吴博文

2017年3月10日

3D Homework

一、在 Unity 中，实现运动的常见手段有哪些（自少四种）。请用一个物体的运动，用不同不同方法实现。（如：修改Transform属性，使用向量Vector3的方法…）

1、直接修改Transorm属性

```
    this.transform.position += Vector3.right;
```

2、使用vector3的moveToward方法

```
transform.position = Vector3.MoveTowards(transform.position,
                                         target.position, step);
```

3、使用transform的rotate方法实现旋转

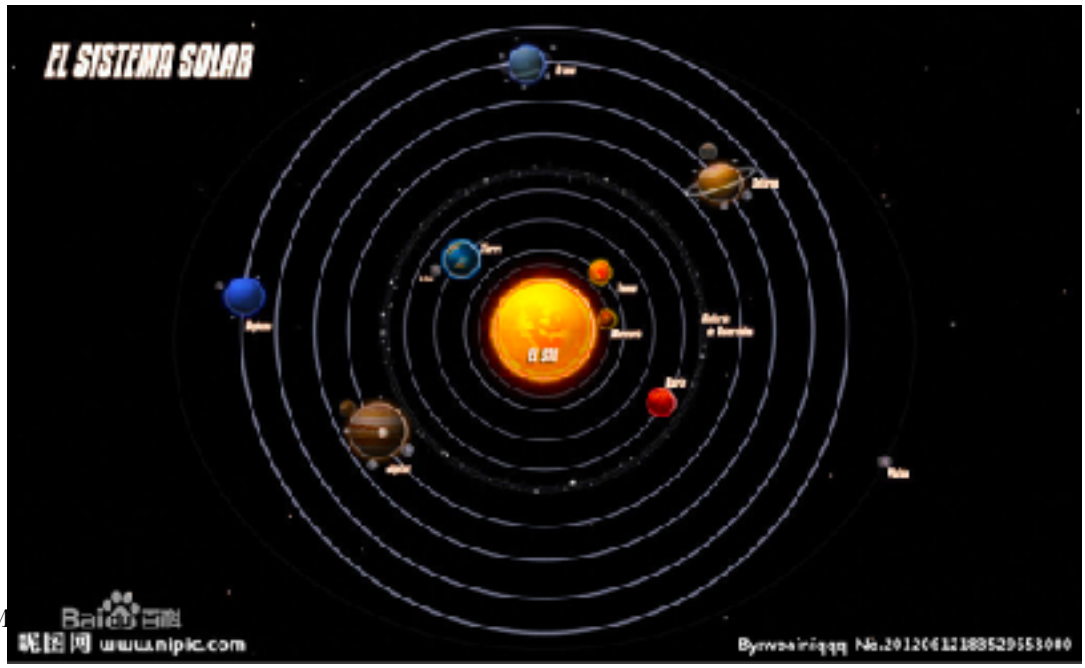
```
        transform.Rotate (Vector3.right);
```

4、使用Quaternion的方法实现旋转

```
transform.rotation = Quaternion.RotateTowards(
                transform.rotation, target.rotation, step);
```

