**河北工业大学**

《**计算机游戏程序设计**》

**实验报告**

**专业：** 软件工程

**班级：** 软件153班

**姓名：**

**学号：**

# 实验四 应用物理引擎实现游戏

**一、实验目的与要求**

1. 熟悉并掌握刚体及使用方法。

2. 熟悉并掌握碰撞器及使用方法。

3. 熟悉并掌握射线及使用方法。

4. 熟悉并掌握关节及使用方法。

**二、实验内容**

**1、打箱子小游戏**

**程序清单：**

public class shoot : MonoBehaviour {

public GameObject CubeBox;

public Texture CoursorTexture; //准心贴图

void Start()

{

//使用循环添加4\*4的立方体墙，用于击打

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

GameObject objCube = (GameObject)Instantiate(CubeBox);

objCube.AddComponent<AutoDestroy>(); //添加销毁的脚本

objCube.transform.position = new Vector3(-2.0f + i, 2f + j, 4f);

}

}

}

void Update()

{

if (Input.GetMouseButtonDown(0))

{

Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

RaycastHit rayHit;

if (Physics.Raycast(ray, out rayHit))

{

Vector3 position = rayHit.point;

GameObject bullet = GameObject.CreatePrimitive(PrimitiveType.Sphere);

bullet.AddComponent<Rigidbody>();

bullet.AddComponent<AutoDestroy>();

bullet.transform.position = Camera.main.transform.position;

bullet.GetComponent<Rigidbody>().AddForce((position - bullet.transform.position) \* 500); //添加指定方向、指定大小的力

}

}

}

void OnGUI()

{

Vector3 mousePos = Input.mousePosition;

//绘制准心

Rect rect = new Rect(mousePos.x - CoursorTexture.width / 2, Screen.height - mousePos.y - CoursorTexture.height / 2, CoursorTexture.width, CoursorTexture.height);

GUI.DrawTexture(rect, CoursorTexture);

}

}

public class AutoDestroy : MonoBehaviour

{

// Use this for initialization

void Start()

{

}

// Update is called once per frame

void Update()

{

}

void OnBecameInvisible()

{

Destroy(this.gameObject);

