## 习题

**1.解释一下rigidbody组件中is Kinematic参数在什么情况下使用？**

1）isKinematic属性是确定刚体是否接受动力学模拟，此影响不仅包括重力感应，还包括速度、阻力、质量等的物理模拟

2）这个属性用于控制物理引擎效果是否会在生效，如果这个属性为True，那么所有通过物理引擎给物体施加外力和设置速度等等就不会再有任何效果了。例如如果想做一个让NPC受击浮空之后，在空中停留的效果，就可以通过设置这个变量来实现，只要该变量为true，物理的重力效果就不会生效，NPC并不会自动往下做自由落体运动，而会在空中保持之前的动画状态，位置也不会发生改变。

**2.编写一个脚本对刚体的几种常用方法进行测试**

实例演示：下面通过实例演示useGravity、isKinematic和velocity的使用。

主要代码如下：

public class GraAndKin\_ts : MonoBehaviour

{

public Rigidbody A, B;

string str\_AG = "";

string str\_AK = "";

string str\_BK = "";

Vector3 v1, v2;

void Start()

{

//为了更好的演示，将重力加速度降低

Physics.gravity = new Vector3(0.0f,-0.5f,0.0f);

A.useGravity = false;

B.useGravity = false;

A.isKinematic = false;

B.isKinematic = false;

str\_AG = "A开启重力感应";

str\_AK = "A关闭物理感应";

str\_BK = "B关闭物理感应";

v1 = A.position;

v2 = B.position;

}

void OnGUI()

{

if (GUI.Button(new Rect(10.0f, 10.0f, 200.0f, 45.0f), str\_AG))

{

if (A.useGravity)

{

A.useGravity = false;

str\_AG = "A开启重力感应";

}

else

{

A.useGravity = true;

str\_AG = "A关闭重力感应";

}

}

if (GUI.Button(new Rect(10.0f, 60.0f, 200.0f, 45.0f), str\_AK))

{

if (A.isKinematic)

{

A.isKinematic = false;

str\_AK = "A关闭物理感应";

}

else

{

A.isKinematic = true;

str\_AK = "A开启物理感应";

}

}

if (GUI.Button(new Rect(10.0f, 110.0f, 200.0f, 45.0f), str\_BK))

{

if (B.isKinematic)

{

B.isKinematic = false;

str\_BK = "B关闭物理感应";

}

else

{

B.isKinematic = true;

str\_BK = "B开启物理感应";

}

}

if (GUI.Button(new Rect(10.0f, 160.0f, 200.0f, 45.0f), "重置"))

{

A.position = v1;

A.rotation = Quaternion.identity;

B.position = v2;

B.rotation = Quaternion.identity;

}

}

**3.了解Unity3D游戏引擎自带的规则碰撞器，并导入一个模型为其添加合适的碰撞器**

一：碰撞器类型碰撞体的

类型包括6中：

1.盒子碰撞器

2.球体碰撞器

3.胶囊碰撞器

4.网络碰撞器

5.车轮碰撞器

6.地形碰撞器  
演示代码：  
public class TestCollider : MonoBehaviour {  
    public Transform cube;//将要碰撞到的GameObject的transform  
    public float speed = 1.0f;  
    public Vector3 dir = Vector3.zero;  
    void Start() {  
        if(cube){  
            dir = cube.position - transform.position;  
            dir = dir.normalized;  
        }  
         
    }  
    void Update() {  
        transform.Translate(dir \* Time.deltaTime \* speed)；  
    }  
  
    void OnCollisionEnter(Collision collisionInfo)  
    {  
        Debug.Log("碰撞到的物体的名字是：" + collisionInfo.gameObject.name);  
    }  
}

**4.在场景中新建物理材质，实现小球从高空落下可弹起的功能。导出package文件。**

package com.example.sft;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.RadialGradient;

import android.graphics.Shader;

import android.util.AttributeSet;

import android.util.Log;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

import android.view.View;

public class TestSurfaceView extends SurfaceView implements SurfaceHolder.Callback{

private Paint cp;

private List<KCircle> KCList;

private Random r;

public TestSurfaceView(Context context, AttributeSet attrs) {

super(context, attrs);

// TODO Auto-generated constructor stub

getHolder().addCallback(this);

cp=new Paint();

cp.setColor(Color.BLACK);

KCList=new ArrayList<KCircle>();

r=new Random();

}

@Override

public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {

// TODO Auto-generated method stub

}

private int getRandomColor(){

return r.nextInt()%0xff000000;

}

@Override

public void surfaceCreated(SurfaceHolder arg0) {

// TODO Auto-generated method stub

}

public void run(){

new MyThread().start();

}

@Override

public void surfaceDestroyed(SurfaceHolder arg0) {

// TODO Auto-generated method stub

}

@Override

public boolean onTouchEvent(MotionEvent event) {

// TODO Auto-generated method stub

if (event.getAction() != MotionEvent.ACTION\_DOWN &&

event.getAction() != MotionEvent.ACTION\_MOVE) {

return false;

}

KCList.add(new KCircle(getRandomColor(),(int)event.getX(),(int)event.getY()));

return true;

}

class MyThread extends Thread{

@Override

public void run() {

// TODO Auto-generated method stub

while(true){

Canvas c=getHolder().lockCanvas();

c.drawPaint(cp);

for(int i=0;i<KCList.size();i++){

KCList.get(i).moveAndDraw(c);

if(KCList.get(i).y>getHeight()){

KCList.remove(i);

}

}

try {

sleep(15);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

getHolder().unlockCanvasAndPost(c);

}

}

}

}

package com.example.sft;

import android.graphics.Canvas;

import android.graphics.Paint;

import android.graphics.RadialGradient;

import android.graphics.Shader;

public class KCircle {

private int color;

public int x;

public int y;

private Paint p;

private int radius;

int v;

int a;

public KCircle(int color){

this.color=color;

x=0;

y=0;

v=0;

a=2;

radius=25;

p=new Paint();

p.setColor(color);

RadialGradient rg=new RadialGradient(x+10, y-10, 25,color+(0xffffffff-color)/2, color, Shader.TileMode.CLAMP);

p.setShader(rg);

p.setAntiAlias(true);

}

public KCircle(int colcr,int x,int y){

this(colcr);

this.x=x;

this.y=y;

}

public void moveAndDraw(Canvas c){

y+=v;

v+=a;

draw(c);

}

public void moveTo(int x,int y){

this.x=x;

this.y=y;

}

public void draw(Canvas c){

RadialGradient rg=new RadialGradient(x+10, y-10, 25,0xffdddddd, color, Shader.TileMode.CLAMP);

p.setShader(rg);

c.drawCircle(x, y, radius, p);

}

}