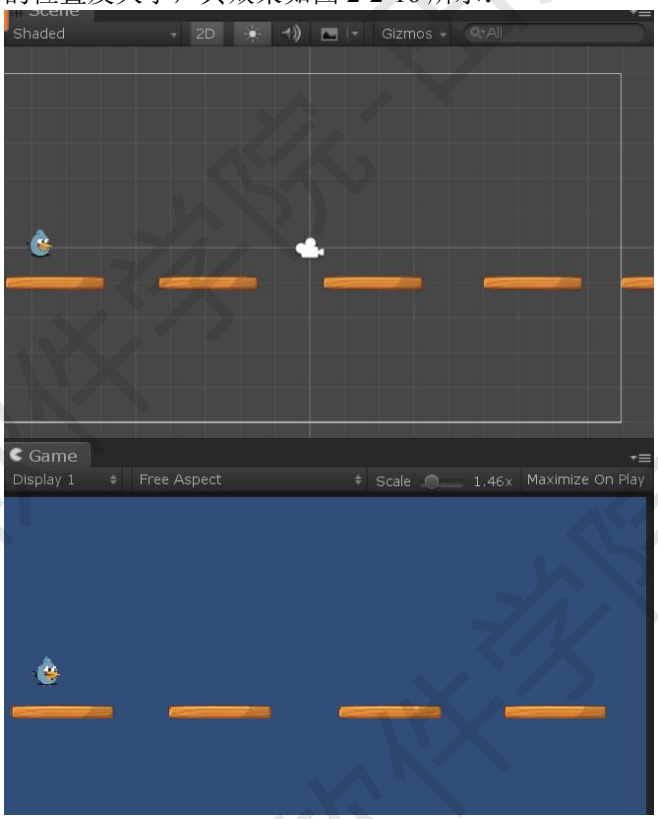


图 2-2-16

（2）改变 Game 窗口中的背景颜色，在 Inspector 面板中 Camera 板块中，将 Clear Flags 参数修改为 Solid Color，双击【Background】调整颜色，如图 2-2-17 所示：

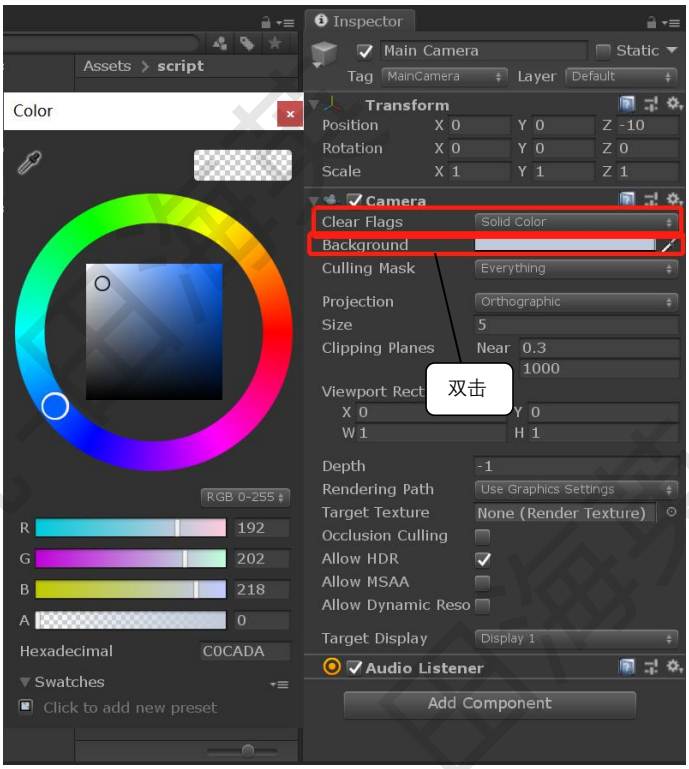


图 2-2-17

步骤 2.2.2　通过代码控制 Bird 移动

1. 修改 MicInput.cs 脚本

（1）为了能够获取 volume 的值，将“public float volume”语句改为“public static float volume”语句，代码修改如图 2-2-18 所示：

```
public class MicInput : MonoBehaviour
{
    1 reference
    public static float volume;    //储存音量大小
    2 references
    AudioClip micRecord;    //储存声音信息
    3 references
    string device;          //设备的名字
```

