**手机安卓端app代码清单**

****

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | SE2018春-G08-迷城逃亡app代码清单 |
| 当前版本： | 0.1.1 |
| 作者： | 吴子乔 |
| 完成日期： | 2018-06-04 |

# 版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 参与者 | 起止日期 | 备注 | 审核人 |
| 0.1.1 | 吴子乔 | 2018-06-01至  2018-06-04 | 代码清单进行编写 | 陈栩,石梦韬 |
|  |  |  |  |  |

目录

[版本历史 2](#_Toc515908535)

[代码清单 2](#_Toc515908536)

[一、模块1（登录模块） 2](#_Toc515908537)

[1.1 UI控制 2](#_Toc515908538)

[1.2注册操作代码 5](#_Toc515908539)

[1.3登录操作代码 6](#_Toc515908540)

[二、模块2（游戏内容控制模块） 8](#_Toc515908541)

[2.1游戏场景动画 8](#_Toc515908542)

[2.2游戏操作代码 25](#_Toc515908543)

[2.3游戏效果控制代码 83](#_Toc515908544)

[三、模块3（排行榜操作模块） 86](#_Toc515908545)

[3.1排行榜下载显示代码 86](#_Toc515908546)

[3.2排行榜更新查看代码 88](#_Toc515908547)

# 代码清单

## 一、模块1（登录模块）

### UI控制

1. 整体画板控制代码

public GameObject p1;

public GameObject p2;

public GameObject p3;

public GameObject p4;

public GameObject p5;

public GameObject p6;

public GameObject p7;

void Start() {

p1.SetActive(true);

p2.SetActive(false);

p3.SetActive(false);

p4.SetActive(false);

p5.SetActive(false);

p6.SetActive(false);

p7.SetActive(false);

SetFirstStatic();

}

public void onchick1\_1()

{

SetStatic(p1, p2);

}

public void onchick1()

{

SetStatic(p2, p1);

GameObject btnObjx = GameObject.Find("returnmain");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(onchick1\_1);

}

public void onchick2\_1()

{

SetStatic(p1, p4);

}

public void onchick2()

{

SetStatic(p4, p2);

GameObject btnObjx = GameObject.Find("returnmain1");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(onchick2\_1);

//补全代码

}

public void onchick3()

{

Application.Quit();

//补全代码

}

public void SetFirstStatic()

{

GameObject btnObj = GameObject.Find("zhucewai");

Button btn = btnObj.GetComponent<Button>();

btn.onClick.AddListener(onchick1);

btnObj = GameObject.Find("denglu");

btn = btnObj.GetComponent<Button>();

btn.onClick.AddListener(onchick2);

btnObj = GameObject.Find("likai");

btn = btnObj.GetComponent<Button>();

btn.onClick.AddListener(onchick3);

}

public void SetStatic(GameObject pn1, GameObject pn2)

{

pn1.SetActive(true);

pn2.SetActive(false);

}

1. 跳往注册页面代码

void Start()

{

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClickssd);

}

private void OnClickssd()

{

SceneManager.LoadScene("screen\_2");

}

1. 跳往登录页面代码

public GameObject p4;

public GameObject p7;

void Start()

{

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClickssd);

}

private void OnClickssd()

{

p7.SetActive(false);

p4.SetActive(true);

}

（4）跳回主菜单页面代码

public GameObject p1;

public GameObject p7;

void Start()

{

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClickssd);

}

private void OnClickssd()

{

p7.SetActive(false);

p1.SetActive(true);

}

（5）离开游戏代码

void Start()

{

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClickssd);

}

private void OnClickssd()

{

Application.Quit();

}

### 1.2注册操作代码

public GameObject penal1;

public GameObject penal2;

public GameObject penal3;

public InputField Inp\_Username = null;

private string s;

// Use this for initialization

void Start()

{

penal1.SetActive(true);

penal2.SetActive(false);

GameObject btnObjx = GameObject.Find("zhucenei");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(onchick);

}

void Onone()

{

penal1.SetActive(true);

penal2.SetActive(false);

}

public void onchick()

{

this.StartCoroutine(this.Login());

}

private IEnumerator Login()

{

//数据库判断信息

WWWForm form = new WWWForm();

form.AddField("name", Inp\_Username.text);

WWW www = new WWW("http://10.64.129.142/createname.php", form);

yield return www;

if (www.error != null)

{

Debug.LogError(www.error);

}

else

Debug.Log(www.text);

//账号不存在

if (www.text == "1")

{

penal3.SetActive(true);

penal1.SetActive(false);

}

else //如果账号已经存在

{

penal1.SetActive(false);

penal2.SetActive(true);

GameObject btnObjx = GameObject.Find("queding1");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(Onone);

}

### 1.3登录操作代码

public GameObject panel;

public GameObject penal2;

public GameObject penal3;

public InputField Inp\_Username=null;

private string s;

// Use this for initialization

void Start()

{

panel.SetActive(true);

penal2.SetActive(false);

GameObject btnObj = GameObject.Find("denglu1");

Button btn = btnObj.GetComponent<Button>();

btn.onClick.AddListener(onchick);

}

void Onone()

{

panel.SetActive(true);

penal2.SetActive(false);

}

public void onchick()

{

this.StartCoroutine(this.Login());

}

private IEnumerator Login()

{

//数据库判断信息

WWWForm form = new WWWForm();

form.AddField("name", Inp\_Username.text);

WWW www = new WWW("http://10.64.129.142/createname.php", form);

yield return www;

if (www.error != null)

{

Debug.LogError(www.error);

}

else

Debug.Log(www.text);

//账号不存在

if (www.text != "1")

{

panel.SetActive(false);

penal3.SetActive(true);

}

else //如果账号已经存在

{

panel.SetActive(false);

penal2.SetActive(true);

GameObject btnObjx = GameObject.Find("queding2");

Button btnx = btnObjx.GetComponent<Button>();

btnx.onClick.AddListener(Onone);

}

## 二、模块2（游戏内容控制模块）

### 2.1游戏场景动画

（1）人类场景动画

/// <summary>

/// A Character that can be used in dialogue via the Say, Conversation and Portrait commands.

/// </summary>

[ExecuteInEditMode]

public class Character : MonoBehaviour, ILocalizable

{

[Tooltip("Character name as displayed in Say Dialog.")]

[SerializeField] protected string nameText; // We need a separate name as the object name is used for character variations (e.g. "Smurf Happy", "Smurf Sad")

[Tooltip("Color to display the character name in Say Dialog.")]

[SerializeField] protected Color nameColor = Color.white;

[Tooltip("Sound effect to play when this character is speaking.")]

[SerializeField] protected AudioClip soundEffect;

[Tooltip("List of portrait images that can be displayed for this character.")]

[SerializeField] protected List<Sprite> portraits;

[Tooltip("Direction that portrait sprites face.")]

[SerializeField] protected FacingDirection portraitsFace;

[Tooltip("Sets the active Say dialog with a reference to a Say Dialog object in the scene. This Say Dialog will be used whenever the character speaks.")]

[SerializeField] protected SayDialog setSayDialog;

[FormerlySerializedAs("notes")]

[TextArea(5,10)]

[SerializeField] protected string description;

protected PortraitState portaitState = new PortraitState();

protected static List<Character> activeCharacters = new List<Character>();

protected virtual void OnEnable()

{

if (!activeCharacters.Contains(this))

{

activeCharacters.Add(this);

}

}

protected virtual void OnDisable()

{

activeCharacters.Remove(this);

}

#region Public members

/// <summary>

/// Gets the list of active characters.

/// </summary>

public static List<Character> ActiveCharacters { get { return activeCharacters; } }

/// <summary>

/// Character name as displayed in Say Dialog.

/// </summary>

public virtual string NameText { get { return nameText; } }

/// <summary>

/// Color to display the character name in Say Dialog.

/// </summary>

public virtual Color NameColor { get { return nameColor; } }

/// <summary>

/// Sound effect to play when this character is speaking.

/// </summary>

/// <value>The sound effect.</value>

public virtual AudioClip SoundEffect { get { return soundEffect; } }

/// <summary>

/// List of portrait images that can be displayed for this character.

/// </summary>

public virtual List<Sprite> Portraits { get { return portraits; } }

/// <summary>

/// Direction that portrait sprites face.

/// </summary>

public virtual FacingDirection PortraitsFace { get { return portraitsFace; } }

/// <summary>

/// Currently display profile sprite for this character.

/// </summary>

/// <value>The profile sprite.</value>

public virtual Sprite ProfileSprite { get; set; }

/// <summary>

/// Current display state of this character's portrait.

/// </summary>

/// <value>The state.</value>

public virtual PortraitState State { get { return portaitState; } }

/// <summary>

/// Sets the active Say dialog with a reference to a Say Dialog object in the scene. This Say Dialog will be used whenever the character speaks.

/// </summary>

public virtual SayDialog SetSayDialog { get { return setSayDialog; } }

/// <summary>

/// Returns the name of the game object.

/// </summary>

public string GetObjectName() { return gameObject.name; }

/// <summary>

/// Returns true if the character name starts with the specified string. Case insensitive.