}

}

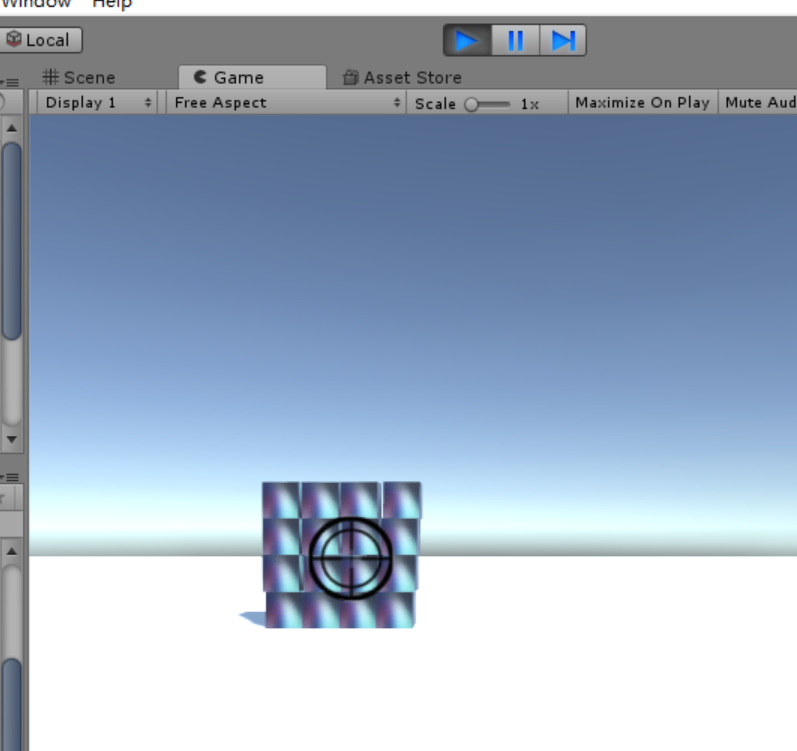


图1.1 打箱子

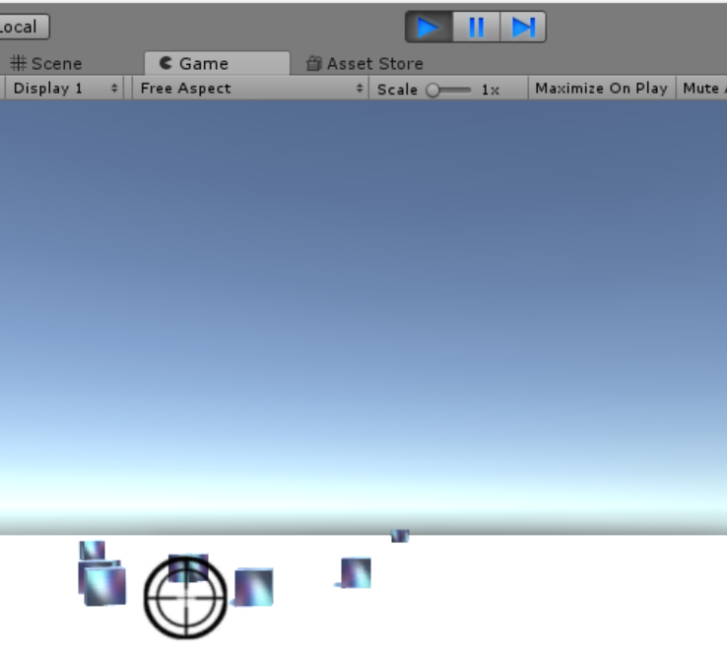


图1.2 击打

**2、关节**

void Start()

{

//获取被关联的对象。即在cube上增加关节

connectobj = GameObject.Find("Cube(1)");

obj = GameObject.Find("Cube");

}

void OnGUI()

{

if (GUILayout.Button("添加链条关节"))

{

ResetJoint();

jointComponent = obj.AddComponent<HingeJoint>();//给cube增加链条关节

HingeJoint hjoint = (HingeJoint)jointComponent;

connectobj.GetComponent<Rigidbody>().useGravity = true; //受重力

hjoint.connectedBody = connectobj.GetComponent<Rigidbody>();//cube(1)作为被关联的对象

}

if (GUILayout.Button("添加固定关节"))

{

ResetJoint();

jointComponent = obj.AddComponent<FixedJoint>();//给cube增加固定关节

FixedJoint fjoint = (FixedJoint)jointComponent;

connectobj.GetComponent<Rigidbody>().useGravity = true;

fjoint.connectedBody = connectobj.GetComponent<Rigidbody>();

}

if (GUILayout.Button("添加弹簧关节"))

{

ResetJoint();

jointComponent = obj.AddComponent<SpringJoint>();//给cube增加弹簧关节

SpringJoint sjoint = (SpringJoint)jointComponent;

connectobj.GetComponent<Rigidbody>().useGravity = true;

sjoint.connectedBody = connectobj.GetComponent<Rigidbody>();

}

if (GUILayout.Button("添加角色关节"))

{

ResetJoint();

jointComponent = obj.AddComponent<CharacterJoint>();//给cube增加角色关节

CharacterJoint cjoint = (CharacterJoint)jointComponent;

connectobj.GetComponent<Rigidbody>().useGravity = true;

cjoint.connectedBody = connectobj.GetComponent<Rigidbody>();

}

if (GUILayout.Button("添加可配置关节"))

{

ResetJoint();

jointComponent = obj.AddComponent<ConfigurableJoint>();//给cube增加可配置关节

ConfigurableJoint cojoint = (ConfigurableJoint)jointComponent;

connectobj.GetComponent<Rigidbody>().useGravity = true;

cojoint.connectedBody = connectobj.GetComponent<Rigidbody>();

}

}

private void ResetJoint()