二、写一个程序，实现一个完整的太阳系， 其他星球围绕太阳的转速必须不一样，且不在一个法平面

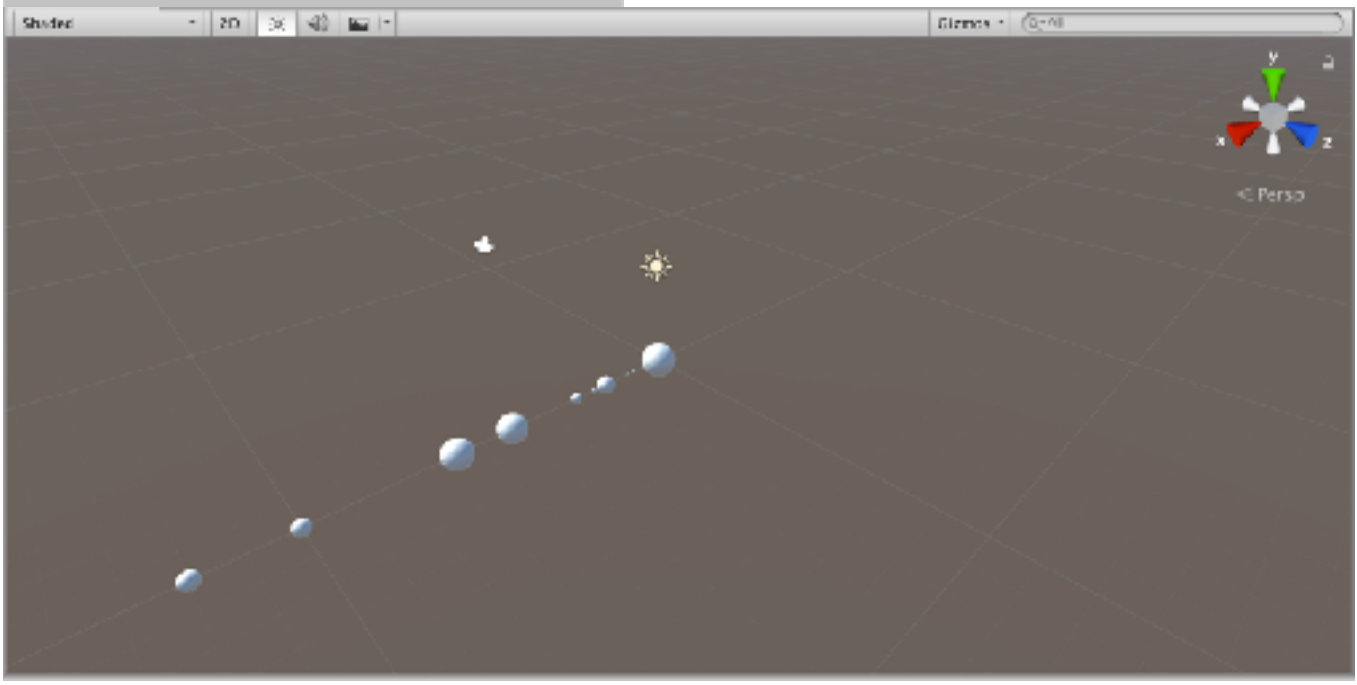
1、第一步 查阅太阳系行星资料



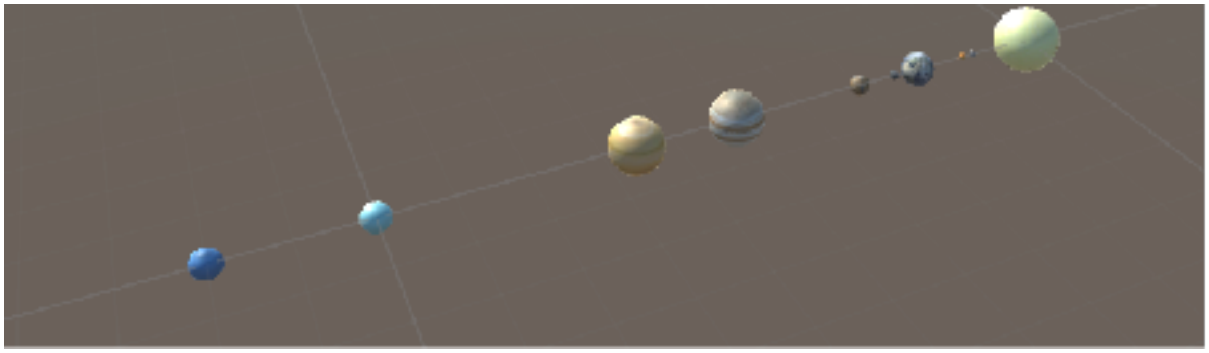
2、创建八大行星和太阳 还有地球卫星的初始游戏对象，并按照感觉设置其相对位置。

游戏对象组织结构，父子之间的关系表示子对象围绕父对象旋转

设置相对位置，这里只是凭感觉设置，毕竟真正的天文轨道复杂许多



3、对球面进行贴图



4、添加脚本

全部球体使用相同的脚本，由于并不知道具体八大行星轨道的法线是多少，所以就采用随机生成的方式。同时为了确保球体是围绕圆心进行运动，法向量应该为（0，y，z）。因为球体初始位置为（x，0，0）。为什么这样做呢?