/// </summary>

public virtual bool NameStartsWith(string matchString)

{

#if NETFX\_CORE

return name.StartsWith(matchString, StringComparison.CurrentCultureIgnoreCase)

|| nameText.StartsWith(matchString, StringComparison.CurrentCultureIgnoreCase);

#else

return name.StartsWith(matchString, true, System.Globalization.CultureInfo.CurrentCulture)

|| nameText.StartsWith(matchString, true, System.Globalization.CultureInfo.CurrentCulture);

#endif

}

/// <summary>

/// Looks for a portrait by name on a character

/// If none is found, give a warning and return a blank sprite

/// </summary>

public virtual Sprite GetPortrait(string portraitString)

{

if (String.IsNullOrEmpty(portraitString))

{

return null;

}

for (int i = 0; i < portraits.Count; i++)

{

if (portraits[i] != null && String.Compare(portraits[i].name, portraitString, true) == 0)

{

return portraits[i];

}

}

return null;

}

#endregion

#region ILocalizable implementation

public virtual string GetStandardText()

{

return nameText;

}

public virtual void SetStandardText(string standardText)

{

nameText = standardText;

}

public virtual string GetDescription()

{

return description;

}

public virtual string GetStringId()

{

// String id for character names is CHARACTER.<Character Name>

return "CHARACTER." + nameText;

}

#endregion

[ExecuteInEditMode]

public class Flowchart : MonoBehaviour, ISubstitutionHandler

{

[HideInInspector]

[SerializeField] protected int version = 0; // Default to 0 to always trigger an update for older versions of Fungus.

[HideInInspector]

[SerializeField] protected Vector2 scrollPos;

[HideInInspector]

[SerializeField] protected Vector2 variablesScrollPos;

[HideInInspector]

[SerializeField] protected bool variablesExpanded = true;

[HideInInspector]

[SerializeField] protected float blockViewHeight = 400;

[HideInInspector]

[SerializeField] protected float zoom = 1f;

[HideInInspector]

[SerializeField] protected Rect scrollViewRect;

[HideInInspector]

[SerializeField] protected List<Block> selectedBlocks = new List<Block>();

[HideInInspector]

[SerializeField] protected List<Command> selectedCommands = new List<Command>();

[HideInInspector]

[SerializeField] protected List<Variable> variables = new List<Variable>();

[TextArea(3, 5)]

[Tooltip("Description text displayed in the Flowchart editor window")]

[SerializeField] protected string description = "";

[Range(0f, 5f)]

[Tooltip("Adds a pause after each execution step to make it easier to visualise program flow. Editor only, has no effect in platform builds.")]

[SerializeField] protected float stepPause = 0f;

[Tooltip("Use command color when displaying the command list in the Fungus Editor window")]

[SerializeField] protected bool colorCommands = true;

[Tooltip("Hides the Flowchart block and command components in the inspector. Deselect to inspect the block and command components that make up the Flowchart.")]

[SerializeField] protected bool hideComponents = true;

[Tooltip("Saves the selected block and commands when saving the scene. Helps avoid version control conflicts if you've only changed the active selection.")]

[SerializeField] protected bool saveSelection = true;

[Tooltip("Unique identifier for this flowchart in localized string keys. If no id is specified then the name of the Flowchart object will be used.")]

[SerializeField] protected string localizationId = "";

[Tooltip("Display line numbers in the command list in the Block inspector.")]

[SerializeField] protected bool showLineNumbers = false;

[Tooltip("List of commands to hide in the Add Command menu. Use this to restrict the set of commands available when editing a Flowchart.")]

[SerializeField] protected List<string> hideCommands = new List<string>();

[Tooltip("Lua Environment to be used by default for all Execute Lua commands in this Flowchart")]

[SerializeField] protected LuaEnvironment luaEnvironment;

[Tooltip("The ExecuteLua command adds a global Lua variable with this name bound to the flowchart prior to executing.")]

[SerializeField] protected string luaBindingName = "flowchart";

protected static List<Flowchart> cachedFlowcharts = new List<Flowchart>();

protected static bool eventSystemPresent;

protected StringSubstituter stringSubstituer;

#if UNITY\_5\_4\_OR\_NEWER

#else

protected virtual void OnLevelWasLoaded(int level)

{

LevelWasLoaded();

}

#endif

protected virtual void LevelWasLoaded()

{

// Reset the flag for checking for an event system as there may not be one in the newly loaded scene.

eventSystemPresent = false;

}

protected virtual void Start()

{

CheckEventSystem();

}

// There must be an Event System in the scene for Say and Menu input to work.

// This method will automatically instantiate one if none exists.

protected virtual void CheckEventSystem()

{

if (eventSystemPresent)

{

return;

}

EventSystem eventSystem = GameObject.FindObjectOfType<EventSystem>();

if (eventSystem == null)

{

// Auto spawn an Event System from the prefab

GameObject prefab = Resources.Load<GameObject>("Prefabs/EventSystem");

if (prefab != null)

{

GameObject go = Instantiate(prefab) as GameObject;

go.name = "EventSystem";

}

}

eventSystemPresent = true;

}

private void SceneManager\_activeSceneChanged(UnityEngine.SceneManagement.Scene arg0, UnityEngine.SceneManagement.Scene arg1)

{

LevelWasLoaded();

}

protected virtual void OnEnable()

{

if (!cachedFlowcharts.Contains(this))

{

cachedFlowcharts.Add(this);

//TODO these pairs could be replaced by something static that manages all active flowcharts

#if UNITY\_5\_4\_OR\_NEWER

UnityEngine.SceneManagement.SceneManager.activeSceneChanged += SceneManager\_activeSceneChanged;

#endif

}

CheckItemIds();

CleanupComponents();

UpdateVersion();

StringSubstituter.RegisterHandler(this);

}

protected virtual void OnDisable()

{

cachedFlowcharts.Remove(this);

#if UNITY\_5\_4\_OR\_NEWER

UnityEngine.SceneManagement.SceneManager.activeSceneChanged -= SceneManager\_activeSceneChanged;

#endif

StringSubstituter.UnregisterHandler(this);

}

protected virtual void UpdateVersion()

{

if (version == FungusConstants.CurrentVersion)

{

// No need to update

return;

}

// Tell all components that implement IUpdateable to update to the new version

var components = GetComponents<Component>();

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

{

var component = components[i];

IUpdateable u = component as IUpdateable;

if (u != null)

{

u.UpdateToVersion(version, FungusConstants.CurrentVersion);

}

}

version = FungusConstants.CurrentVersion;

}

protected virtual void CheckItemIds()

{

// Make sure item ids are unique and monotonically increasing.

// This should always be the case, but some legacy Flowcharts may have issues.

List<int> usedIds = new List<int>();

var blocks = GetComponents<Block>();

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

{

var block = blocks[i];

if (block.ItemId == -1 || usedIds.Contains(block.ItemId))

{

block.ItemId = NextItemId();

}

usedIds.Add(block.ItemId);

}

var commands = GetComponents<Command>();

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

{

var command = commands[i];

if (command.ItemId == -1 || usedIds.Contains(command.ItemId))

{

command.ItemId = NextItemId();

}

usedIds.Add(command.ItemId);

}

}

protected virtual void CleanupComponents()

{

// Delete any unreferenced components which shouldn't exist any more

// Unreferenced components don't have any effect on the flowchart behavior, but

// they waste memory so should be cleared out periodically.

// Remove any null entries in the variables list

// It shouldn't happen but it seemed to occur for a user on the forum

variables.RemoveAll(item => item == null);

var allVariables = GetComponents<Variable>();

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

{

var variable = allVariables[i];

if (!variables.Contains(variable))

{

DestroyImmediate(variable);

}

}

var blocks = GetComponents<Block>();

var commands = GetComponents<Command>();

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

{

var command = commands[i];

bool found = false;

for (int j = 0; j < blocks.Length; j++)

{

var block = blocks[j];

if (block.CommandList.Contains(command))

{

found = true;

break;

}

}

if (!found)

{

DestroyImmediate(command);

}

}

var eventHandlers = GetComponents<EventHandler>();

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

{

var eventHandler = eventHandlers[i];

bool found = false;

for (int j = 0; j < blocks.Length; j++)

{

var block = blocks[j];

if (block.\_EventHandler == eventHandler)

{

found = true;

break;

}

}

if (!found)

{

DestroyImmediate(eventHandler);

}

}

}

protected virtual Block CreateBlockComponent(GameObject parent)

{

Block block = parent.AddComponent<Block>();

return block;

}

（2）狼人场景动画

[ExecuteInEditMode]

public class Flowchart : MonoBehaviour, ISubstitutionHandler

{

[HideInInspector]

[SerializeField] protected int version = 0; // Default to 0 to always trigger an update for older versions of Fungus.

[HideInInspector]

[SerializeField] protected Vector2 scrollPos;

[HideInInspector]

[SerializeField] protected Vector2 variablesScrollPos;

[HideInInspector]

[SerializeField] protected bool variablesExpanded = true;

[HideInInspector]

[SerializeField] protected float blockViewHeight = 400;

[HideInInspector]

[SerializeField] protected float zoom = 1f;

[HideInInspector]

[SerializeField] protected Rect scrollViewRect;

[HideInInspector]

[SerializeField] protected List<Block> selectedBlocks = new List<Block>();

[HideInInspector]

[SerializeField] protected List<Command> selectedCommands = new List<Command>();

[HideInInspector]

[SerializeField] protected List<Variable> variables = new List<Variable>();

[TextArea(3, 5)]

[Tooltip("Description text displayed in the Flowchart editor window")]

[SerializeField] protected string description = "";

[Range(0f, 5f)]

[Tooltip("Adds a pause after each execution step to make it easier to visualise program flow. Editor only, has no effect in platform builds.")]

[SerializeField] protected float stepPause = 0f;

[Tooltip("Use command color when displaying the command list in the Fungus Editor window")]

[SerializeField] protected bool colorCommands = true;

[Tooltip("Hides the Flowchart block and command components in the inspector. Deselect to inspect the block and command components that make up the Flowchart.")]

[SerializeField] protected bool hideComponents = true;

[Tooltip("Saves the selected block and commands when saving the scene. Helps avoid version control conflicts if you've only changed the active selection.")]