图 2-2-18

2. 创建脚本 QuaverCtrl.cs

（1）创建一个 C#脚本文件并命名为 QuaverCtrl（创建过程见上面任务 2.1），如图 2-2-19 所示：

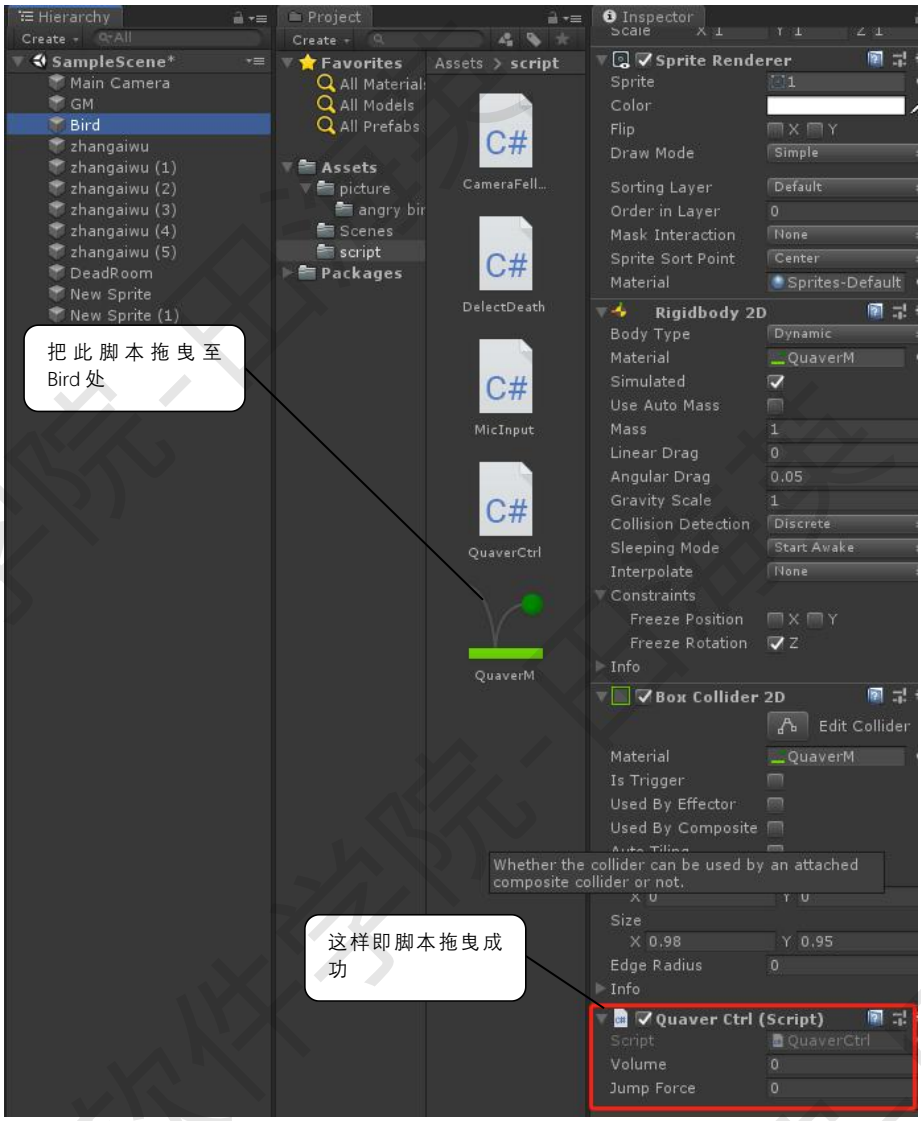
图 2-2-19

（2）编写代码实现通过获取 volume 的值来控制 Bird 的移动的功能，脚本代码如图 2-2-20 所示：

```
public float volume; //记录音量大小
6 references
Rigidbody2D rg;
1 reference
public float jumpForce; //将跳起的力度设置成500（根据每个人计算机声卡的不同，设置力度不同）
2 references
float tempTime=0;
2 references
float maxSpeed = 5f; //限制X轴最大速度为5
// Start is called before the first frame update
0 references
void Start()
{
    rg = GetComponent<Rigidbody2D>(); //获取数据
}

// Update is called once per frame
0 references
void Update () {
    volume = MicInput.volume; //获取MicInput脚本中volume值

    if (volume > 0)
    {
        MoveForward ();
        if (rg.velocity.x > maxSpeed)
        {
            rg.velocity = new Vector2 (maxSpeed, rg.velocity.y);
        } //限制X 轴最大速度
    }
     if (volume > 0.4) {
         if (Time.time-tempTime>2){


         Jump();
             tempTime = Time.time;
         }

     } // 将MicInput中的音量导入该脚本中，并且根据音量判断Bird是跳起还是前进，且为了避免Bird跳得过高还添加了t

     }

 1 reference
 void  Jump(){
     rg.AddForce (Vector2.up * jumpForce * volume);
  } //设置跳起的高度
  1 reference
  void MoveForward(){
     rg.AddForce (Vector2.right *50 * volume);
  } //设置前进的长度，其中“50”为根据自己的计算机设置不同的力度
```

图 2-2-20

（2）把 QuaverCtrl.cs 脚本拖曳至 Bird 处，如图 2-2-21 所示：



2. 修改 Bird 移动参数

（1）因 Bird 向前滑行时需修改其摩擦系数，所以在 Project 面板下创建 Physics Material 2D，命名为。点击【鼠标右键】，再点击【Creat -->Physics Material】并命名为“QuaverM”。如图 2-2-22 所示：

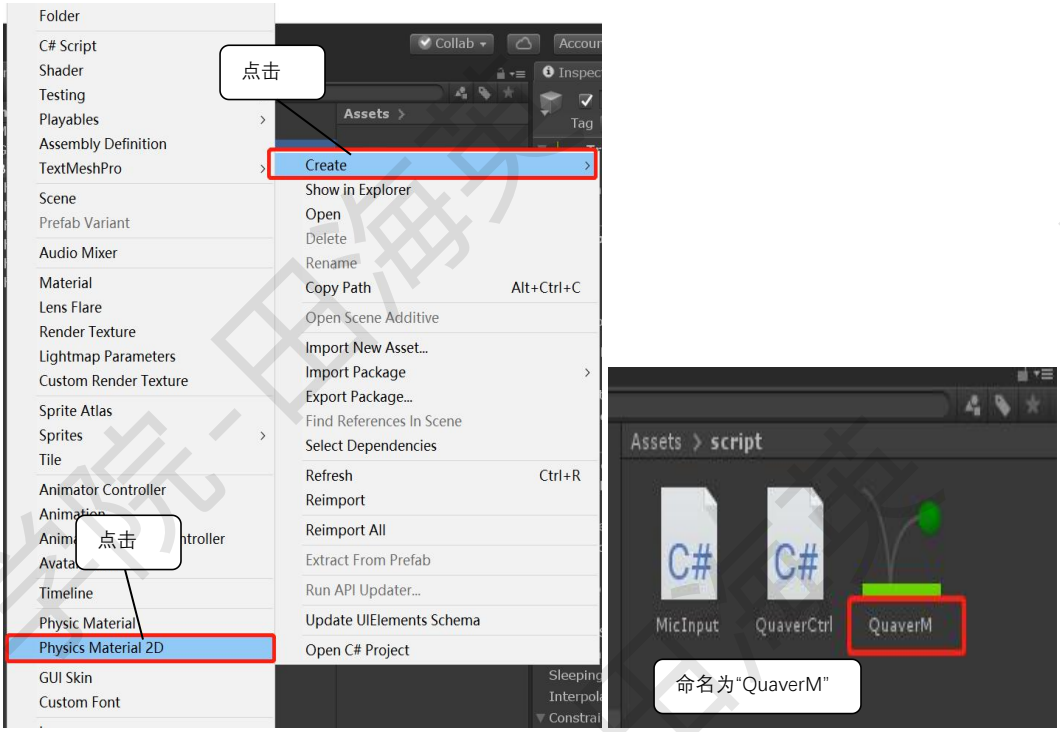


图 2-2-22

（2）将 QuaverM（摩擦系数）的 Friction 参数修改为 0.6，再将其拖拽至 Bird 及障碍物的 Inspector 面板中的 Material 参数框中。修改摩擦系数如图 2-2-22 所示，对象拖拽如图 2-2-4 所示。

