# #3d游戏第三次作业

##1、简答并用程序验证【建议做】

* 游戏对象运动的本质是什么？

游戏对象运动的本质是游戏对象每一帧在空间的变化，即坐标的变换。空间变换包含了游戏对象中的transform中的position和rotation属性，其中一个是绝对或相对位置的改变，一个是所处位置的角度的旋转变换。

* 请用三种方法以上方法，实现物体的抛物线运动。（如，修改Transform属性，使用向量Vector3的方法…）
  1. 利用transform中的position属性方法直接改变位置，水平方向匀速移动，竖直方向有加速度。

```javascript

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ParaCruve1 : MonoBehaviour

{

public float horizontalSpeed = 5.00f;

public float verticalSpeed = 0.00f;

const float g = 0.02f;

// Use this for initialization

void Start()

{

Debug.Log("start!");

}

// Update is called once per frame

void Update()

{

this.transform.position += Vector3.right \* Time.deltaTime \* horizontalSpeed;

this.transform.position += Vector3.down \* Time.deltaTime \* verticalSpeed;

verticalSpeed += g;

}

}

```

* 1. 新建一个Vector3变量change，水平方向数值保持不变，竖直方向匀加速，然后让游戏对象的position与其相加即可实现抛物线运动。

```javascript

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ParaCruve2 : MonoBehaviour

{

public float horizontalSpeed = 5.00f;

public float verticalSpeed = 0.00f;

const float g = 0.02f;

// Start is called before the first frame update

void Start()

{

Debug.Log("start!");

}

// Update is called once per frame

void Update()

{

Vector3 change = new Vector3(Time.deltaTime \* horizontalSpeed, -Time.deltaTime \* verticalSpeed, 0.00f);

this.transform.position += change;

verticalSpeed += g;

}

}

```

* 1. 与（2）类似，但不调用position，而是调用Translate函数

```javascript

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ParaCruve3 : MonoBehaviour

{

public float horizontalSpeed = 5.00f;

public float verticalSpeed = 0.00f;

const float g = 0.02f;

// Start is called before the first frame update

void Start()

{

Debug.Log("start!");

}

// Update is called once per frame

void Update()

{

Vector3 change = new Vector3(Time.deltaTime \* horizontalSpeed, -Time.deltaTime \* verticalSpeed, 0.0f);

this.transform.Translate(change);

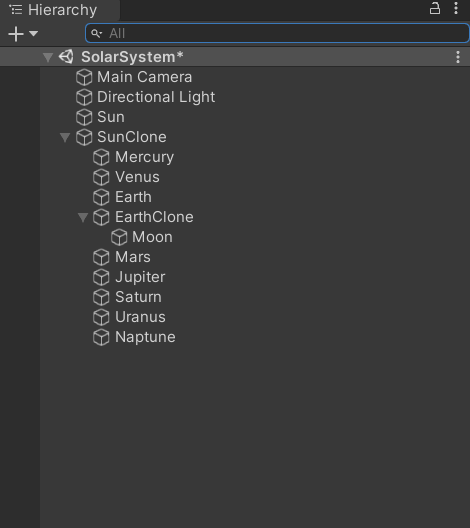
verticalSpeed += g;

}

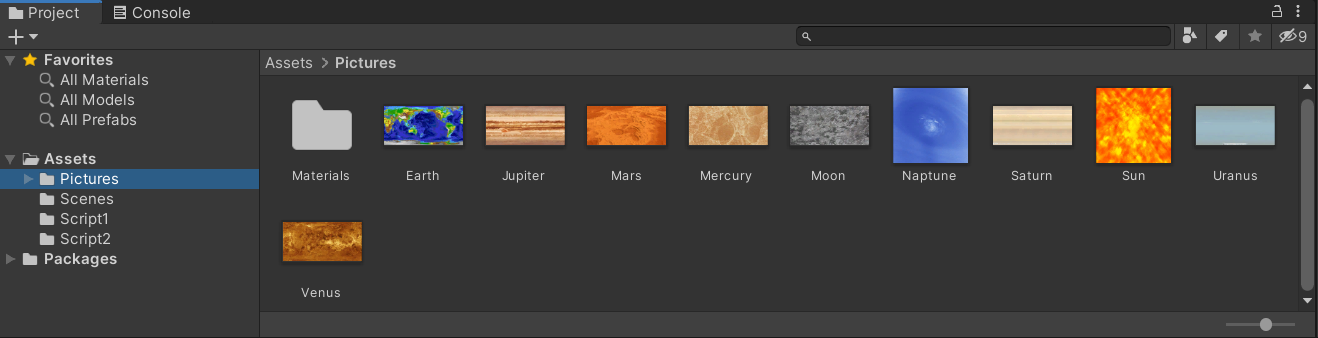
}