[SerializeField] protected bool saveSelection = true;

[Tooltip("Unique identifier for this flowchart in localized string keys. If no id is specified then the name of the Flowchart object will be used.")]

[SerializeField] protected string localizationId = "";

[Tooltip("Display line numbers in the command list in the Block inspector.")]

[SerializeField] protected bool showLineNumbers = false;

[Tooltip("List of commands to hide in the Add Command menu. Use this to restrict the set of commands available when editing a Flowchart.")]

[SerializeField] protected List<string> hideCommands = new List<string>();

[Tooltip("Lua Environment to be used by default for all Execute Lua commands in this Flowchart")]

[SerializeField] protected LuaEnvironment luaEnvironment;

[Tooltip("The ExecuteLua command adds a global Lua variable with this name bound to the flowchart prior to executing.")]

[SerializeField] protected string luaBindingName = "flowchart";

protected static List<Flowchart> cachedFlowcharts = new List<Flowchart>();

protected static bool eventSystemPresent;

protected StringSubstituter stringSubstituer;

#if UNITY\_5\_4\_OR\_NEWER

#else

protected virtual void OnLevelWasLoaded(int level)

{

LevelWasLoaded();

}

#endif

protected virtual void LevelWasLoaded()

{

// Reset the flag for checking for an event system as there may not be one in the newly loaded scene.

eventSystemPresent = false;

}

protected virtual void Start()

{

CheckEventSystem();

}

// There must be an Event System in the scene for Say and Menu input to work.

// This method will automatically instantiate one if none exists.

protected virtual void CheckEventSystem()

{

if (eventSystemPresent)

{

return;

}

EventSystem eventSystem = GameObject.FindObjectOfType<EventSystem>();

if (eventSystem == null)

{

// Auto spawn an Event System from the prefab

GameObject prefab = Resources.Load<GameObject>("Prefabs/EventSystem");

if (prefab != null)

{

GameObject go = Instantiate(prefab) as GameObject;

go.name = "EventSystem";

}

}

eventSystemPresent = true;

}

private void SceneManager\_activeSceneChanged(UnityEngine.SceneManagement.Scene arg0, UnityEngine.SceneManagement.Scene arg1)

{

LevelWasLoaded();

}

protected virtual void OnEnable()

{

if (!cachedFlowcharts.Contains(this))

{

cachedFlowcharts.Add(this);

//TODO these pairs could be replaced by something static that manages all active flowcharts

#if UNITY\_5\_4\_OR\_NEWER

UnityEngine.SceneManagement.SceneManager.activeSceneChanged += SceneManager\_activeSceneChanged;

#endif

}

CheckItemIds();

CleanupComponents();

UpdateVersion();

StringSubstituter.RegisterHandler(this);

}

protected virtual void OnDisable()

{

cachedFlowcharts.Remove(this);

#if UNITY\_5\_4\_OR\_NEWER

UnityEngine.SceneManagement.SceneManager.activeSceneChanged -= SceneManager\_activeSceneChanged;

#endif

StringSubstituter.UnregisterHandler(this);

}

protected virtual void UpdateVersion()

{

if (version == FungusConstants.CurrentVersion)

{

// No need to update

return;

}

// Tell all components that implement IUpdateable to update to the new version

var components = GetComponents<Component>();

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

{

var component = components[i];

IUpdateable u = component as IUpdateable;

if (u != null)

{

u.UpdateToVersion(version, FungusConstants.CurrentVersion);

}

}

version = FungusConstants.CurrentVersion;

}

protected virtual void CheckItemIds()

{

// Make sure item ids are unique and monotonically increasing.

// This should always be the case, but some legacy Flowcharts may have issues.

List<int> usedIds = new List<int>();

var blocks = GetComponents<Block>();

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

{

var block = blocks[i];

if (block.ItemId == -1 || usedIds.Contains(block.ItemId))

{

block.ItemId = NextItemId();

}

usedIds.Add(block.ItemId);

}

var commands = GetComponents<Command>();

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

{

var command = commands[i];

if (command.ItemId == -1 || usedIds.Contains(command.ItemId))

{

command.ItemId = NextItemId();

}

usedIds.Add(command.ItemId);

}

}

protected virtual void CleanupComponents()

{

// Delete any unreferenced components which shouldn't exist any more

// Unreferenced components don't have any effect on the flowchart behavior, but

// they waste memory so should be cleared out periodically.

// Remove any null entries in the variables list

// It shouldn't happen but it seemed to occur for a user on the forum

variables.RemoveAll(item => item == null);

var allVariables = GetComponents<Variable>();

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

{

var variable = allVariables[i];

if (!variables.Contains(variable))

{

DestroyImmediate(variable);

}

}

var blocks = GetComponents<Block>();

var commands = GetComponents<Command>();

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

{

var command = commands[i];

bool found = false;

for (int j = 0; j < blocks.Length; j++)

{

var block = blocks[j];

if (block.CommandList.Contains(command))

{

found = true;

break;

}

}

if (!found)

{

DestroyImmediate(command);

}

}

var eventHandlers = GetComponents<EventHandler>();

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

{

var eventHandler = eventHandlers[i];

bool found = false;

for (int j = 0; j < blocks.Length; j++)

{

var block = blocks[j];

if (block.\_EventHandler == eventHandler)

{

found = true;

break;

}

}

if (!found)

{

DestroyImmediate(eventHandler);

}

}

}

protected virtual Block CreateBlockComponent(GameObject parent)

{

Block block = parent.AddComponent<Block>();

return block;

}

### 2.2游戏操作代码

（1）角色选择代码

void Start () {

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClicks);

}

public void OnClicks()

{

if (MapCreation.gameChoice == 1)

SceneManager.LoadScene("people");

else if (MapCreation.gameChoice == 0)

SceneManager.LoadScene("wolf");

}

（2）地图的自动生成代码

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MapCreation : MonoBehaviour {

//draft:37=9+1+8+1+8+1+9；29=2+4+3+5+3+5+3+4

//用来装饰初始化地图（37\*29）所需物体的数组

//0--普通地板（21.25像素）；1--竖直墙体；2--水平墙体；

//3--Wall5（图片）；4--Wall27；5--；6--；

//7--竖直3块墙体右墙面；8--竖直3块墙体左墙面

//9--竖直三块墙体小墙面；10--朝上一块墙面；11--数值一块墙面

//12--入口；//13--出口

public GameObject[] item;

//关卡数值

public int level = 1;

//将一整个迷宫以二维数组的形式存放

//0代表这个区域可以行走，-1代表障碍，1代表（一段时间内）行走过该区域，下同

private static int [,]maze;

public static int getMaze(int x, int y)

{

if(x >= 37 || y >= 29 || x < 0 || y < 0)

{

//不能行得通的块

return -1;

}

else

{

return maze[x, y];

}

}

//将该迷宫简化成4\*4，即分别从左到右，从上到下编号1-16，并记录大块起点和终点位置

private bool [,]mazeBlock;

public static int startx, starty;

public static int endx, endy;

//结构体，用来保存相邻两块区域之间所有的障碍块（区域）以及可能联通的（区域）

struct NextBlockInformation

{

//相邻区域块号，计算方式(i+1)\*4+(j+1)

public int blockNumber1, blockNumber2;

//记录障碍块区域的左上角位置和右下角坐标位置

public int blockStartX, blockStartY;

public int blockEndX, blockEndY;

//记录联通块区域的左上角位置和右下角坐标位置

public int blankStartX, blankStartY;

public int blankEndX, blankEndY;

};

NextBlockInformation[] infor;

//游戏选择（0-狼人，1-玩家）

public static int gameChoice = 0;

//角色列表（0-狼人，1-玩家，2-主角()，3-猎人）

public GameObject[] characterList;

//道具列表

public GameObject[] itemList;

//时间计时

int timeval;

//人物生成次数

int people\_times;

//道具生成次数

int item\_times;

//实例化

private void Awake()

{

//初始化时间序列

timeval = 0;

//初始化人物生成次数

people\_times = 0;

//初始化道具生成次数

item\_times = 0;

print(gameChoice);

//初始化迷宫

maze = new int[37, 29];

for (int i = 0; i < 37; i++)

{

for (int j = 0; j < 29; j++)

{

maze[i, j] = 0;

}

}

print(gameChoice);

//初始化迷宫块

mazeBlock = new bool[4, 4];

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

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

{

mazeBlock[i, j] = false;

}

}

print(gameChoice);

//生成背景图

CreateBackground();

print(gameChoice);

//随机生成起始点，并初始化一条路可通的迷宫

startx = 3 \* Random.Range(0, 2);

starty = 3 \* Random.Range(0, 2);

WalkBlock(startx, starty, 1);

print(gameChoice);

/\*for(int i=0;i<4;i++)

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

{

print(mazeBlock[i,j]+" "+i+" "+j);

}

\*/

print(gameChoice);

//生成墙体

CreateWall();

print(gameChoice);

//CreateItem(item[4],new Vector3(-10,-10,0),Quaternion.identity);

print(gameChoice);

//生成敌方角色

CreateEnemy();

print(gameChoice);

//生成4个道具

CreateItems(4);

//生成入口和出口

}

// Update is called once per frame

void Update () {

if(timeval >= 3600)

{

timeval = 0;

people\_times = people\_times + 1;

item\_times = item\_times + 1;

//选择狼人吃人类

if (gameChoice == 0)

{

//最多生成4次人物以及10次新道具

if (people\_times <= 4)

{

//6区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, 3, 0), Quaternion.identity);

//7区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, 3, 0), Quaternion.identity);

//10区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, -5, 0), Quaternion.identity);

//11区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, -5, 0), Quaternion.identity);

}

else

{

if(GameObject.Find("MapCreation/People(Clone)") == null)