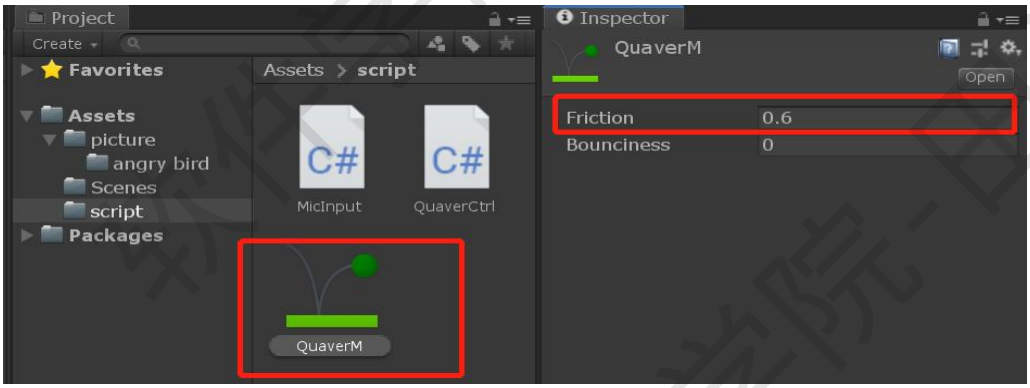


图 2-2-22

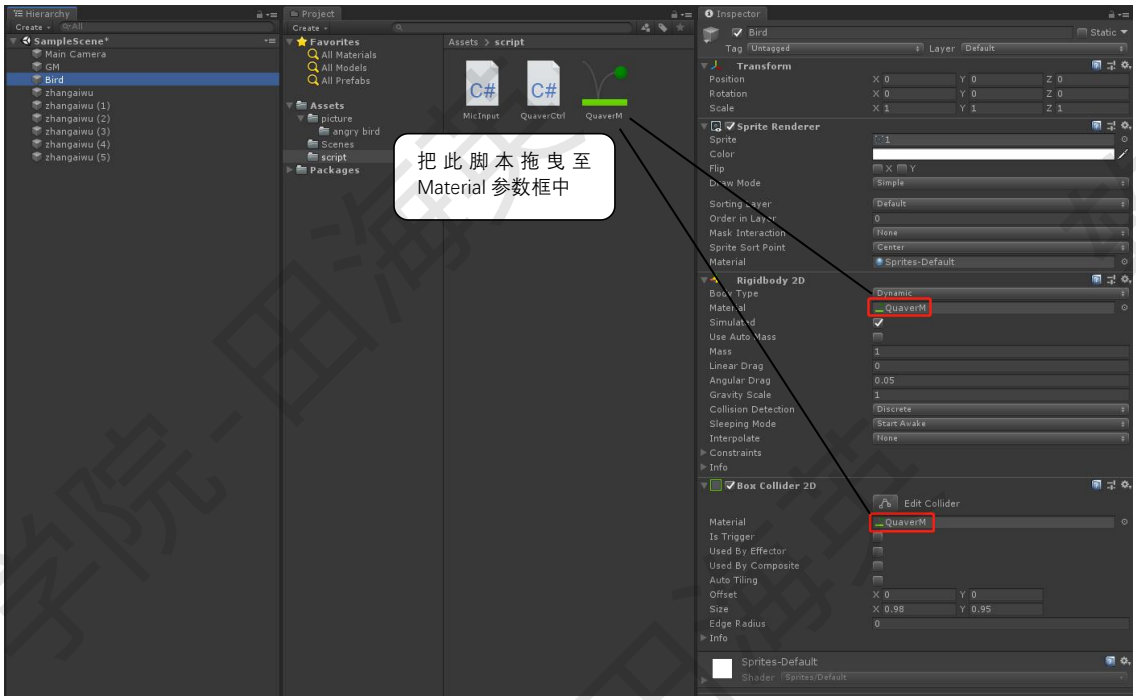
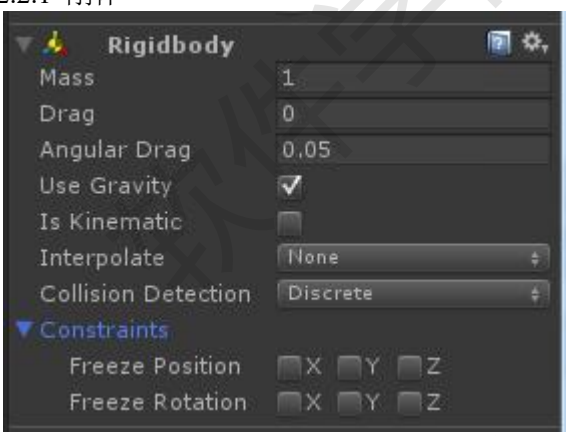


图 2-2-24

相关知识

在 Unity 场景中，物体要具有现实世界的物理属性，比如重力，弹力等，需要添给物体添加组件实现。

2.2.1 刚体



刚体（Rigidbody）组件让物体具有物理效果，带有刚体组件的物体会具有现实世界的重力效应，该组件包含多个属性信息：

Mass：物体的质量（默认单位为 kg），建议同一个场景中，物体之间的质量差不要大于 100 倍；

Drag：阻力，0 表示没有空气阻力，阻力越大，运动越慢，阻力极大时物体会停止运动；

Angular Drag ：物体在扭矩下旋转时承受的空气阻力。0 表示没有空气阻力，阻力极大时物体会停止运动；

Use Gravity：是否使用重力，开启此项，物体受重力影响；
Is Kinematic：是否开启动力学，如果启用，物体不再受物理引擎的影响只通过 Transform 组件完成操作；
Interpolate：插值，用于控制刚体运动的抖动情况；
Collision Detection：碰撞检测，避免高速运动的物体穿过其他对象而未发生碰撞，有三个选项：
✓ Discrete 离散碰撞检测，是默认值。
✓ Continuous 连续碰撞检测，该模式用于监测与动态碰撞体（带有 Rigidbody）的碰撞，使用连续碰撞检测模式检测与网格碰撞体（不带 Rigidbody）碰撞。
✓ Continuous Dynamic 连续动态碰撞检测，该模式用于检测与采用连续碰撞模式或连续动态碰撞模式物体的碰撞。
Constraints：刚体移动的约束。

2.2.2 碰撞体

碰撞体（Colliders）也属于物理组件，与刚体一起添加到物体上才能触发碰撞。
Box Collider: 盒碰撞体，可以选择各种形状；
Is Trigger: 触发器，是否触发事件；
Material：物理材质，选择不同类型的碰撞；
Center：碰撞器在物体的局部空间中的位置；
Radius：碰撞器的大小（半径大小）；