```

* 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，且不在一个法平面上。
  + 1. 首先添加“星球”并赋予图片材质



EarthClone（空对象）：考虑到地球有自转会影响到月球的公转，故不将Moon作为Earth的子级，同理，也不将八大星球作为太阳的子集，将它们统一放在SunClone中。



* + 1. 将下列代码放入SunClone中：（行星的公转与自转）

```javascript

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SolarSystem : MonoBehaviour

{

// Use this for initialization

void Start()

{

}

// Update is called once per frame

void Update()

{

//Rotation

float rotationStandarTime = 20.0f;

GameObject.Find("Mercury").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 58.65f));

GameObject.Find("Venus").transform.Rotate(Vector3.up \* Time.deltaTime \* (rotationStandarTime / 224.7f));

GameObject.Find("Earth").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 1.0f));

GameObject.Find("Moon").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 27.32f));

GameObject.Find("Mars").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 1.0f));

GameObject.Find("Jupiter").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 0.24f));

GameObject.Find("Saturn").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 0.24f));

GameObject.Find("Uranus").transform.Rotate(Vector3.up \* Time.deltaTime \* (rotationStandarTime / 0.71f));

GameObject.Find("Naptune").transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 0.67f));

//Orbital revolution

Transform center = this.transform;

float reStandarTime = 360.0f;

GameObject.Find("Mercury").transform.RotateAround(center.position, new Vector3(0.1f, -1, 0), reStandarTime / 88.0f);

GameObject.Find("Venus").transform.RotateAround(center.position, new Vector3(0.2f, -1, 0), reStandarTime / 224.7f);

GameObject.Find("Earth").transform.RotateAround(center.position, new Vector3(0.3f, -1, 0), reStandarTime / 365.25f);

GameObject.Find("EarthClone").transform.RotateAround(center.position, new Vector3(0.3f, -1, 0), reStandarTime / 365.25f);

GameObject.Find("Moon").transform.RotateAround(GameObject.Find("EarthClone").transform.position, new Vector3(0, -1, 0), reStandarTime / 27.32f);

GameObject.Find("Mars").transform.RotateAround(center.position, new Vector3(0.5f, -1, 0), reStandarTime / 686.98f);

GameObject.Find("Jupiter").transform.RotateAround(center.position, new Vector3(0.6f, -1, 0), reStandarTime / 4328.9f);

GameObject.Find("Saturn").transform.RotateAround(center.position, new Vector3(0.7f, -1, 0), reStandarTime / 10767.5f);

GameObject.Find("Uranus").transform.RotateAround(center.position, new Vector3(0.8f, -1, 0), reStandarTime / 30776.8f);

GameObject.Find("Naptune").transform.RotateAround(center.position, new Vector3(0.9f, -1, 0), reStandarTime / 60152.0f);

}

}