{

//场上所有人物都被吃光，进入下一关

MainCharacter.youWin = true;

MainCharacter.isOver = true;

}

}

//最多生成10次道具

if(item\_times <= 10)

{

CreateItems(5);

}

}

//选择了人类

else

{

//最多生成10次道具

if (item\_times <= 10)

{

CreateItems(5);

}

}

}

else

{

timeval = timeval + 1;

}

}

//重新将Instantiate封装成一个方法，用CreateItem函数实现，防止显得太乱

//每个物品生成之后会直接生成在菜单下面，为了放在MapCreation下面，需要将每个物体生成之后通过transform.setparent放置，语句太多可以通过函数简化

private void CreateItem(GameObject createGameObject, Vector3 createPosition, Quaternion createRotation)

{

//实例化游戏物体

GameObject itemGo = Instantiate(createGameObject, createPosition, createRotation);

//将游戏物品放到Unity3d--MapCreation对象下面

itemGo.transform.SetParent(gameObject.transform);

}

//生成地面以及周围墙体

private void CreateBackground()

{

//生成地板

for (int i = -18; i <= 18; i++)

{

for (int j = -14; j <= 14; j++)

{

CreateItem(item[0], new Vector3(i, j, 0), Quaternion.identity);

maze[i + 18, j + 14] = 0;

}

}

//生成墙体

CreateItem(item[1], new Vector3(-18.7f, 0, 0), Quaternion.identity);

CreateItem(item[1], new Vector3(18.7f, 0, 0), Quaternion.identity);

CreateItem(item[2], new Vector3(0, -14.7f, 0), Quaternion.identity);

CreateItem(item[2], new Vector3(0, 14.7f, 0), Quaternion.identity);

}

//生成墙体的代码

private void CreateWall()

{

//初始化大块墙面之间的间隔

InitNextBlockInformation();

//构造竖直墙面

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

{

for(int j = 0; j < 3; j++)

{

//两个迷宫块之间有通路，在这里面不要加障碍

if (mazeBlock[i, j]==true&& mazeBlock[i, j+1]==true)

{

//根据第几行第几列转换为数值

int ans = (i \* 4 + (j + 1)) + (i \* 4 + (j + 2));

//将矩阵对应下标值变为-1

for(int m = infor[ans].blockStartX;m<= infor[ans].blankStartX; m++)

{

for(int n = infor[ans].blockStartY; n >= infor[ans].blankStartY; n--)

{

maze[(m+18), (-n+14)] = -1;

if (n == infor[ans].blankStartY)

{

CreateItem(item[9], new Vector3(m, n+1, 0), Quaternion.identity);

if(i!=0)

CreateItem(item[11], new Vector3(m, n+3, 0), Quaternion.identity);

}

else

if (i != 0)

CreateItem(item[11], new Vector3(m, n, 0), Quaternion.identity);

}

}

for (int m = infor[ans].blankEndX; m <= infor[ans].blockEndX; m++)

{

for (int n = infor[ans].blankEndY; n >= infor[ans].blockEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

if(n == infor[ans].blankEndY)

CreateItem(item[10], new Vector3(m, n, 0), Quaternion.identity);

else

CreateItem(item[11], new Vector3(m, n, 0), Quaternion.identity);

}

}

}

//否则两个迷宫块之间没有通路，在这里面根据以随机概率的形式要不要让他们形成通路

else

{

int border = 85;

//if (level == 1) border = 50;

//else if (level == 2) border = 70;

//else if (level == 3) border = 90;

bool setblock;

//随机概率，设置两个区域之间相联通

if (Random.Range(1, 101) <= border)

{

//根据第几行第几列转换为数值

int ans = (i \* 4 + (j + 1)) + (i \* 4 + (j + 2)); ;

//将矩阵对应下标值变为-1

for (int m = infor[ans].blockStartX; m <= infor[ans].blankStartX; m++)

{

for (int n = infor[ans].blockStartY; n >= infor[ans].blankStartY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

if (n == infor[ans].blankStartY)

{

CreateItem(item[9], new Vector3(m, n + 1, 0), Quaternion.identity);

if(i!=0)

CreateItem(item[11], new Vector3(m, n +3 , 0), Quaternion.identity);

}

else

if (i != 0)

CreateItem(item[11], new Vector3(m, n, 0), Quaternion.identity);

}

}

for (int m = infor[ans].blankEndX; m <= infor[ans].blockEndX; m++)

{

for (int n = infor[ans].blankEndY; n >= infor[ans].blockEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

if (n == infor[ans].blankEndY)

CreateItem(item[10], new Vector3(m, n, 0), Quaternion.identity);

else

CreateItem(item[11], new Vector3(m, n, 0), Quaternion.identity);

}

}

}

//随机概率，设置两个区域不连通

else

{

//根据第几行第几列转换为数值

int ans = (i \* 4 + (j + 1)) + (i \* 4 + (j + 2)); ;

//将矩阵对应下标值变为-1

for (int m = infor[ans].blockStartX; m <= infor[ans].blockEndX; m++)

{

for (int n = infor[ans].blockStartY; n >= infor[ans].blockEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

CreateItem(item[11], new Vector3(m, n, 0), Quaternion.identity);

if(n == infor[ans].blockStartY)

{

CreateItem(item[11], new Vector3(m, n+1, 0), Quaternion.identity);

CreateItem(item[11], new Vector3(m, n+2, 0), Quaternion.identity);

}

}

}

}

}

}

}

//构造水平墙面

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

{

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

{

/\*\*/

//两个迷宫块之间有通路，在这里面不要加障碍

if (mazeBlock[i, j] == true && mazeBlock[i + 1, j] == true)

{

//根据第几行第几列转换为数值

int ans = (i \* 4 + (j + 1)) + ((i + 1) \* 4 + (j + 1));

//将矩阵对应下标值变为-1

for (int m = infor[ans].blockStartX; m <= infor[ans].blankStartX; m++)

{

for (int n = infor[ans].blockStartY; n >= infor[ans].blankEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

//CreateItem(item[6], new Vector3(m, n, 0), Quaternion.identity);

}

if(m== infor[ans].blankStartX)

CreateItem(item[7], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

else

CreateItem(item[6], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

}

for (int m = infor[ans].blankEndX; m <= infor[ans].blockEndX; m++)

{

for (int n = infor[ans].blankStartY; n >= infor[ans].blockEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

//CreateItem(item[6], new Vector3(m, n, 0), Quaternion.identity);

}

if (m == infor[ans].blankEndX)

CreateItem(item[8], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

else

CreateItem(item[6], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

}

}

//否则两个迷宫块之间没有通路，在这里面根据以随机概率的形式要不要让他们形成通路

else

{

int border = 95;

//边界区域有围墙的概率

//if (level == 1) border = 35;

//else if (level == 2) border = 50;

//else if (level == 3) border = 65;

bool setblock;

//随机概率，设置两个区域之间相联通

if (Random.Range(1, 101) <= border)

{

//根据第几行第几列转换为数值

int ans = (i \* 4 + (j + 1)) + ((i + 1) \* 4 + (j + 1));

//将矩阵对应下标值变为-1

for (int m = infor[ans].blockStartX; m <= infor[ans].blankStartX; m++)

{

//for (int n = infor[ans].blockStartY; n >= infor[ans].blankStartY; n--)

for (int n = infor[ans].blockStartY; n >= infor[ans].blankEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

//CreateItem(item[6], new Vector3(m, n, 0), Quaternion.identity);

}

if (m == infor[ans].blankStartX)

CreateItem(item[7], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

else

CreateItem(item[6], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

}

for (int m = infor[ans].blankEndX; m <= infor[ans].blockEndX; m++)

{

//for (int n = infor[ans].blankEndY; n >= infor[ans].blockEndY; n--)

for (int n = infor[ans].blankStartY; n >= infor[ans].blockEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

//CreateItem(item[6], new Vector3(m, n, 0), Quaternion.identity);

}

if (m == infor[ans].blankEndX)

CreateItem(item[8], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

else

CreateItem(item[6], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

}

}

//随机概率，设置两个区域不连通

else

{

//根据第几行第几列转换为数值

int ans = (i \* 4 + (j + 1)) + ((i + 1) \* 4 + (j + 1));

//将矩阵对应下标值变为-1

for (int m = infor[ans].blockStartX; m <= infor[ans].blockEndX; m++)

{

for (int n = infor[ans].blockStartY; n >= infor[ans].blockEndY; n--)

{

maze[(m + 18), (-n + 14)] = -1;

//CreateItem(item[6], new Vector3(m, n, 0), Quaternion.identity);

}

CreateItem(item[6], new Vector3(m, infor[ans].blockStartY - 1, 0), Quaternion.identity);

}

}

}

}

}

//构造水平和竖直墙面之间的交叉墙面

CreateItem(item[4], new Vector3(0, 0, 0), Quaternion.identity);

maze[18, 14] = -1;

maze[18, 15] = -1;

maze[18, 16] = -1;

CreateItem(item[4], new Vector3(9, 0, 0), Quaternion.identity);

maze[27, 14] = -1;

maze[27, 15] = -1;

maze[27, 16] = -1;

CreateItem(item[4], new Vector3(-9, 0, 0), Quaternion.identity);

maze[9, 14] = -1;

maze[9, 15] = -1;

maze[9, 16] = -1;

CreateItem(item[4], new Vector3(0, 8, 0), Quaternion.identity);

maze[18, 6] = -1;

maze[18, 7] = -1;

maze[18, 8] = -1;

CreateItem(item[4], new Vector3(9, 8, 0), Quaternion.identity);

maze[27, 6] = -1;

maze[27, 7] = -1;

maze[27, 8] = -1;