{

Destroy(jointComponent);//销毁之前添加的所有关节组件

obj.transform.position = new Vector3(0.0f, 10.0f, 0.0f);

connectobj.gameObject.transform.position = new Vector3(0.0f, 20.0f, 0.0f);

connectobj.GetComponent<Rigidbody>().useGravity = false; //不感应重力

obj.GetComponent<Rigidbody>().useGravity = false;

}

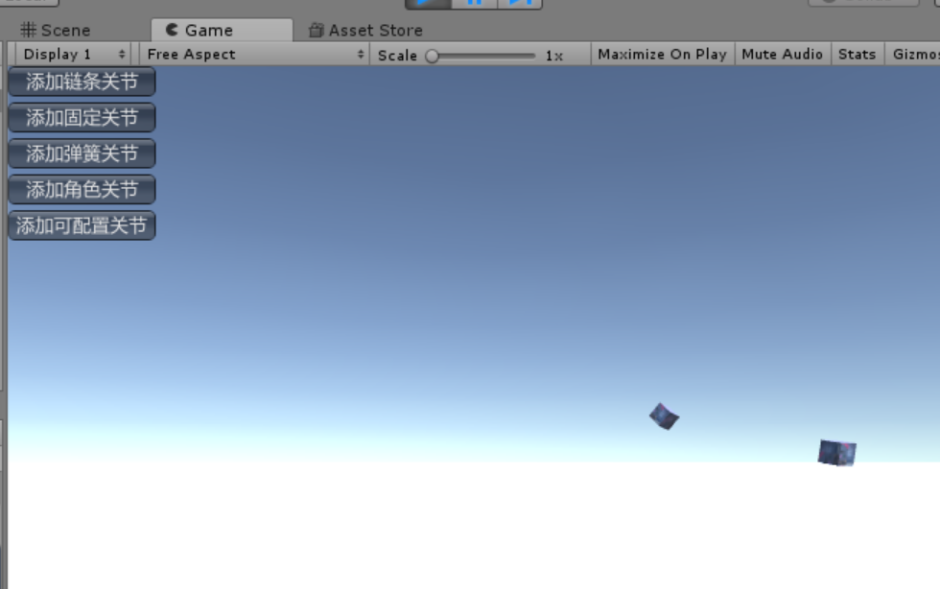


图2.1 弹簧关节

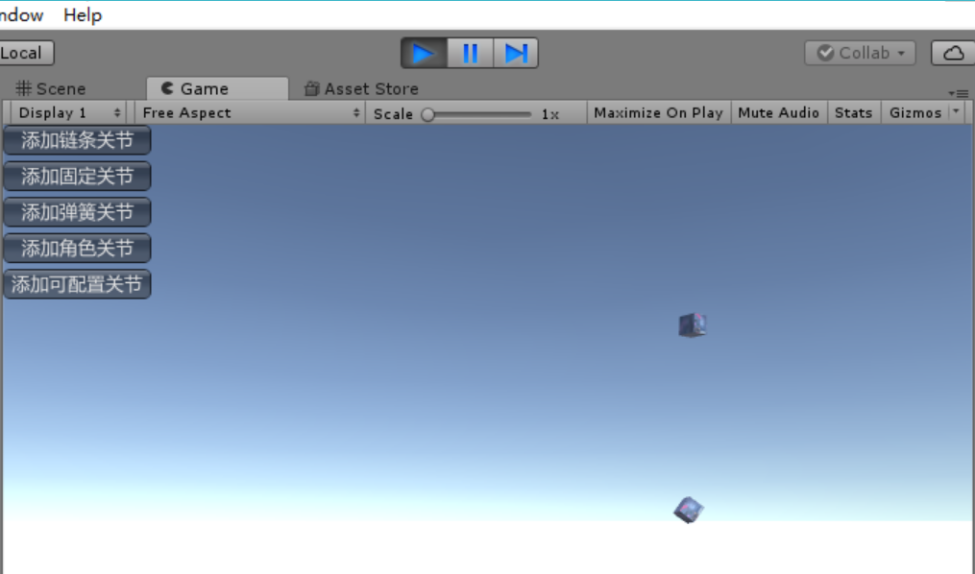


图 2.2 固定关节