```

其中，要记得把SunClone的坐标修改为Sun的坐标，以免产生不可预期的效果。

* + 1. 将下列代码放入Sun中：(太阳自转)

```javascript

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SunRotation : MonoBehaviour

{

// Use this for initialization

void Start()

{

}

// Update is called once per frame

void Update()

{

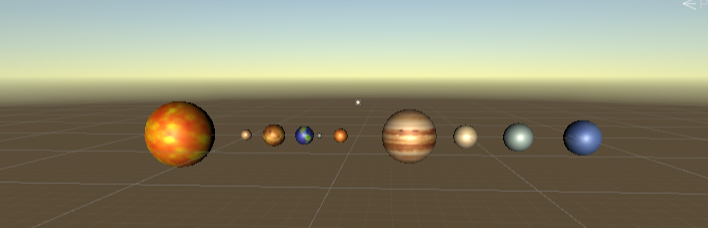
float rotationStandarTime = 20.0f;

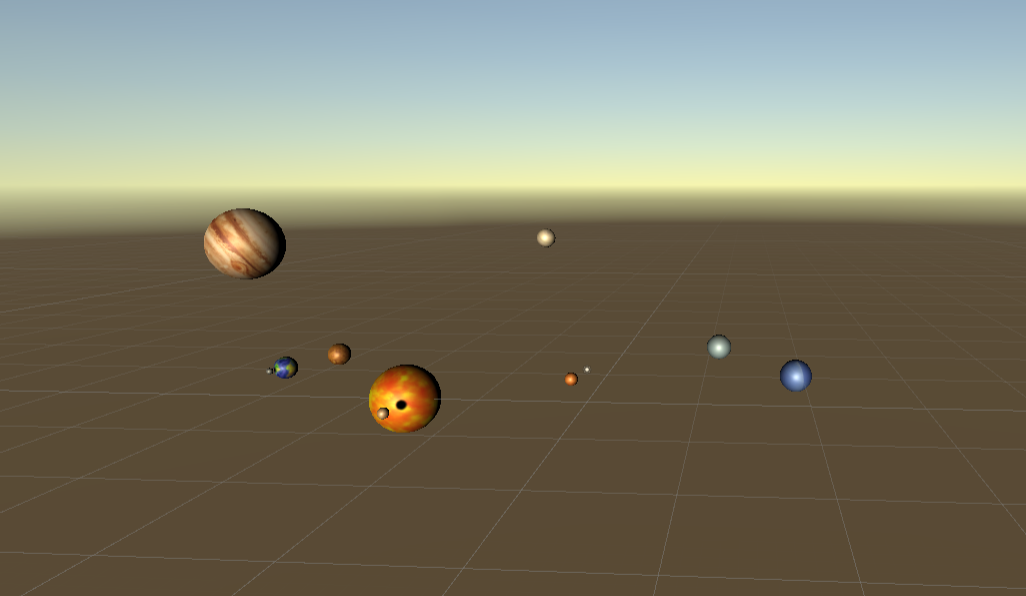
this.transform.Rotate(Vector3.down \* Time.deltaTime \* (rotationStandarTime / 25.05f));

}

}

```





### 2、编程实践

###### • 阅读以下游戏脚本

```

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!

```

\*\*程序需要满足的要求：

• play the game ( http://www.flash-game.net/game/2535/priests-and-devils.html )