CreateItem(item[4], new Vector3(-9, 8, 0), Quaternion.identity);

maze[9, 6] = -1;

maze[9, 7] = -1;

maze[9, 8] = -1;

CreateItem(item[4], new Vector3(0, -8, 0), Quaternion.identity);

maze[18, 22] = -1;

maze[18, 23] = -1;

maze[18, 24] = -1;

CreateItem(item[4], new Vector3(9, -8, 0), Quaternion.identity);

maze[27, 22] = -1;

maze[18, 23] = -1;

maze[18, 24] = -1;

CreateItem(item[4], new Vector3(-9, -8, 0), Quaternion.identity);

maze[9, 22] = -1;

maze[18, 23] = -1;

maze[18, 24] = -1;

//构造最上层墙面

for (int i = -18; i <= 18; i++)

{

if (i == -9 || i == 9) {

CreateItem(item[11], new Vector3(i, 13, 0), Quaternion.identity);

CreateItem(item[4], new Vector3(i, 14, 0), Quaternion.identity);

}

else

{

CreateItem(item[3], new Vector3(i, 13.5f, 0), Quaternion.identity);

}

maze[i + 18, 0] = -1;

maze[i + 18, 1] = -1;

}

//人类有入口和出口

if (gameChoice == 1)

{

//生成入口和出口

int ExtranceNum = startx \* 4 + (starty + 1);

int exitNum = endx \* 4 + (endy + 1);

switch (ExtranceNum)

{

//maze[(m + 18), (-n + 14)] = -1;

case 1:

CreateItem(item[12], new Vector3(-14, 11.5f, 0), Quaternion.identity);

maze[5, 3] = -1;

maze[6, 3] = -1;

maze[7, 3] = -1;

maze[5, 2] = -1;

maze[6, 2] = -1;

maze[7, 2] = -1;

break;

case 4:

CreateItem(item[12], new Vector3(14, 11.5f, 0), Quaternion.identity);

maze[31, 3] = -1;

maze[32, 3] = -1;

maze[33, 3] = -1;

maze[31, 2] = -1;

maze[32, 2] = -1;

maze[33, 2] = -1;

break;

case 13:

CreateItem(item[12], new Vector3(-14, -13.5f, 0), Quaternion.identity);

maze[5, 27] = -1;

maze[6, 27] = -1;

maze[7, 27] = -1;

maze[5, 28] = -1;

maze[6, 28] = -1;

maze[7, 28] = -1;

break;

case 16:

CreateItem(item[12], new Vector3(14, -13.5f, 0), Quaternion.identity);

maze[31, 27] = -1;

maze[32, 27] = -1;

maze[33, 27] = -1;

maze[31, 28] = -1;

maze[32, 28] = -1;

maze[33, 28] = -1;

break;

default:

break;

}

switch (exitNum)

{

case 1:

CreateItem(item[13], new Vector3(-14, 11.5f, 0), Quaternion.identity);

maze[5, 3] = -1;

maze[6, 3] = -1;

maze[7, 3] = -1;

maze[5, 2] = -1;

maze[6, 2] = -1;

maze[7, 2] = -1;

break;

case 2:

CreateItem(item[13], new Vector3(-5, 11.5f, 0), Quaternion.identity);

maze[14, 3] = -1;

maze[13, 3] = -1;

maze[12, 3] = -1;

maze[14, 2] = -1;

maze[13, 2] = -1;

maze[12, 2] = -1;

break;

case 3:

CreateItem(item[13], new Vector3(5, 11.5f, 0), Quaternion.identity);

maze[22, 3] = -1;

maze[23, 3] = -1;

maze[24, 3] = -1;

maze[22, 2] = -1;

maze[23, 2] = -1;

maze[24, 2] = -1;

break;

case 4:

CreateItem(item[13], new Vector3(14, 11.5f, 0), Quaternion.identity);

maze[31, 3] = -1;

maze[32, 3] = -1;

maze[33, 3] = -1;

maze[31, 2] = -1;

maze[32, 2] = -1;

maze[33, 2] = -1;

break;

case 5:

CreateItem(item[13], new Vector3(-17, 3.5f, 0), Quaternion.identity);

maze[2, 11] = -1;

maze[1, 11] = -1;

maze[0, 11] = -1;

maze[2, 10] = -1;

maze[1, 10] = -1;

maze[0, 10] = -1;

break;

case 8:

CreateItem(item[13], new Vector3(17, 3.5f, 0), Quaternion.identity);

maze[34, 11] = -1;

maze[35, 11] = -1;

maze[36, 11] = -1;

maze[34, 10] = -1;

maze[35, 10] = -1;

maze[36, 10] = -1;

break;

case 9:

CreateItem(item[13], new Vector3(-17, -5.5f, 0), Quaternion.identity);

maze[2, 19] = -1;

maze[1, 19] = -1;

maze[0, 19] = -1;

maze[2, 20] = -1;

maze[1, 20] = -1;

maze[0, 20] = -1;

break;

case 12:

CreateItem(item[13], new Vector3(17, -5.5f, 0), Quaternion.identity);

maze[2, 19] = -1;

maze[1, 19] = -1;

maze[0, 19] = -1;

maze[2, 20] = -1;

maze[1, 20] = -1;

maze[0, 20] = -1;

break;

case 13:

CreateItem(item[13], new Vector3(-14, -13.5f, 0), Quaternion.identity);

maze[5, 27] = -1;

maze[6, 27] = -1;

maze[7, 27] = -1;

maze[5, 28] = -1;

maze[6, 28] = -1;

maze[7, 28] = -1;

break;

case 14:

CreateItem(item[13], new Vector3(-5, -13.5f, 0), Quaternion.identity);

maze[14, 27] = -1;

maze[13, 27] = -1;

maze[12, 27] = -1;

maze[14, 28] = -1;

maze[13, 28] = -1;

maze[12, 28] = -1;

break;

case 15:

CreateItem(item[13], new Vector3(5, -13.5f, 0), Quaternion.identity);

maze[22, 27] = -1;

maze[23, 27] = -1;

maze[24, 27] = -1;

maze[22, 28] = -1;

maze[23, 28] = -1;

maze[24, 28] = -1;

break;

case 16:

CreateItem(item[13], new Vector3(14, -13.5f, 0), Quaternion.identity);

maze[31, 27] = -1;

maze[32, 27] = -1;

maze[33, 27] = -1;

maze[31, 28] = -1;

maze[32, 28] = -1;

maze[33, 28] = -1;

break;

default:break;

}

}

print(gameChoice);

//print(maze);

/\*

for(int j = 0; j < 29; j++) {

for(int i = 0; i < 37; i++)

{

print(i+" "+j+" "+maze[i,j]);

}

}\*/

}

//生成道具的代码（生成道具的数量）

private void CreateItems(int number)

{

bool item\_canProduce;

int item\_x, item\_y, item\_num;

//这里敌方人物不作为道具生成的障碍块，因为敌方人物对于道具的生成来说没有任何影响

//这里道具生成之后就

for(int i = 1; i <= number; i++)

{

item\_canProduce = false;

while (!item\_canProduce)

{

//这里为了实现方便，道具已经生成的位置不会重新再生成

//假如有什么特别需要，可以再进行更改

//横坐标下标

item\_x = Random.Range(0, 37);

//纵坐标下标

item\_y = Random.Range(0, 29);

//道具下标，对应ItemList[下标]的道具

item\_num = Random.Range(1, 10);

if(maze[item\_x, item\_y] == 0)

{

item\_canProduce = true;

maze[item\_x, item\_y] = 1;

CreateItem(itemList[item\_num], new Vector3(item\_x-18, -item\_y+14, 0), Quaternion.identity);

}

}

}

}

//生成敌方主角的代码

private void CreateEnemy()

{

//和gameChoice有关

//print(gameChoice);

//选择人类，生成狼人

if (gameChoice == 1)

{

//1区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-18, 12, 0), Quaternion.identity);

//2区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, 12, 0), Quaternion.identity);

//3区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, 12, 0), Quaternion.identity);

//4区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(18, 12, 0), Quaternion.identity);

//5区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-18, 3, 0), Quaternion.identity);

//8区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(18, 3, 0), Quaternion.identity);

//9区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-18, -5, 0), Quaternion.identity);

//12区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(18, -5, 0), Quaternion.identity);

//13区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-18, -14, 0), Quaternion.identity);

//14区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, -14, 0), Quaternion.identity);

//15区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, -14, 0), Quaternion.identity);

//16区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(18, -14, 0), Quaternion.identity);

}

//否则生成猎人

else if(gameChoice == 0)

{

//生成四个猎人（分别在地图中1256,3478,9101314,11121516四个大块中）

CreateItem(characterList[3], new Vector3(-16, 10, 0), Quaternion.identity);

CreateItem(characterList[3], new Vector3(16, 10, 0), Quaternion.identity);

CreateItem(characterList[3], new Vector3(-16, -12, 0), Quaternion.identity);

CreateItem(characterList[3], new Vector3(16, -12, 0), Quaternion.identity);

}

//选择人类和选择狼人共同在地图的中间部分生成敌方玩家

//6区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, 3, 0), Quaternion.identity);

//7区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, 3, 0), Quaternion.identity);

//10区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(-4, -5, 0), Quaternion.identity);

//11区块

CreateItem(characterList[(1 - gameChoice)], new Vector3(4, -5, 0), Quaternion.identity);

}

//行走大块迷宫函数，输入参数分别代表：横坐标，纵坐标，行走的次数

//这里从入口到出口行走是单项的，不必担心出现死路的情况

private void WalkBlock(int x,int y,int times)

{

if (mazeBlock[x, y] == false && x>=0 && x<4 && y>=0 && y<4)