任务 2.2　设置游戏失败机制与重新开始机制

任务要求

小贝已经成功完成 2D 游戏的场景搭建意见 UI 设计和物理特性设置，需要进行游戏失败和重新开始的判断，需要完成如下工作内容：

(1) 设置游戏失败机制

(2) 设置游戏重新开始机制

任务实现

步骤 2.2.1 创建 DeadRoom

（1）创建一个空物体并命名为 DeadRoom。在 Hierarchy 面板上点击【鼠标右键】，接着点击【Create Empty】，如图 2-2-1 所示：

图 2-2-1

（2）在 Inspector 面板中选择 Select Icon，如图 2-2-2 所示修改 Icon：

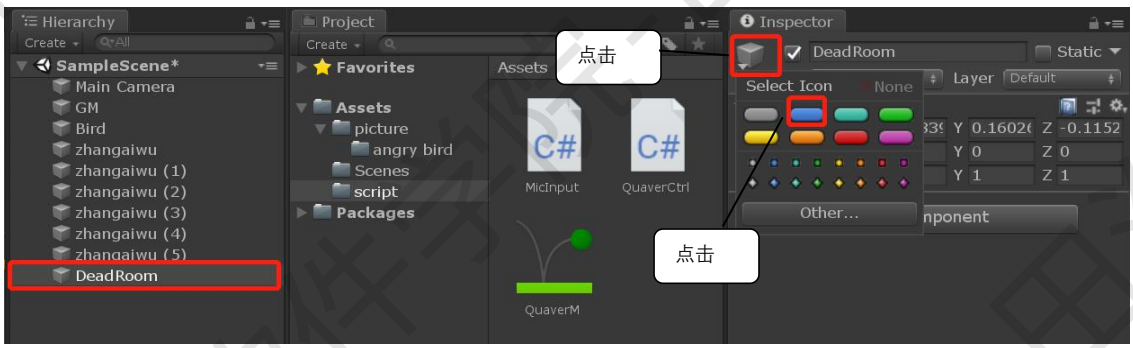


图 2-2-2

步骤 2.2.2 添加 Collider 属性

（1）将该物体移动到地形 UI 的下方，然后添加 Box Collider 2D。点击【Add Component】出现弹窗，再点击【Box Collider 2D】即属性添加成功，如若没有找到 Box Collider 2D，则点击【搜索框】进行查找，如图 2-2-2 所示：

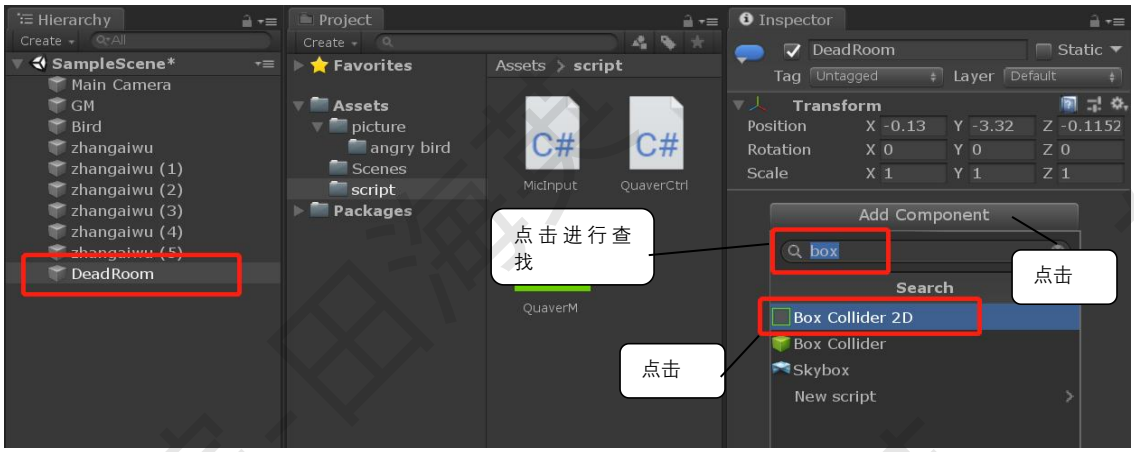


图 2-2-2

（2）点击勾选【Is Trigger】选项。如图 2-2-4 所示：

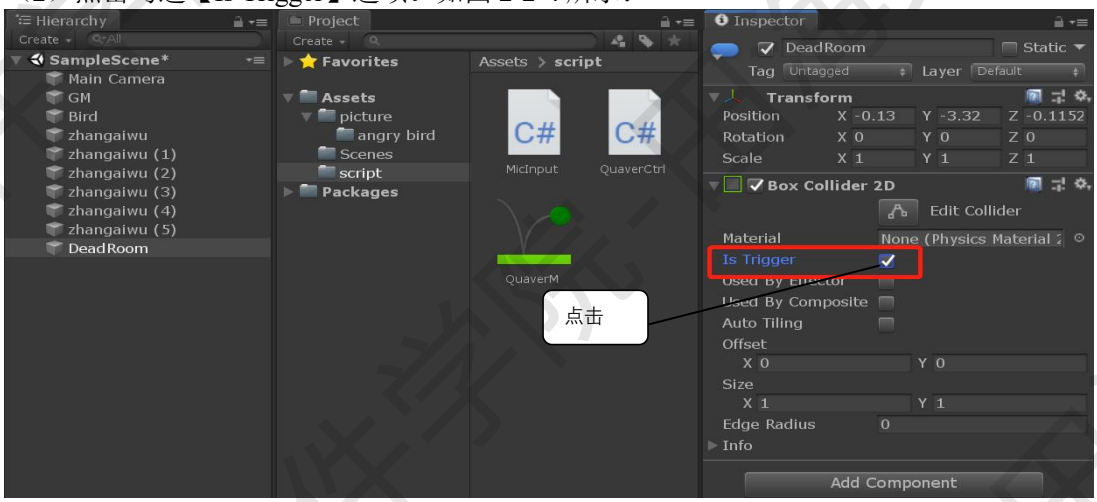


图 2-2-4

（2）在 Inspector 面板中选中点击【Edit Collider】，如图 2-2-5 所示。修改 Collider 大小，将碰撞框拉宽，碰撞框拉宽后的效果图如图 2-2-2 所示。