• 列出游戏中提及的事物（Objects）

魔鬼（devil0，1，2），牧师（priest0，1，2），船（boat），左岸（from）‘右岸（to）。

• 用表格列出玩家动作表（规则表），注意，动作越少越好\*\*

|规则名称|条件|

|------|------|

|牧师或魔鬼上左岸|船已靠左岸并且船上有牧师或魔鬼|

|牧师或魔鬼上右岸|船已靠右岸并且船上有牧师或魔鬼|

|魔鬼上船|岸上有魔鬼，船未载满两人|

|牧师上船|岸上有牧师，船未载满两人|

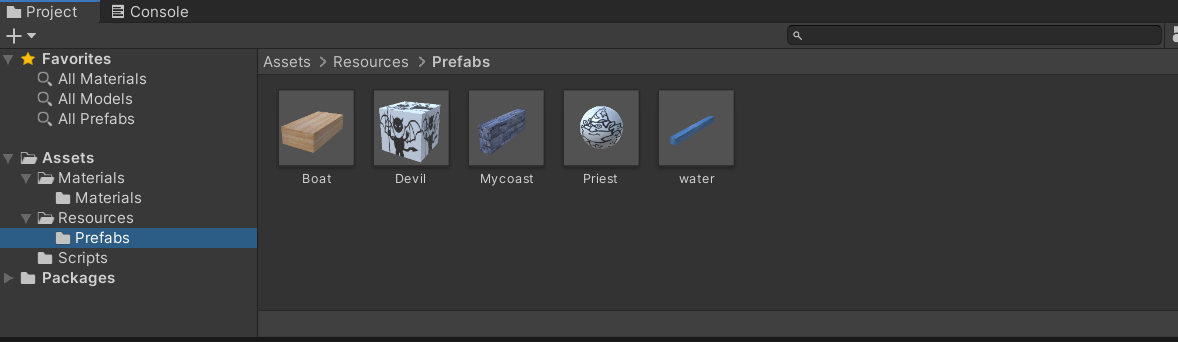
|船只开动|船上有人|

|击杀规则|一边的牧师数量少于魔鬼数量|

|游戏胜利规则|所有角色从左岸到达右岸，且全部存活|

|游戏失败规则|牧师被击杀|

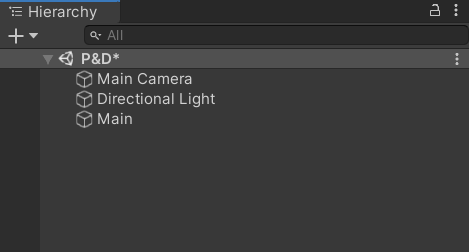
(\*\*• 请将游戏中对象做成预制\*\*



\*\*• 在场景控制器 LoadResources 方法中加载并初始化 长方形、正方形、球 及其色彩代表游戏中的对象。\*\*

\*\*• 使用 C# 集合类型 有效组织对象\*\*

\*\*• 整个游戏仅 主摄像机 和 一个 Empty 对象， 其他对象必须代码动态生成！！！ 。 整个游戏不许出现 Find 游戏对象，SendMessage 这类突破程序结构的 通讯耦合 语句。 违背本条准则，不给分\*\*



\*\*• 请使用课件架构图编程，不接受非 MVC 结构程序\*\*

\*\*• 注意细节，例如：船未靠岸，牧师与魔鬼上下船运动中，均不能接受用户事件！\*\*)

#### 游戏架构

\*使用了MVC架构。\*

1. 场景中的所有GameObject就是Model，它们受到Controller的控制，比如说牧师和魔鬼受到MyCharacterController类的控制，船受到BoatController类的控制，河岸受到CoastController类的控制。

2. View就是UserGUI和ClickGUI，它们展示游戏结果，并提供用户交互的渠道（点击物体和按钮）。

3. Controller：除了刚才说的MyCharacterController、BoatController、CoastController以外，还有更高一层的Controller：FirstController（场景控制器），FirstController控制着这个场景中的所有对象，包括其加载、通信、用户输入。

最高层的Controller是Director类，一个游戏中只能有一个实例，它控制着场景的创建、切换、销毁、游戏暂停、游戏退出等等最高层次的功能。

以下是本游戏的UML图：

![在这里插入图片描述](https://img-blog.csdnimg.cn/20200928092408555.jpg?x-oss-process=image/watermark,type\_ZmFuZ3poZW5naGVpdGk,shadow\_10,text\_aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3FxXzQzNzE5NDM3,size\_16,color\_FFFFFF,t\_70#pic\_center)

#### Director

```javascript

public class Director: System.Object

{

private static Director \_instance;

public SceneController curren{ get; set;}

public static Director get\_Instance(){

if (\_instance == null)

{

\_instance = new Director();

}

return \_instance;

}

}

```

Director是最高层的控制器，运行游戏时始终只有一个实例，它掌控着场景的加载、切换等，也可以控制游戏暂停、结束等等。但它\*\*并不控制场景中的具体对象\*\*，控制场景对象的任务交给了SceneController（场景控制器）。

Director类使用了单例模式。第一次调用Director.getInstance()时，会创建一个新的Director对象，保存在\_instance，此后每次调用getInstance，都回返回\_instance。也就是说Director最多只有一个实例。这样，我们在任何Script中的任何地方通过Director.getInstance()都能得到同一个Director对象，也就可以获得同一个currentSceneController，这样我们就可以轻易实现类与类之间的通信，比如说我在其他控制器中就可以使用Director.getInstance().somethingHappen()来告诉导演某一件事情发生了，导演就可以在somethingHappen()方法中做出对应的反应。

#### SceneController 接口

```javascript

public interface SceneController

{

void loadResources();

}

```

interface（接口）不能直接用来创建对象！必须先有一个类实现（继承）它，在我的这个游戏中就是FirstController类。

SceneController 是导演控制场景控制器的渠道。在上面的Director 类中，currentSceneController （FirstController类）就是SceneController的实现，所以Director可以调用SceneController接口中的方法，来实现对场景的控制。