{

mazeBlock[x, y] = true;

//判断上下左右四个方位是否已经走过

bool []hasPassed = new bool[4];

//上下左右方向

int[,] direction = new int[,]{ { x, y + 1 }, { x, y - 1 }, { x - 1, y }, { x + 1, y } };

//记录行走方向

int walkDirection;

//初始化是否走过迷宫

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

{

int xx = direction[i, 0], yy = direction[i, 1];

//判断下一步是否仍然停在大的迷宫区域内

if(xx >= 0 && xx < 4 && yy >= 0 && yy < 4)

{

hasPassed[i] = mazeBlock[xx, yy];

}

else

{

hasPassed[i] = true;

}

}

//局部变量用来记录循环次数，可能会出现死路，这里刚好五步能到达边缘

int looptimes = 0;

while (true)

{

//随机决定行走的方向

walkDirection = Random.Range(0, 4);

//未行走过的退出循环

if (hasPassed[walkDirection] == false || looptimes > 30)

{

break;

}

looptimes++;

}

//假如次数大于等于五次

if (times >= 5)

{

//假如走到边缘，决定出口

if (x==0||x==3||y==0||y==3)

{

endx = x;

endy = y;

return;

}

//否则继续走

else{

WalkBlock(direction[walkDirection, 0], direction[walkDirection, 1], times + 1);

}

}

else

{

//继续走

WalkBlock(direction[walkDirection, 0], direction[walkDirection, 1], times + 1);

}

}

}

//初始化障碍块

private void InitNextBlockInformation()

{

infor = new NextBlockInformation[33];

//这里将16块区域按照从左到右，从上到下的形式编号（1-16）

//infor[n]的n表示为相邻两块区域的和

//这里的坐标以地图中心为原点，到时候矩阵maze换算的时候加上偏移值（地图中能生成东西）

infor[3].blockNumber1 = 1;

infor[3].blockNumber1 = 2;

infor[3].blockStartX = -9;

infor[3].blockStartY = 12;

infor[3].blockEndX = -9;

infor[3].blockEndY = 9;

infor[3].blankStartX = -9;

infor[3].blankStartY = 12;

infor[3].blankEndX = -9;

infor[3].blankEndY = 9;

infor[5].blockNumber1 = 2;

infor[5].blockNumber1 = 3;

infor[5].blockStartX = 0;

infor[5].blockStartY = 12;

infor[5].blockEndX = 0;

infor[5].blockEndY = 9;

infor[5].blankStartX = 0;

infor[5].blankStartY = 12;

infor[5].blankEndX = 0;

infor[5].blankEndY = 9;

infor[7].blockNumber1 = 3;

infor[7].blockNumber1 = 4;

infor[7].blockStartX = 9;

infor[7].blockStartY = 12;

infor[7].blockEndX = 9;

infor[7].blockEndY = 9;

infor[7].blankStartX = 9;

infor[7].blankStartY = 12;

infor[7].blankEndX = 9;

infor[7].blankEndY = 9;

infor[6].blockNumber1 = 1;

infor[6].blockNumber1 = 5;

infor[6].blockStartX = -18;

infor[6].blockStartY = 8;

infor[6].blockEndX = -10;

infor[6].blockEndY = 6;

infor[6].blankStartX = -16;

infor[6].blankStartY = 8;

infor[6].blankEndX = -13;

infor[6].blankEndY = 6;

infor[8].blockNumber1 = 2;

infor[8].blockNumber1 = 6;

infor[8].blockStartX = -8;

infor[8].blockStartY = 8;

infor[8].blockEndX = -1;

infor[8].blockEndY = 6;

infor[8].blankStartX = -6;

infor[8].blankStartY = 8;

infor[8].blankEndX = -3;

infor[8].blankEndY = 6;

infor[10].blockNumber1 = 3;

infor[10].blockNumber1 = 7;

infor[10].blockStartX = 1;

infor[10].blockStartY = 8;

infor[10].blockEndX = 8;

infor[10].blockEndY = 6;

infor[10].blankStartX = 3;

infor[10].blankStartY = 8;

infor[10].blankEndX = 6;

infor[10].blankEndY = 6;

infor[12].blockNumber1 = 4;

infor[12].blockNumber1 = 8;

infor[12].blockStartX = 10;

infor[12].blockStartY = 8;

infor[12].blockEndX = 18;

infor[12].blockEndY = 6;

infor[12].blankStartX = 13;

infor[12].blankStartY = 8;

infor[12].blankEndX = 16;

infor[12].blankEndY = 6;

infor[11].blockNumber1 = 5;

infor[11].blockNumber1 = 6;

infor[11].blockStartX = -9;

infor[11].blockStartY = 5;

infor[11].blockEndX = -9;

infor[11].blockEndY = 1;

infor[11].blankStartX = -9;

infor[11].blankStartY = 4;

infor[11].blankEndX = -9;

infor[11].blankEndY = 1;

infor[13].blockNumber1 = 6;

infor[13].blockNumber1 = 7;

infor[13].blockStartX = 0;

infor[13].blockStartY = 5;

infor[13].blockEndX = 0;

infor[13].blockEndY = 1;

infor[13].blankStartX = 0;

infor[13].blankStartY = 4;

infor[13].blankEndX = 0;

infor[13].blankEndY = 1;

infor[15].blockNumber1 = 7;

infor[15].blockNumber1 = 8;

infor[15].blockStartX = 9;

infor[15].blockStartY = 5;

infor[15].blockEndX = 9;

infor[15].blockEndY = 1;

infor[15].blankStartX = 9;

infor[15].blankStartY = 4;

infor[15].blankEndX = 9;

infor[15].blankEndY = 1;

infor[14].blockNumber1 = 5;

infor[14].blockNumber1 = 9;

infor[14].blockStartX = -18;

infor[14].blockStartY = 0;

infor[14].blockEndX = -10;

infor[14].blockEndY = -2;

infor[14].blankStartX = -16;

infor[14].blankStartY = 0;

infor[14].blankEndX = -13;

infor[14].blankEndY = -2;

infor[16].blockNumber1 = 6;

infor[16].blockNumber1 = 10;

infor[16].blockStartX = -8;

infor[16].blockStartY = 0;

infor[16].blockEndX = -1;

infor[16].blockEndY = -2;

infor[16].blankStartX = -6;

infor[16].blankStartY = 0;

infor[16].blankEndX = -3;

infor[16].blankEndY = -2;

infor[18].blockNumber1 = 7;

infor[18].blockNumber1 = 11;

infor[18].blockStartX = 1;

infor[18].blockStartY = 0;

infor[18].blockEndX = 8;

infor[18].blockEndY = -2;

infor[18].blankStartX = 3;

infor[18].blankStartY = 0;

infor[18].blankEndX = 6;

infor[18].blankEndY = -2;

infor[20].blockNumber1 = 8;

infor[20].blockNumber1 = 12;

infor[20].blockStartX = 10;

infor[20].blockStartY = 0;

infor[20].blockEndX = 18;

infor[20].blockEndY = -2;

infor[20].blankStartX = 13;

infor[20].blankStartY = 0;

infor[20].blankEndX = 16;

infor[20].blankEndY = -2;

infor[19].blockNumber1 = 9;

infor[19].blockNumber1 = 10;

infor[19].blockStartX = -9;

infor[19].blockStartY = -3;

infor[19].blockEndX = -9;

infor[19].blockEndY = -7;

infor[19].blankStartX = -9;

infor[19].blankStartY = -4;

infor[19].blankEndX = -9;

infor[19].blankEndY = -7;

infor[21].blockNumber1 = 10;

infor[21].blockNumber1 = 11;

infor[21].blockStartX = 0;

infor[21].blockStartY = -3;

infor[21].blockEndX = 0;

infor[21].blockEndY = -7;

infor[21].blankStartX = 0;

infor[21].blankStartY = -4;

infor[21].blankEndX = 0;

infor[21].blankEndY = -7;

infor[23].blockNumber1 = 11;

infor[23].blockNumber1 = 12;

infor[23].blockStartX = 9;

infor[23].blockStartY = -3;

infor[23].blockEndX = 9;

infor[23].blockEndY = -7;

infor[23].blankStartX = 9;

infor[23].blankStartY = -4;

infor[23].blankEndX = 9;

infor[23].blankEndY = -7;

infor[22].blockNumber1 = 9;

infor[22].blockNumber1 = 13;

infor[22].blockStartX = -18;

infor[22].blockStartY = -8;

infor[22].blockEndX = -10;

infor[22].blockEndY = -10;

infor[22].blankStartX = -16;

infor[22].blankStartY = -8;

infor[22].blankEndX = -13;

infor[22].blankEndY = -10;

infor[24].blockNumber1 = 10;

infor[24].blockNumber1 = 14;

infor[24].blockStartX = -8;

infor[24].blockStartY = -8;

infor[24].blockEndX = -1;

infor[24].blockEndY = -10;

infor[24].blankStartX = -6;

infor[24].blankStartY = -8;

infor[24].blankEndX = -3;

infor[24].blankEndY = -10;

infor[26].blockNumber1 = 11;

infor[26].blockNumber1 = 15;

infor[26].blockStartX = 1;

infor[26].blockStartY = -8;

infor[26].blockEndX = 8;

infor[26].blockEndY = -10;

infor[26].blankStartX = 3;

infor[26].blankStartY = -8;

infor[26].blankEndX = 6;

infor[26].blankEndY = -10;

infor[28].blockNumber1 = 12;

infor[28].blockNumber1 = 16;

infor[28].blockStartX = 10;

infor[28].blockStartY = -8;

infor[28].blockEndX = 18;

infor[28].blockEndY = -10;

infor[28].blankStartX = 13;

infor[28].blankStartY = -8;

infor[28].blankEndX = 16;

infor[28].blankEndY = -10;

infor[27].blockNumber1 = 13;

infor[27].blockNumber1 = 14;

infor[27].blockStartX = -9;

infor[27].blockStartY = -11;

infor[27].blockEndX = -9;

infor[27].blockEndY = -14;

infor[27].blankStartX = -9;

infor[27].blankStartY = -11;

infor[27].blankEndX = -9;

infor[27].blankEndY = -14;

infor[29].blockNumber1 = 14;

infor[29].blockNumber1 = 15;

infor[29].blockStartX = 0;

infor[29].blockStartY = -11;

infor[29].blockEndX = 0;

infor[29].blockEndY = -14;

infor[29].blankStartX = 0;

infor[29].blankStartY = -11;

infor[29].blankEndX = 0;

infor[29].blankEndY = -14;

infor[31].blockNumber1 = 15;

infor[31].blockNumber1 = 16;

infor[31].blockStartX = 9;

infor[31].blockStartY = -11;

infor[31].blockEndX = 9;

infor[31].blockEndY = -14;

infor[31].blankStartX = 9;

infor[31].blankStartY = -11;

infor[31].blankEndX = 9;

infor[31].blankEndY = -14;