证明：

球体轨迹上任意一点与球心形成的向量应时刻与法向量保持垂直，从向量运算的角度讲，就是两向量点乘为0。即（0,y,z）点乘 （x，0，0）等于 0。所以说这个理论是正确的。

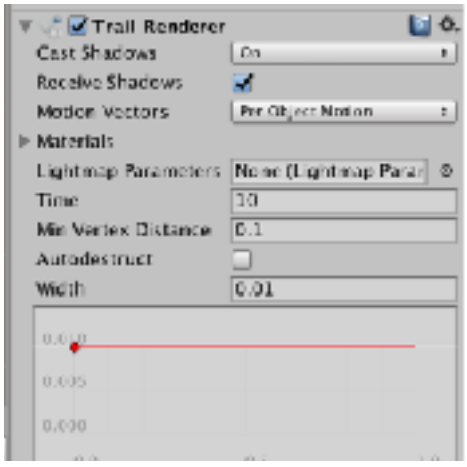
脚本如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Revolve : MonoBehaviour {

	public float speed;
	private Vector3 axis;
	// Use this for initialization
	void Start () {
		axis.Set(0, Random.Range (1, 20), Random.Range (1,
20));
	}

	// Update is called once per frame
	void Update () {
		transform.RotateAround
(this.transform.parent.position, axis, speed);
	}
}
```

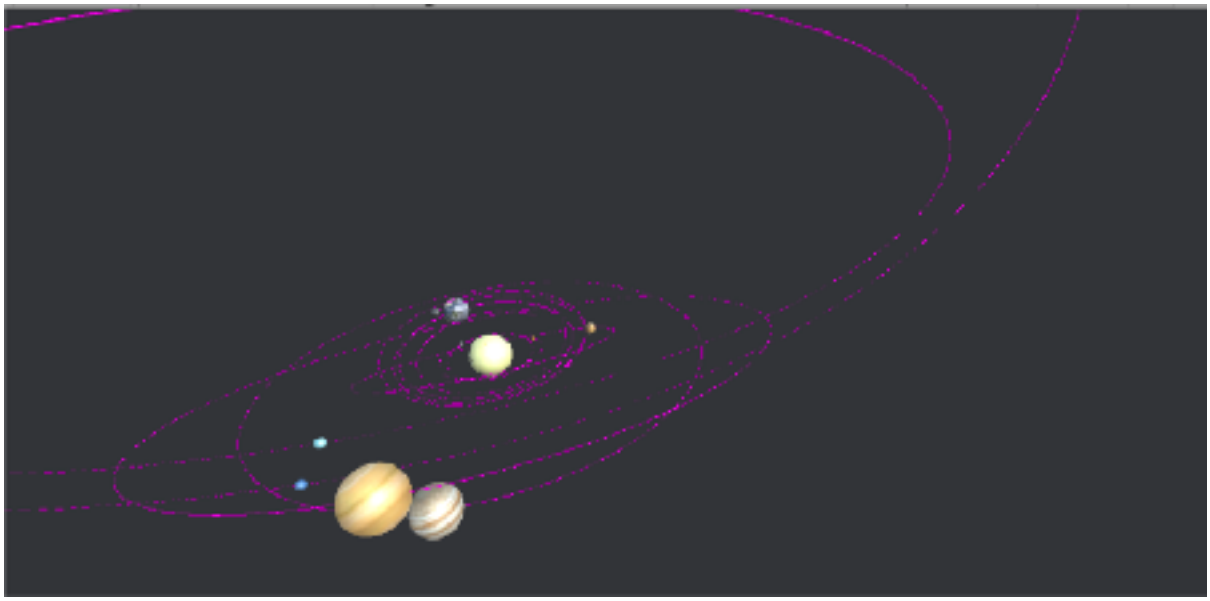
其中将speed设为公有变量，方便调整

5、添加轨迹

AddComponent->Effects->Trail Renderer



6、完成!



三、编程实践

Priests and Devils