#### Moveable

Moveable是一个可挂载在GameObject上的类。在Moveable类中只有三个函数，Update函数用于控制船的移动，Controller可以通过SetDestination()方法让GameObject移动起来。

```javascript

public class moveable: MonoBehaviour

{

readonly float move\_speed = 20;

private int move\_to\_where;//0->not move, 1->to middle, 2->to destination

private Vector3 dest;

private Vector3 middle;

public static int cn\_move = 0;//0->can move, 1->cant move

void Update(){

if (cn\_move == 1)

return;

else{

if(move\_to\_where == 1){

transform.position = Vector3.MoveTowards(transform.position, middle, move\_speed\*Time.deltaTime);

if (transform.position == middle)

move\_to\_where = 2;

}

else if(move\_to\_where == 2){

transform.position = Vector3.MoveTowards(transform.position, dest, move\_speed\*Time.deltaTime);

if (transform.position == dest)

move\_to\_where = 0;

}

}

}

public void SetDestination(Vector3 \_dest){

if (cn\_move == 1)

return;

else{

middle = \_dest;

dest = \_dest;

if (\_dest.y < transform.position.y) {

middle.y = transform.position.y;

} else {

middle.x = transform.position.x;

}

move\_to\_where = 1;

}

}

public void reset(){

if (cn\_move == 1)

return;

else{

move\_to\_where = 0;

}

}

}

```

此处我们不让物体直接移动到目的地dest，因为那样可能会直接穿过河岸物体。因此我们用middle来保存一个中间位置，让物体先移动到middle，再移动到dest，这就实现了一个折线的移动，不会穿越河岸。moving\_status记录着目前该物体处于哪种移动状态。

#### MyCharacterController

MyCharacterController封装了一个GameObject，表示游戏角色（牧师或恶魔）。我们要关注6个角色与船和河岸的联系，尤其是在船移动时我们必须保证角色和船是绑定关系（一起移动）。

```javascript

public class MyCharacterController{

readonly GameObject character;

readonly moveable Cmove;

readonly ClickGUI clickgui;

readonly int Ctype;//0->priset, 1->devil

private bool isOnboat;

private CoastController coastcontroller;

public MyCharacterController(string Myname){

if(Myname == "priest"){

character = Object.Instantiate(Resources.Load("Prefabs/Priest", typeof(GameObject)), Vector3.zero, Quaternion.identity,null) as GameObject;

Ctype = 0;

}

else{

character = Object.Instantiate(Resources.Load("Prefabs/Devil", typeof(GameObject)), Vector3.zero, Quaternion.identity,null) as GameObject;

Ctype = 1;

}

Cmove = character.AddComponent(typeof(moveable)) as moveable;

clickgui = character.AddComponent(typeof(ClickGUI)) as ClickGUI;

clickgui.setController(this);

}

public int getType(){

return Ctype;

}

public void setName(string name){

character.name = name;

}

public string getName(){

return character.name;

}

public void setPosition(Vector3 postion){

character.transform.position = postion;

}

public void moveToPosition(Vector3 \_dest){

Cmove.SetDestination (\_dest);

}

public void getOnBoat(BoatController tem\_boat){

coastcontroller = null;

character.transform.parent = tem\_boat.getGameObject ().transform;

isOnboat = true;

}

public void getOnCoast(CoastController coastCon){

coastcontroller = coastCon;

character.transform.parent = null;

isOnboat = false;

}

public bool \_isOnBoat(){

return isOnboat;

}

public CoastController getCoastController(){

return coastcontroller;

}

public void reset(){

Cmove.reset ();

coastcontroller = (Director.get\_Instance ().curren as MySceneController).fromCoast;

getOnCoast(coastcontroller);

setPosition (coastcontroller.getEmptyPosition ());

coastcontroller.getOnCoast (this);

}

public void Mypause(){

moveable.cn\_move = 1;

}

public void MyConti(){

moveable.cn\_move = 0;

}

}

```

在构造函数中实例化了一个perfab，创建GameObject，因此我们每 new MyCharacterController() 一次，场景中就会多一个游戏角色。

构造函数还将clickGUI挂载到了这个角色上，以监测“鼠标点击角色”的事件。

#### BoatController和CoastController

BoatController和CoastController类似于MyCharacterController，封装了船GameObject和河岸GameObject。我们将它们是一种“容器”，游戏角色要进入它们的空位中。因此它们要提供getEmptyPosition()方法，给出自己的空位，让游戏角色能够移动到合适的位置。