}

}

（3）角色控制及道具使用代码

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class MainCharacter : MonoBehaviour

{

//变量委托，可能要通过道具单机事件改变人物的属性

//要通过实例改变属性必须在start/update里面改变！

//public static MainCharacter Instance

//{

// public set;

// public get;

//}

//预置体列表（1-玩家/0-狼人/2-傀儡(仅改变tag)）

public GameObject[] prefabList;

public Sprite[] spriteList;

public RuntimeAnimatorController[] animaList;

//道具列表（1-猎枪,2-防狼喷雾,3-铲子,4-照明弹,5-隐身,6-傀儡,7-时光药水,8-体力药水,9-手电筒）

public GameObject[] itemList;

//定义人物的属性

//角色等级

public int level = 1;

//游戏分数

public int score = 0;

//角色速度

public float speed = 0.5f;

//离敌方距离

public static float distance = 0f;

//角色位置

public Vector2 dest = Vector2.zero;

//角色体力

public int healthPoint;

//角色最大体力

public int max\_healthPoint;

//视野范围

public float visibleRange;

//当前状态（state=0(宿夜),state=1(普通之夜),state=2(月圆之夜)）

public static int state;

//关卡时间

public long levelTime;

//剩余时间

public long currentTime;

//装备总量

public static int itemNum = 0;

//所有装备

//((1-拳头,2-防狼喷雾,3-铲子,4-照明弹,5-隐身衣,6-傀儡,7-时光药水,8-体力药水,9-手电筒))

//装备箱（已有装备，最大容量为8）

public int[] itemBox;

//装备最大使用时间

public long[] item\_canUseTime;

//装备使用剩余时间

public long[] item\_remainedTime;

//判断角色当前状态是否在移动

public int[] isOnMove;

//定义游戏是否结束

public static bool isOver = false;

public static bool youWin = false;

//判断体力耗尽，角色是否可以移动

public static bool canMove = false;

//判断角色碰到碰撞块，角色是否以及加分

public static bool canAddScore = false;

//定义照明单时间

public float timer = 600f;

//定义遮罩图层1

public Texture mask1;

//定义遮罩图层2

public Texture mask2;

//定义遮罩图层3

public Texture mask3;

//定义遮罩材料

public Material sd;

//照明弹标志位

private int koo;

private void Awake()

{

}

// Use this for initialization

void Start()

{

//Instance = this;

//print(GetComponent<Renderer>().material.color);

//初始化游戏玩家（游戏选择可以通过调用获得）

if (MapCreation.gameChoice == 0)

{

GameObject.Find("character\_value").GetComponent<Text>().text = "狼人";

}

else

{

GameObject.Find("character\_value").GetComponent<Text>().text = "人类";

}

spriteList = new Sprite[2];

spriteList[0] = prefabList[0].GetComponent<SpriteRenderer>().sprite;

spriteList[1] = prefabList[1].GetComponent<SpriteRenderer>().sprite;

//狼人

if (MapCreation.gameChoice == 0)

{

this.transform.localScale = new Vector3(1f, 1f, 0f);

GameObject.Find("MainCharacter").GetComponent<BoxCollider2D>().size = new Vector3(1f, 0.6f, 0f);

GameObject.Find("MainCharacter").GetComponent<BoxCollider2D>().offset = new Vector3(0f, -0.2f, 0f);

}

GameObject.Find("MainCharacter").GetComponent<SpriteRenderer>().sprite = spriteList[MapCreation.gameChoice];

GameObject.Find("MainCharacter").GetComponent<Animator>().runtimeAnimatorController = animaList[MapCreation.gameChoice];

//初始化玩家的游戏位置

int startx = MapCreation.startx;

int starty = MapCreation.starty;

int endx = MapCreation.endx;

int endy = MapCreation.endy;

if (startx == 0 && starty == 0)

this.transform.position = new Vector3(-14f, 10f, 0f);

else if (startx == 3 && starty == 0)

this.transform.position = new Vector3(-14f, -12f, 0f);

else if (startx == 0 && starty == 3)

this.transform.position = new Vector3(14f, 10f, 0f);

else

this.transform.position = new Vector3(14f, -12f, 0f);

dest = this.transform.position;

//初始化游戏玩家的属性

//先定义角色能移动

canMove = true;

//角色等级

level = 1;

//游戏分数

score = 0;

//角色速度

speed = 0.5f + 0.05f \* (level - 1);

//离敌方距离

//distance = 0f;

//角色最大体力

max\_healthPoint = 500;// + 10 \* (level - 1);

//角色体力

healthPoint = max\_healthPoint;

//视野范围

//visibleRange;

//当前状态（state=0(宿夜),state=1(普通之夜),state=2(月圆之夜)）

state = 1;

//关卡时间（数字显示：18000/3600=5分钟，每60个数字为1秒）

levelTime = 18000;

//剩余时间

currentTime = levelTime;

//装备总量

itemNum = 0;

//装备箱（已有装备，最大容量为8，每个数组存放装备编号）

itemBox = new int[8];

item\_canUseTime = new long[10];

item\_remainedTime = new long[10];

for (int i = 0; i < 8; i++)

{

itemBox[i] = 0;

}

for (int i = 0; i < 10; i++)

{

item\_canUseTime[i] = 0;

item\_remainedTime[i] = 0;

}

//隐身衣和防狼喷雾的作用时间为10秒

item\_canUseTime[5] = 600;

item\_canUseTime[2] = 600;

//默认情况下角色碰到碰撞块可以加分

canAddScore = true;

//this.gameObject.GetComponent<AudioSource>().Pause();

this.gameObject.GetComponent<AudioSource>().mute = true;

}

// Update is called once per frame

void Update()

{

}

// Destroy

void OnDestroy()

{

//if (Instance != null)

//{

// Instance = null;

//}

}

//向上移动函数

public void Ononeup()

{

dest = (Vector2)this.transform.position + Vector2.up \* speed;

Vector2 dir = dest - (Vector2)this.transform.position;

//changeOnMove(0);

GetComponent<Animator>().SetFloat("drix", dir.x);

GetComponent<Animator>().SetFloat("driy", dir.y);

}

//向下移动函数

public void Ononedown()

{

dest = (Vector2)this.transform.position + Vector2.down \* speed;

Vector2 dir = dest - (Vector2)this.transform.position;

//changeOnMove(1);

GetComponent<Animator>().SetFloat("drix", dir.x);

GetComponent<Animator>().SetFloat("driy", dir.y);

}

//向右移动函数

public void Ononeright()

{

dest = (Vector2)this.transform.position + Vector2.right \* speed;

Vector2 dir = dest - (Vector2)this.transform.position;

//changeOnMove(2);

GetComponent<Animator>().SetFloat("drix", dir.x);

GetComponent<Animator>().SetFloat("driy", dir.y);

}

//向左移动函数

public void Ononeleft()

{

dest = (Vector2)this.transform.position + Vector2.left \* speed;

Vector2 dir = dest - (Vector2)this.transform.position;

//changeOnMove(3);

GetComponent<Animator>().SetFloat("drix", dir.x);

GetComponent<Animator>().SetFloat("driy", dir.y);

}

//移动的时候属性变化

public void changeOnMove()

{

if (canAddScore == true)

{

//体力

healthPoint = healthPoint - 1;

//分数（根据夜晚的状态变化）

//宿夜

if (state == 0)

{

if (MapCreation.gameChoice == 0)

//狼人

score = (int)((float)score + Mathf.Floor(speed \* 3));

else

//人类

score = (int)((float)score + Mathf.Floor(speed \* 5));

}

//普通之夜

else if (state == 1)

score = (int)((float)score + Mathf.Floor(speed \* 4));

//月圆之夜

else

{

if (MapCreation.gameChoice == 0)

//狼人

score = (int)((float)score + Mathf.Floor(speed \* 5));

else

//人类

score = (int)((float)score + Mathf.Floor(speed \* 3));

}

//print(score);

//isOnMove[index] = 0;

}

}

//背包1的事件

public void bag1Event()

{

//item0为透明的，就是什么也没有

bool ans = useItem(1);

if (ans == true)

{

GameObject.Find("bag1\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[0] = 0;

}

}

//背包2的事件

public void bag2Event()

{

//item0为透明的，就是什么也没有

bool ans = useItem(2);

if (ans == true)

{

GameObject.Find("bag2\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[1] = 0;

}

}

//背包3的事件

public void bag3Event()

{

//item0为透明的，就是什么也没有

bool ans = useItem(3);

if (ans == true)