```
Priests and Devils is a puzzle game in which you will help the
Priests and Devils to cross the river within the time limit.
There are 3 priests and 3 devils at one side of the river.
They all want to get to the other side of this river, but
there is only one boat and this boat can only carry two
persons each time. And there must be one person steering the
boat from one side to the other side. In the flash game, you
can click on them to move them and click the go button to move
the boat to the other direction. If the priests are out
numbered by the devils on either side of the river, they get
killed and the game is over. You can try it in many ways. Keep
all priests alive! Good luck!
```

游戏中提到的事物：魔鬼 牧师 船 两岸

行为表

行为	条件
开船	船上至少有一人
牧师上船	船上有空位
魔鬼上船	船上有空位

行为	条件
左边下船	左边船上有乘客
右边下船	右边船上有乘客
游戏胜利	6人均抵达对岸
游戏失败	一边的人数不均

根据行为表 定义interface IUserAction 所提供的接口

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public interface IUserAction {
//    void reset();
      void drive();

      void priest_to_boat_at_begin();
      void devil_to_boat_at_begin();
      void priest_to_boat_at_end();
      void devil_to_boat_at_end();

      void left_off_boat();
      void right_off_boat();
}
```

这些接口由genGameObject实现，以实现UI和模型之间的通讯。

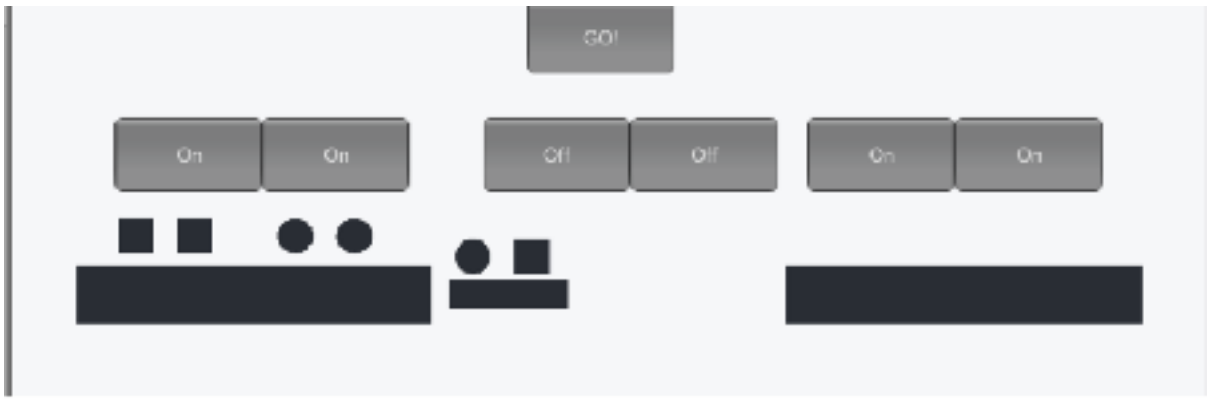
在genGameObject中使用Queue管理在各个区域的游戏对象

```
      private Queue<GameObject> priest_begin;
      private Queue<GameObject> priest_end;

      private Queue<GameObject> devil_begin;
      private Queue<GameObject> devil_end;

      private Queue<GameObject> onTheWay;
      private GameObject ship;
```

在updata函数中通过setPostion使得各个collection中的游戏对象都在正确的位置：

```
        void setPostion(Queue<GameObject> obj_queue, Vector3 first, Vector3 move) {
            GameObject[] temp = obj_queue.ToArray();
            for (int i = 0; i < temp.Length; ++i) {
                temp[i].transform.localPosition = first + i * move;
            }
        }
```