1. BoatController

```javascript

public class BoatController{

readonly GameObject boat;

readonly moveable Cmove;

readonly Vector3 fromPos = new Vector3 (5, 1, 0);

readonly Vector3 toPos = new Vector3 (-5, 1, 0);

readonly Vector3[] from\_pos;

readonly Vector3[] to\_pos;

private int TFflag;//-1->to, 1->from

private MyCharacterController[] passenger = new MyCharacterController[2];

public BoatController(){

TFflag = 1;

from\_pos = new Vector3[]{ new Vector3 (4.5f, 1.5f, 0), new Vector3 (5.5f, 1.5f, 0) };

to\_pos = new Vector3[]{ new Vector3 (-5.5f, 1.5f, 0), new Vector3 (-4.5f, 1.5f, 0) };

boat = Object.Instantiate (Resources.Load ("Prefabs/Boat", typeof(GameObject)), fromPos, Quaternion.identity, null) as GameObject;

boat.name = "boat";

Cmove = boat.AddComponent (typeof(moveable)) as moveable;

boat.AddComponent (typeof(ClickGUI));

}

public void boatMove(){

if (TFflag == 1) {

Cmove.SetDestination (toPos);

TFflag = -1;

} else {

Cmove.SetDestination (fromPos);

TFflag = 1;

}

}

public void getOnBoat(MyCharacterController tem\_cha){

int index = getEmptyIndex ();

passenger [index] = tem\_cha;

}

public MyCharacterController getOffBoat(string object\_name){

for (int i = 0; i < passenger.Length; i++) {

if (passenger [i] != null && passenger [i].getName () == object\_name) {

MyCharacterController temp\_character = passenger [i];

passenger [i] = null;

return temp\_character;

}

}

return null;

}

public int getEmptyIndex(){

for (int i = 0; i < passenger.Length; i++) {

if (passenger [i] == null)

return i;

}

return -1;

}

public bool IfEmpty(){

for (int i = 0; i < passenger.Length; i++) {

if (passenger [i] != null)

return false;

}

return true;

}

public Vector3 getEmptyPosition(){

Vector3 pos;

int index = getEmptyIndex ();

if (TFflag == 1) {

pos = from\_pos [index];

} else {

pos = to\_pos [index];

}

return pos;

}

public GameObject getGameObject(){

return boat;

}

public int getTFflag(){

return TFflag;

}

public int[] getCharacterNum(){

int[] count = { 0, 0 };

for (int i = 0; i < passenger.Length; i++) {

if (passenger [i] == null)

continue;

if (passenger [i].getType () == 0) {

count [0]++;

} else {

count [1]++;

}

}

return count;

}

public void reset(){

Cmove.reset ();

if (TFflag == -1) {

boatMove ();

}

passenger = new MyCharacterController[2];

}

public void Mypause(){

moveable.cn\_move = 1;

}

public void MyConti(){

moveable.cn\_move = 0;

}

}

```

2. CoastController

```javascript

public class CoastController{

readonly GameObject coast;

readonly Vector3 from\_pos = new Vector3(9,1,0);

readonly Vector3 to\_pos = new Vector3(-9,1,0);

readonly Vector3[] postion;

readonly int TFflag;//-1->to, 1->from

private MyCharacterController[] passengerPlaner;

public CoastController(string to\_or\_from){

postion = new Vector3[] {

new Vector3 (6.5f, 2.25f, 0),

new Vector3 (7.5f, 2.25f, 0),

new Vector3 (8.5f, 2.25f, 0),

new Vector3 (9.5f, 2.25f, 0),

new Vector3 (10.5f, 2.25f, 0),

new Vector3 (11.5f, 2.25f, 0)

};

passengerPlaner = new MyCharacterController[6];

if(to\_or\_from == "from"){

coast = Object.Instantiate(Resources.Load("Prefabs/Mycoast", typeof(GameObject)), from\_pos, Quaternion.identity, null) as GameObject;

coast.name = "from";

TFflag = 1;

}

else{

coast = Object.Instantiate(Resources.Load("Prefabs/Mycoast", typeof(GameObject)), to\_pos, Quaternion.identity, null) as GameObject;

coast.name = "to";

TFflag = -1;

}

}

public int getTFflag(){

return TFflag;

}

public MyCharacterController getOffCoast(string object\_name){

for(int i=0; i<passengerPlaner.Length; i++){

if(passengerPlaner[i] != null && passengerPlaner[i].getName() == object\_name){

MyCharacterController myCharacter = passengerPlaner[i];

passengerPlaner[i] = null;

return myCharacter;

}

}

return null;

}

public int getEmptyIndex(){

for(int i=0; i<passengerPlaner.Length; i++){

if(passengerPlaner[i] == null){

return i;

}

}

return -1;

}

public Vector3 getEmptyPosition(){

int index = getEmptyIndex();

Vector3 pos = postion[index];

pos.x \*= TFflag;

return pos;

}

public void getOnCoast(MyCharacterController myCharacter){

int index = getEmptyIndex();

passengerPlaner[index] = myCharacter;

}

public int[] getCharacterNum(){

int[] count = {0,0};

for(int i=0; i<passengerPlaner.Length; i++){

if(passengerPlaner[i] == null) continue;

if(passengerPlaner[i].getType() == 0) count[0]++;

else count[1]++;

}

return count;

}

public void reset(){

passengerPlaner = new MyCharacterController[6];

}

}