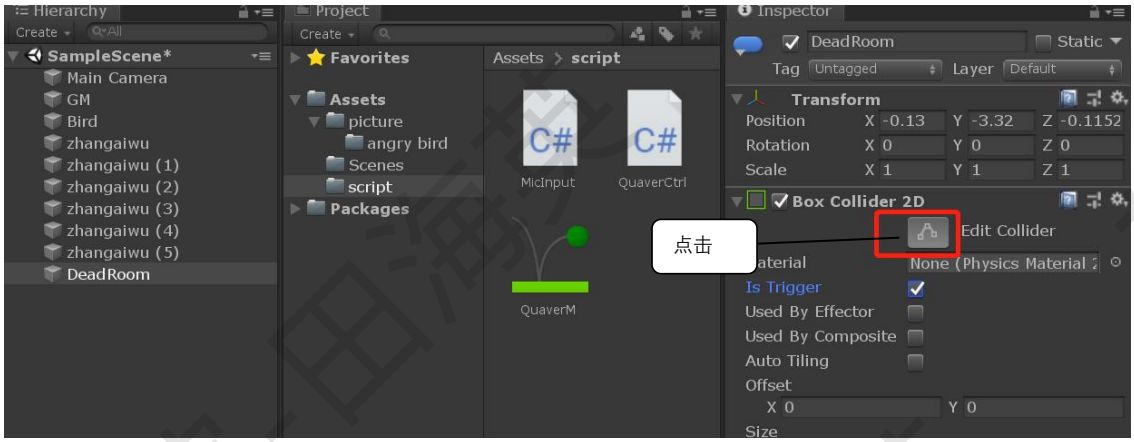


图 2-2-5

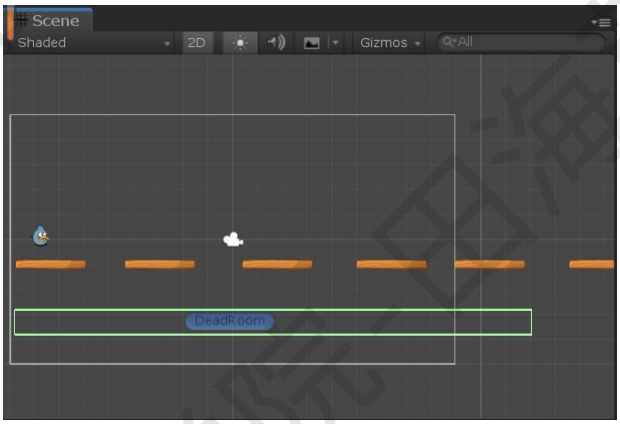


图 2-2-6

步骤 2.2.2 设置游戏重新开始机制

1.创建脚本 DelectDeath.cs

（1）当 Bird 从地形中间滑落时，游戏重新开始，创建脚本 DelectDeath，如图 2-2-4 所示。脚本 DelectDeath.cs 代码如图 2-2-4 所示：

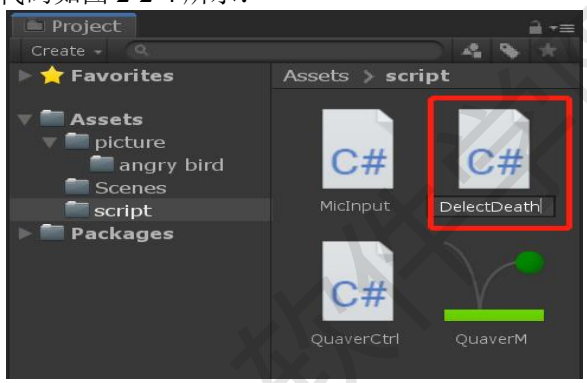


图 2-2-4

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; //添加此项，因为在游戏重新开始时要调用管理场景功能

0 references
public class DelectDeath : MonoBehaviour
{
    // Start is called before the first frame update
    0 references
    void Start()
    {

    }

    // Update is called once per frame
    0 references
    void Update()
    {

    }
    0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Player") {
            SceneManager.LoadScene (0);

        }
    } //添加Trigger函数
}
```

图 2-2-4

2.修改 Bird 属性

（1）由于脚本中设置的是碰到 tag “Player”，所以在 Inspector 面板中，需要对应地将 Bird 的“Tag”属性修改为 Player。在 Inspector 面板中点击【Tag】，接着点击【Player】如图 2-2-9 所示：

图 2-2-9

（2）将脚本文件绑定在 DeadRoom 上，如图 2-2-10 所示完成脚本拖拽。

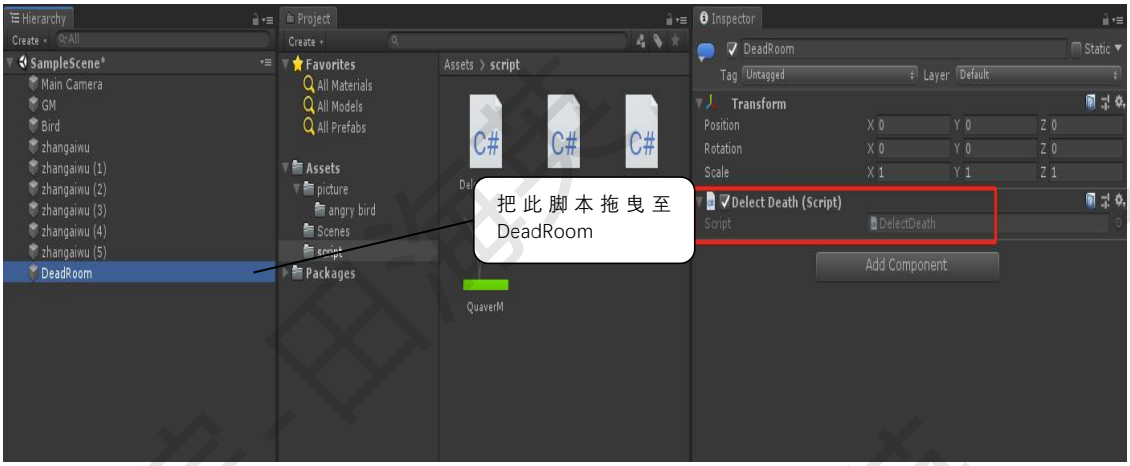


图 2-2-10

相关知识

2.2.1 添加碰撞体和刚体

Unity 环境模拟物理世界的效果，刚体和碰撞体是不可缺失的，具体细节请参考相关知识 2.2.1 和 2.2.2 部分，再次不再赘述。

2.2.2 C# 程序运行

C#程序必须依托物体起作用，所以，程序文件必须拖拽到物体上面，才起作用。

任务 2.4 实现相机跟随功能及障碍物制作

任务要求

“八分音符”的场景已经设计并完成搭建，要想看到物体的运动，必须有一双明亮的眼睛--摄像机，小贝需还需要完成：

（1）完成摄像机的部署

（2）添加场景中的障碍物

（3）发布游戏

任务实现

步骤 2.2.1 部署摄像机

1.创建脚本 CameraFefllow.cs

（1）为了使相机跟随 Bird 移动，需创建脚本 CameraFellow 如图 2-4-1 所示，CameraFellow.cs 具体代码如图 2-4-2 所示。

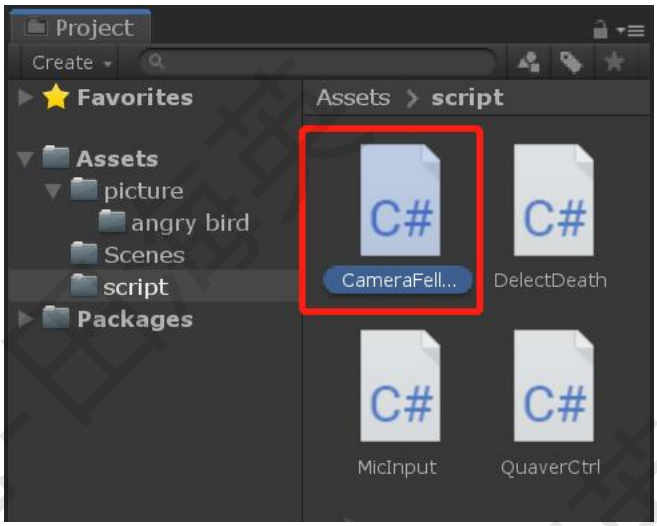


图 2-4-1