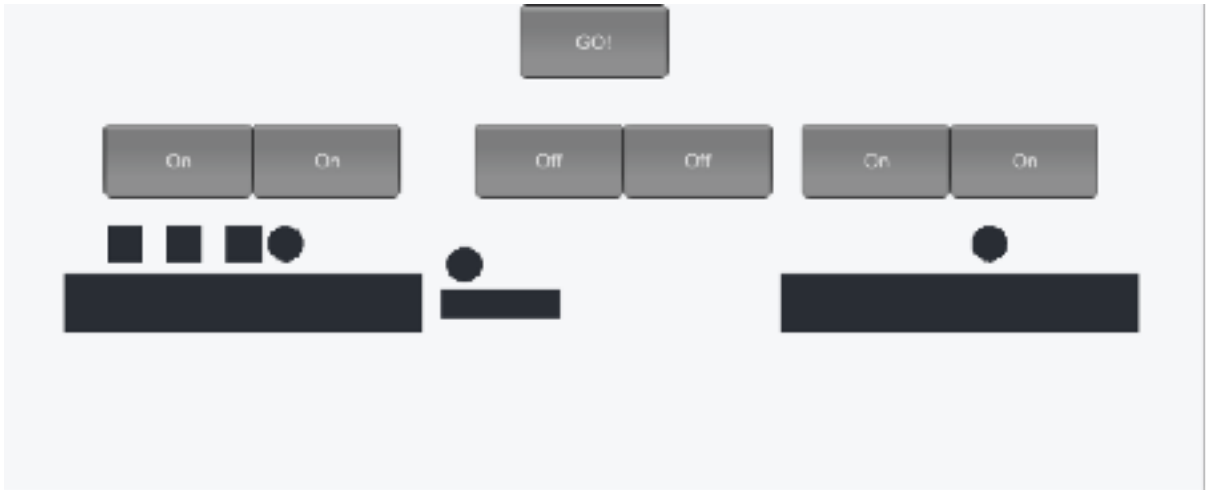
至此已经基本实现牧师和魔鬼上下船的操作，效果如下：



接下来就是实现船的移动
在update中添加以下代码，以实现船的移动：

```
if (ship_info == ship_location.moving_to_end) {
    float step = 3 * Time.deltaTime;
    ship.transform.position = Vector3.MoveTowards(ship.transform.position, ship_end_position,
    if (ship.transform.position == ship_end_position)
        ship_info = ship_location.end;
} else if (ship_info == ship_location.moving_to_begin) {
    float step = 3 * Time.deltaTime;
    ship.transform.position = Vector3.MoveTowards(ship.transform.position, ship_begin_positi
    if (ship.transform.position == ship_begin_position)
        ship_info = ship_location.begin;
}
```

经过测试，可以正常开船， 上船和下船，效果如下图：



基本行为已经完成，接下来补充游戏规则和添加reset方法，实现胜利和gameover

其中通过一个GenGameObject中的公有变量表示当前游戏的状态，实现模型和UI之间的交流。

```
    void count_num_of_object() {
        num_of_devil_at_the_begin = devil_begin.Count;
        num_of_devil_at_the_end = devil_end.Count;
        num_of_priest_at_the_begin = priest_begin.Count;
        num_of_priest_at_the_end = priest_end.Count;
        GameObject[] temp = onTheWay.ToArray();
        foreach (GameObject item in onTheWay)
        {
            if (item.tag == "priest") {
                if (ship_info == ship_location.begin) {
                    num_of_priest_at_the_begin++;
                } else {
                    num_of_priest_at_the_end++;
                }
            } else {
                if (ship_info == ship_location.begin) {
                    num_of_devil_at_the_begin++;
                } else {
                    num_of_devil_at_the_end++;
                }
            }
        }
    }
    private void check() {
        count_num_of_object();
        // check at the begin
        if (num_of_devil_at_the_begin > num_of_priest_at_the_begin) {
            game_state = State.over;
        } else if (num_of_devil_at_the_end > num_of_priest_at_the_end) {
            game_state = State.over;
        } else if (num_of_devil_at_the_end + num_of_priest_at_the_end == 6) {
            game_state = State.win;
        }
    }

    public void reset() {
        Application.LoadLevel(Application.loadedLevelName);
        game_state = State.normal;
    }
```

基本功能完成。
总结：界面还比较丑，上下船移动不完善，代码有些冗余，但是MVC架构还是正确的。