```

MyCharacterController、BoatController、CoastController有一些方法名是重复的，看起来似乎功能有点重复，但我们不只用一个函数操控游戏角色的上船，原因是\*\*不要在一个类中操作另一个类，那会加强两个类之间的耦合性。\*\*

例如：MyCharacterController中的getOnBoat()只应该操作MyCharacterController中的成员，BoatController中的GetOnBoat()只应该操作BoatController中的成员。

#### View

用户交互界面，其中包括游戏场景，游戏对象的生成，游戏的暂停、继续、重开功能。包括以下两个类：

• UserGUI类：设置游戏的暂停、继续与重开的游戏的按钮。

```javascript

public class UserGUI : MonoBehaviour {

private UserAction action;

private GUIStyle MyStyle;

private GUIStyle MyButtonStyle;

public int if\_win\_or\_not;

void Start(){

action = Director.get\_Instance ().curren as UserAction;

MyStyle = new GUIStyle ();

MyStyle.fontSize = 40;

MyStyle.normal.textColor = new Color (255f, 0, 0);

MyStyle.alignment = TextAnchor.MiddleCenter;

MyButtonStyle = new GUIStyle ("button");

MyButtonStyle.fontSize = 30;

}

void reStart(){

if (GUI.Button (new Rect (Screen.width/2-Screen.width/8, Screen.height/2+100, 150, 50), "Restart", MyButtonStyle)) {

if\_win\_or\_not = 0;

action.restart ();

moveable.cn\_move = 0;

}

}

void IsPause(){

if (GUI.Button (new Rect (Screen.width / 2 - 350, Screen.height / 2 + 100, 150, 50), "Pause", MyButtonStyle)) {

if (moveable.cn\_move == 0) {

action.pause ();

moveable.cn\_move = 1;

}

} else if (GUI.Button (new Rect (Screen.width-Screen.width/2, Screen.height / 2 + 100, 150, 50), "Continue", MyButtonStyle)) {

if (moveable.cn\_move == 1) {

action.Coninu();

moveable.cn\_move = 0;

}

}

}

void OnGUI(){

IsPause ();

reStart ();

if(moveable.cn\_move == 1)

GUI.Label (new Rect (Screen.width/2-Screen.width/8, 50, 100, 50), "Pausing", MyStyle);

if (if\_win\_or\_not == -1) {

GUI.Label (new Rect (Screen.width/2-Screen.width/8, 50, 100, 50), "Game Over!!!", MyStyle);

IsPause ();

reStart ();

} else if (if\_win\_or\_not == 1) {

GUI.Label (new Rect (Screen.width/2-Screen.width/8, 50, 100, 50), "You Win!!!", MyStyle);

IsPause ();

reStart ();

}

}

}

```

• ClickGUI类：设置游戏对象的点击事件。

```javascript

public class ClickGUI : MonoBehaviour {

// Use this for initialization

UserAction action;

MyCharacterController character;

public void setController(MyCharacterController tem){

character = tem;

}

void Start(){

action = Director.get\_Instance ().curren as UserAction;

}

void OnMouseDown(){

if (gameObject.name == "boat") {

action.moveboat ();

} else {

action.isClickChar (character);

}

}

}