```
0 references
public class CameraFellow : MonoBehaviour
{
    1 reference
    public Transform player;
    // Start is called before the first frame update
    0 references
    void Start()
    {

    }

    // Update is called once per frame
    0 references
    void Update()
    {
        SetPos ();
    } //调用SetPos函数
    1 reference
    void SetPos()
    {
        transform.position = new Vector3 (player.position.x, transform.position.y, transform.position.z);
    } //将Bird的x值赋给相机，y值和z值不变
}
```

图 2-4-2

（2）将制作好的脚本文件绑定到 Main Camera 上，将 Bird 拖曳至脚本 Player 处，如图 2-4-3 所示完成脚本绑定及对象拖曳。

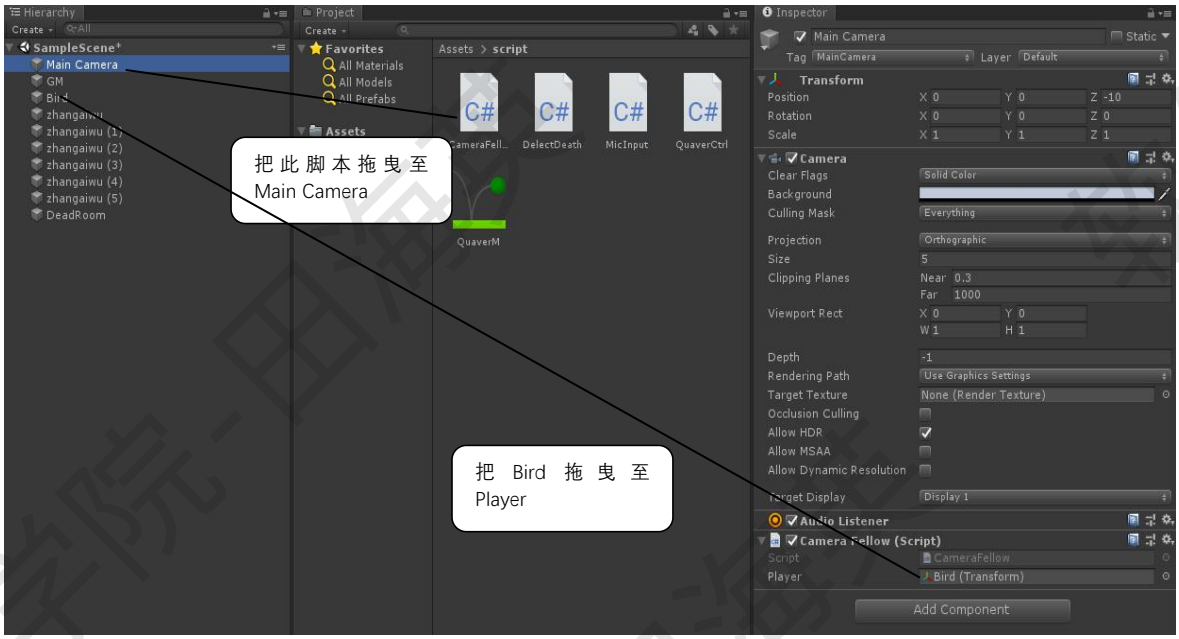


图 2-4-3