```

#### Controller

对于整个游戏，我们需要一个场记用于指挥所有角色的行为，与用户的交互等，这个类就是MySceneController。

```javascript

public class MySceneController : MonoBehaviour, SceneController, UserAction{

readonly Vector3 water\_pos = new Vector3 (0, 0.5f, 0);

UserGUI user;

public CoastController fromCoast;

public CoastController toCoast;

public BoatController boat;

//private MyCharacterController[] characters;

private List<MyCharacterController> team;

void Awake(){

Director director = Director.get\_Instance ();

director.curren = this;

user = gameObject.AddComponent<UserGUI> () as UserGUI;

//characters = new MyCharacterController[6];

team = new List<MyCharacterController>();

loadResources ();

}

public void loadResources(){

GameObject water = Instantiate (Resources.Load ("Prefabs/water", typeof(GameObject)), water\_pos, Quaternion.identity, null) as GameObject;

water.name = "water";

fromCoast = new CoastController ("from");

toCoast = new CoastController ("to");

boat = new BoatController ();

for (int i = 0; i < 3; i++) {

MyCharacterController tem = new MyCharacterController ("priest");

tem.setName ("priest" + i);

tem.setPosition (fromCoast.getEmptyPosition ());

tem.getOnCoast (fromCoast);

fromCoast.getOnCoast (tem);

team.Add (tem);

}

for (int i = 0; i < 3; i++) {

MyCharacterController tem = new MyCharacterController ("devil");

tem.setName ("devil" + i);

tem.setPosition (fromCoast.getEmptyPosition ());

tem.getOnCoast (fromCoast);

fromCoast.getOnCoast (tem);

team.Add (tem);

}

}

public void moveboat(){

if (boat.IfEmpty ())

return;

boat.boatMove ();

//check whether game over

user.if\_win\_or\_not = checkGameOver();

}

public void isClickChar (MyCharacterController tem\_char){

if (moveable.cn\_move == 1)

return;

if (tem\_char.\_isOnBoat ()) {

CoastController tem\_coast;

if (boat.getTFflag () == -1) {

tem\_coast = toCoast;

} else {

tem\_coast = fromCoast;

}

boat.getOffBoat (tem\_char.getName ());

tem\_char.moveToPosition (tem\_coast.getEmptyPosition ());

tem\_char.getOnCoast (tem\_coast);

tem\_coast.getOnCoast (tem\_char);

} else {

CoastController tem\_coast2 = tem\_char.getCoastController ();

if (boat.getEmptyIndex () == -1)

return;

if (boat.getTFflag () != tem\_coast2.getTFflag ())

return;

tem\_coast2.getOffCoast (tem\_char.getName());

tem\_char.moveToPosition (boat.getEmptyPosition ());

tem\_char.getOnBoat (boat);

boat.getOnBoat (tem\_char);

}

//check whether game over;

user.if\_win\_or\_not = checkGameOver();

}

public void restart(){

boat.reset ();

fromCoast.reset ();

toCoast.reset ();

foreach (MyCharacterController i in team) {

i.reset ();

}

moveable.cn\_move = 0;

}

public void pause(){

boat.Mypause ();

foreach (MyCharacterController i in team) {

i.Mypause();

}

}

public void Coninu (){

boat.MyConti ();

foreach (MyCharacterController i in team) {

i.MyConti();

}

}

private int checkGameOver(){

if (moveable.cn\_move == 1)

return 0;

int from\_priest = 0;

int from\_devil = 0;

int to\_priest = 0;

int to\_devil = 0;

int[] from\_count = fromCoast.getCharacterNum ();

from\_priest = from\_count [0];

from\_devil = from\_count [1];

int[] to\_count = toCoast.getCharacterNum ();

to\_priest = to\_count [0];

to\_devil = to\_count [1];

if (to\_devil + to\_priest == 6)

return 1;//you win

int[] boat\_count = boat.getCharacterNum();

if (boat.getTFflag () == 1) {

from\_priest += boat\_count [0];

from\_devil += boat\_count [1];

} else {

to\_priest += boat\_count [0];

to\_devil += boat\_count [1];

}

if (from\_priest < from\_devil && from\_priest > 0)

return -1;//you lose

if(to\_priest < to\_devil && to\_priest > 0)

return -1;//you lose

return 0;//not yet finish

}

}

```

有了场记我们还缺少导演的。于是我们用Director类实现单例模式，在游戏开始时仅创建一个实例，这个实例控制着游戏的运行，暂停，互动。

```javascript

public class Director: System.Object

{

private static Director \_instance;

public SceneController curren{ get; set;}

public static Director get\_Instance(){

if (\_instance == null)

{

\_instance = new Director();

}

return \_instance;

}

}

```

以下是成品图：



最后感谢[师兄的博客](https://blog.csdn.net/hellowangld/article/details/79805000)帮助。3、思考题【选做】

* 使用向量与变换，实现并扩展 Tranform 提供的方法，如 Rotate、RotateAround 等