步骤 2.4.2 制作障碍物

1.创建障碍物

（1）在地形 UI 上设置障碍物，在 Hierarchy 面板中依此点击【2D Object --> Sprite 】，将 picture -->angry bird 文件夹中的图片拖曳至新建的 Sprite 上，如图 2-4-4 所示创建障碍物 UI。

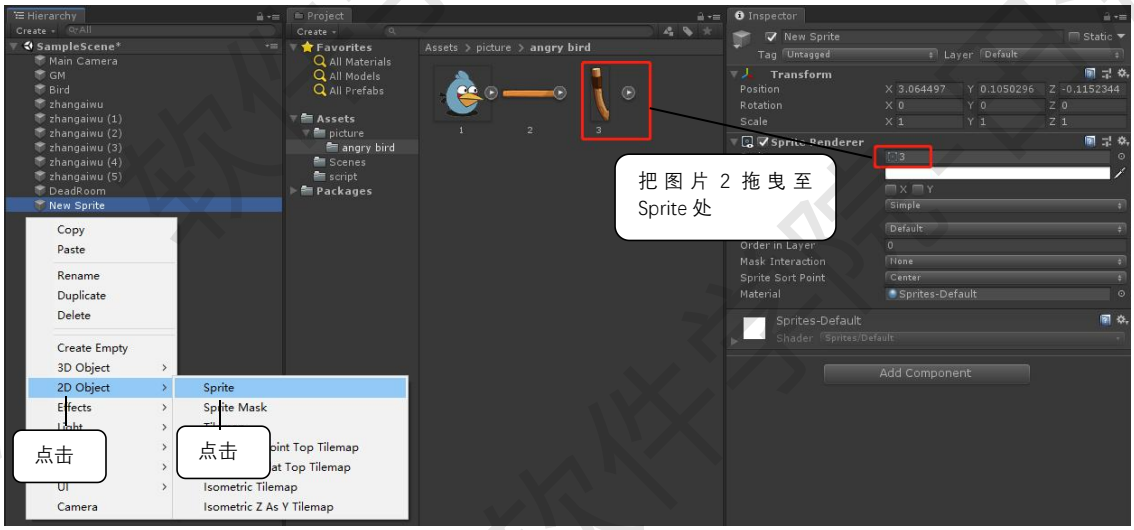


图 2-4-4

2.添加 Collider 属性

（1）添加 Box Collider 2D 属性，勾选点击【Is Trigger】选项。如图 2-4-5 所示：

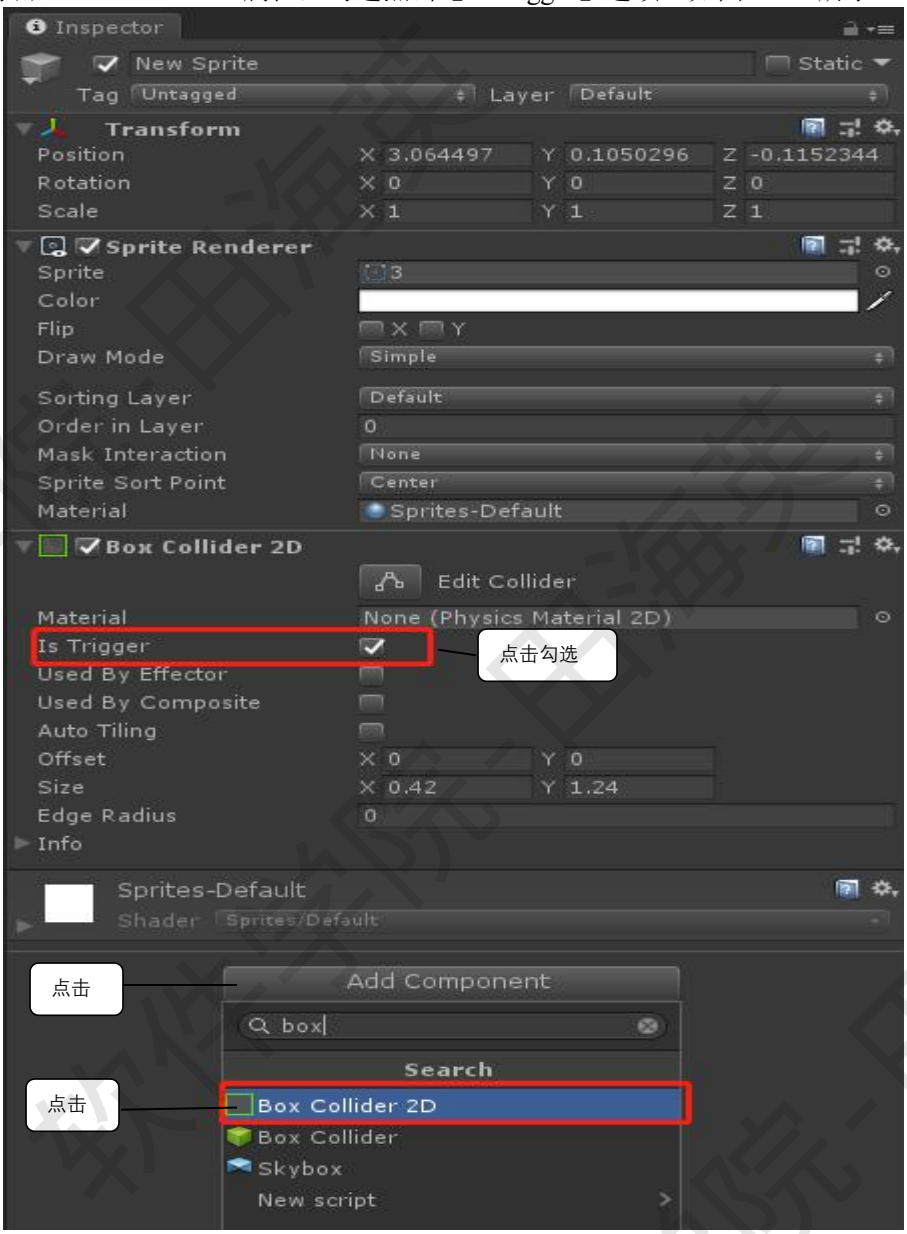


图 2-4-5

（2）将 DetectDeath 脚本拖曳至该障碍物上，如图 2-4-6 所示，调整障碍物到合适位置，复制创建多个障碍物，如图 2-4-7 所示。

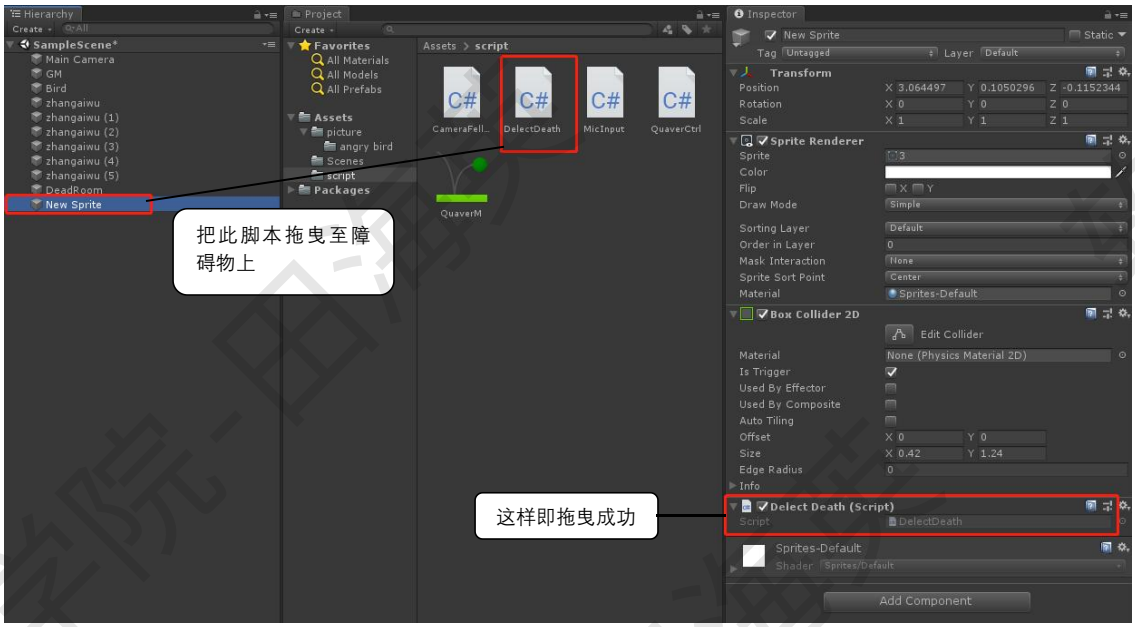


图 2-4-6

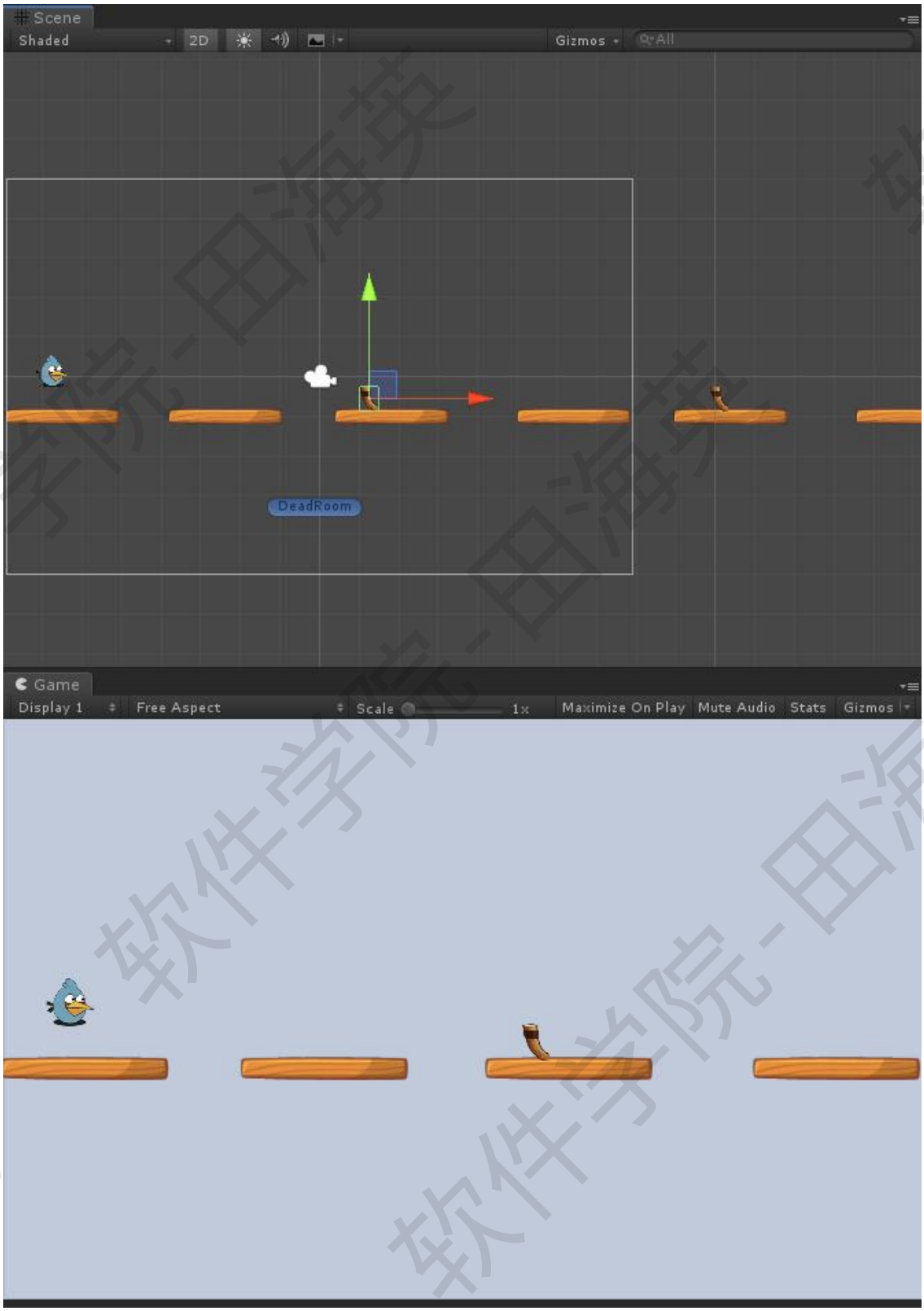

Scene
Shaded
2D
Gizmos
Q:All
DeadRoom
Game
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos

图 2-4-7

步骤 2.4.3 完成游戏

到这里已经完成游戏制作，我们可以试着运行一下。点击【开始】图标，即可进入游戏运行页面。如图 2-4-8 所示：

图 2-4-8

运行效果如图 2-4-9 所示：

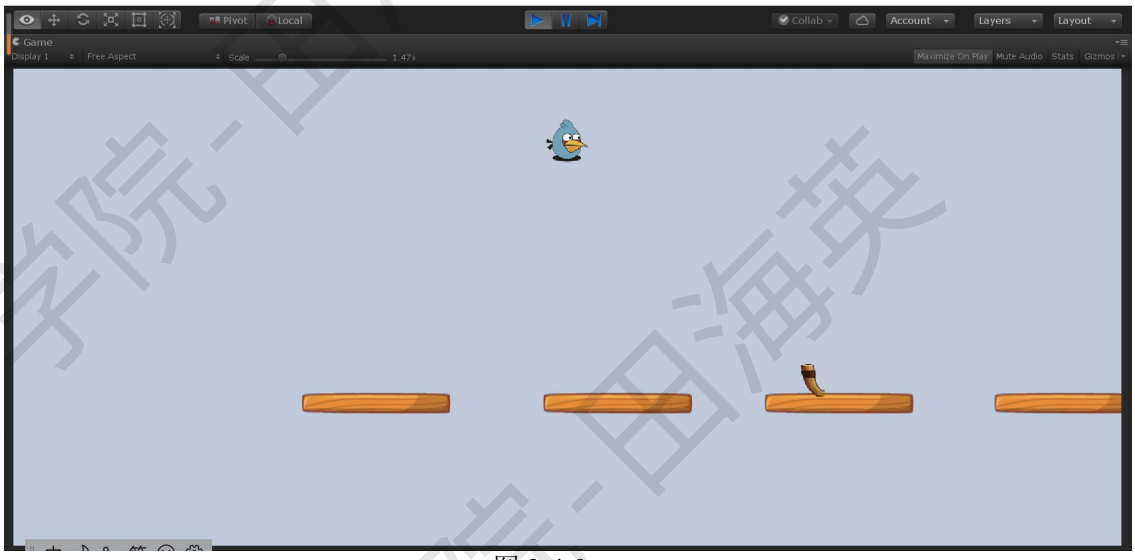


图 2-4-9

相关知识

2.4.1 Box Collider 2D

盒子碰撞器（Box Collider 2D）是 Unity 2D 中的碰撞器组件，用于检测物体之间的碰撞情况。

Material：物理材质，可用于确定碰撞的属性（例如摩擦和弹性）。
Is Trigger： 2D 盒型碰撞体作为触发器运行，请选中此框。
Used by Effector： 2D 盒型碰撞体由附加的 2D 效应器组件使用，请选中此框。
Used by Composite：碰撞体由附加的 2D 复合碰撞体 (Composite Collider 2D) 使用，请勾选此复选框。
Auto Tiling： 如果所选精灵的精灵渲染器 (Sprite Renderer) 组件将 Draw Mode 设置为 Tiled__，请勾选此复选框。如果没有启用 Auto Tiling__，2D 碰撞体几何形状不会自动重复。
Offset：设置 2D 碰撞体几何形状的局部偏移。
Size：按局部空间单位设置盒体的大小。
Edge Radius：控制边缘周围的半径，使顶点为圆形。

知识检验

1. 在自己的电脑上安装好 Unity Hub 及 Unity
2. 按照项目二的操作步骤，完成“八分音符”的项目创建