{

GameObject.Find("bag3\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[2] = 0;

}

}

//背包4的事件

public void bag4Event()

{

bool ans = useItem(4);

if (ans == true)

{

//item0为透明的，就是什么也没有

GameObject.Find("bag4\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[3] = 0;

}

}

//背包5的事件

public void bag5Event()

{

//item0为透明的，就是什么也没有

bool ans = useItem(5);

if (ans == true)

{

GameObject.Find("bag5\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[4] = 0;

}

}

//背包6的事件

public void bag6Event()

{

bool ans = useItem(6);

if (ans == true)

{

//item0为透明的，就是什么也没有

GameObject.Find("bag6\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[5] = 0;

}

}

//背包7的事件

public void bag7Event()

{

//item0为透明的，就是什么也没有

bool ans = useItem(7);

if (ans == true)

{

GameObject.Find("bag7\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[6] = 0;

}

}

//背包8的事件

public void bag8Event()

{

//item0为透明的，就是什么也没有

bool ans = useItem(2);

if (ans == true)

{

GameObject.Find("bag8\_image").GetComponent<Image>().sprite = itemList[0].GetComponent<SpriteRenderer>().sprite;

itemBox[7] = 0;

}

}

//使用道具函数（传入参数为背包数组下标）

public bool useItem(int index)

{

//使用结果作为返回值，成功使用返回true，否返回false

bool useAns = true;

int itemName = itemBox[index - 1];

GameObject obj = null;

switch (itemName)

{

//人类和狼人共同拥有的五个道具：时光药水、体力药水、铲子、照明弹、手电筒

case 1:

//拳头：假如附近区域有狼（敌人），附近区域的狼消失

//选择人类，消灭在这块区域内的狼人

if(MapCreation.gameChoice == 1)

{

obj = FindClosestObject("wolf", 4.0f);

if(obj != null)

{

Destroy(obj);

}

else

{

useAns = false;

}

}

//选择狼人：

//能用则改变角色属性

if(useAns == true) {

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 180;

//人类

else

score = score + 220;

}

else if (state == 1)

score = score + 200;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 220;

//人类

else

score = score + 180;

}

}

break;

case 2:

//防狼喷雾（×）：改变狼朝你方向行走的概率（和隐身衣功能差不多，可以取消）

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 80;

//人类

else

score = score + 120;

}

else if (state == 1)

score = score + 100;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 120;

//人类

else

score = score + 80;

}

//item\_remainedTime[2] = item\_canUseTime[2];

break;

case 3:

//铲子（√）：玩家附近的障碍消失

obj = FindClosestObject("destroyWall", 1.0f);

if (obj != null)

{

Destroy(obj);

}

else

{

useAns = false;

}

//能用则改变角色属性

if(useAns == true)

{

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 80;

//人类

else

score = score + 120;

}

else if (state == 1)

score = score + 100;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 120;

//人类

else

score = score + 80;

}

}

break;

case 4:

//照明弹（O）：3秒内整个地图亮度调为1

//狼人

if (MapCreation.gameChoice == 0)

{

sd.SetTexture("\_Mask", mask2);

koo = 1;//确定是否使用照明弹

}

//人类

else

{

sd.SetTexture("\_Mask", mask2);

koo = 1;//确定是否使用照明

}

break;

case 5:

//隐身衣：隐身自己，隐身期间敌人碰到你照样也没关系

item\_remainedTime[5] = item\_canUseTime[5];

GetComponent<Renderer>().material.color = new Color(1.0f, 1.0f, 1.0f, 0.5f);

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 80;

//人类

else

score = score + 120;

}

else if (state == 1)

score = score + 100;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 120;

//人类

else

score = score + 80;

}

break;

case 6:

//傀儡：让最近的狼人去追新生成的人类

if(MapCreation.gameChoice == 1)

{

Instantiate(prefabList[2], this.gameObject.transform.position, Quaternion.identity);

//让距离最近的狼人去追傀儡

obj = FindClosestObject("wolf", 100.0f);

obj.GetComponent<Wolf>().traceDummy = true;

}

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 180;

//人类

else

score = score + 220;

}

else if (state == 1)

score = score + 200;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 220;

//人类

else

score = score + 180;

}

break;

case 7:

//（√）时光药水：关卡时间(加减十秒的整数倍，绝对值不超过60)s（已做）

currentTime = currentTime + Random.Range(-6, 7) \* 600;

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 40;

//人类

else

score = score + 60;

}

else if (state == 1)

score = score + 50;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 60;

//人类

else

score = score + 40;

}

break;

case 8:

//（√）体力药水：增加角色体力（已做）

healthPoint = healthPoint + Random.Range(0, max\_healthPoint - healthPoint + 1);

if (state == 0)

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 40;

//人类

else

score = score + 60;

}

else if (state == 1)

score = score + 50;

else

{

//狼人

if (MapCreation.gameChoice == 0)

score = score + 60;

//人类

else

score = score + 40;

}

break;

case 9:

//手电筒（O）：视野范围+1

//狼人

if (MapCreation.gameChoice == 0)

{

sd.SetTexture("\_Mask", mask3);

}

//人类

else

{

sd.SetTexture("\_Mask", mask3);

}

break;

default:

useAns = false;

break;

}

return useAns;

}

//每隔Time.deltatime调用一次

private void FixedUpdate()

{

if (isOver == false)

{

Vector2 temp = Vector2.MoveTowards(transform.position, dest, speed);

GetComponent<Rigidbody2D>().MovePosition(temp);

//在角色能移动的前提下进行移动

if (canMove == true)

{

if ((Vector2)transform.position == dest)

{

GameObject btnObjup = GameObject.Find("upBtn");

Button btnup = btnObjup.GetComponent<Button>();

btnup.onClick.AddListener(Ononeup);

GameObject btnObjdown = GameObject.Find("downBtn");

Button btndown = btnObjdown.GetComponent<Button>();

btndown.onClick.AddListener(Ononedown);

GameObject btnObjright = GameObject.Find("rightBtn");

Button btnright = btnObjright.GetComponent<Button>();

btnright.onClick.AddListener(Ononeright);

GameObject btnObjleft = GameObject.Find("leftBtn");

Button btnleft = btnObjleft.GetComponent<Button>();

btnleft.onClick.AddListener(Ononeleft);

}

else

{

//移动完毕，改变分数和体力等属性状态

changeOnMove();

if (healthPoint <= 0)

{

//选择狼人

if (MapCreation.gameChoice == 0)

{

//狼人体力耗尽，狼人死掉，游戏结束

isOver = true;

youWin = false;

canMove = false;

}

//选择人类

else

{

//人类体力耗尽，不能移动，等着被狼人吃掉

canMove = false;

}

}

}

}

//道具箱按钮的监听事件

GameObject bag1 = GameObject.Find("bag1");

Button button\_bag1 = bag1.GetComponent<Button>();

button\_bag1.onClick.AddListener(bag1Event);

GameObject bag2 = GameObject.Find("bag2");

Button button\_bag2 = bag2.GetComponent<Button>();

button\_bag2.onClick.AddListener(bag2Event);

GameObject bag3 = GameObject.Find("bag3");

Button button\_bag3 = bag3.GetComponent<Button>();

button\_bag3.onClick.AddListener(bag3Event);

GameObject bag4 = GameObject.Find("bag4");

Button button\_bag4 = bag4.GetComponent<Button>();

button\_bag4.onClick.AddListener(bag4Event);

GameObject bag5 = GameObject.Find("bag5");

Button button\_bag5 = bag5.GetComponent<Button>();

button\_bag5.onClick.AddListener(bag5Event);

GameObject bag6 = GameObject.Find("bag6");

Button button\_bag6 = bag6.GetComponent<Button>();

button\_bag6.onClick.AddListener(bag6Event);

GameObject bag7 = GameObject.Find("bag7");

Button button\_bag7 = bag7.GetComponent<Button>();

button\_bag7.onClick.AddListener(bag7Event);

GameObject bag8 = GameObject.Find("bag8");

Button button\_bag8 = bag8.GetComponent<Button>();

button\_bag8.onClick.AddListener(bag8Event);

//showCharacterProperties();

//print(itemNum);

//道具栏状态（改变等级，离敌方距离（(玩家进入区域时)找玩家所属区域的敌方），体力，速度，分数等属性）

//等级

level = (int)Mathf.Floor(score / 2500) + 1;

GameObject.Find("rank\_value").GetComponent<Text>().text = level.ToString();

//距离

GameObject.Find("distance\_value").GetComponent<Text>().text = distance.ToString();

//体力

//print(score);

string healthPoint\_str1 = healthPoint.ToString();

string healthPoint\_str2 = max\_healthPoint.ToString();

string healthPoint\_str = healthPoint\_str1 + " / " + healthPoint\_str2;

GameObject.Find("blood\_value").GetComponent<Text>().text = healthPoint\_str;

//速度

speed = (float)(0.5 + (level - 1) \* 0.05);

GameObject.Find("speed\_value").GetComponent<Text>().text = speed.ToString();

//离（最近的）敌方距离

//选择狼人，获取与最近人类的距离

if(MapCreation.gameChoice == 0)

distance = getNearestDistance("hunter");

//选择人类，获取与最近狼人的距离

else

distance = getNearestDistance("wolf");

if (distance <= 3.5f)

{

this.gameObject.GetComponent<AudioSource>().mute = false;

//this.gameObject.GetComponent<AudioSource>().Play();

}

else

{

this.gameObject.GetComponent<AudioSource>().mute = true;

//this.gameObject.GetComponent<AudioSource>().Pause();

}

GameObject.Find("distance\_value").GetComponent<Text>().text = distance.ToString();

//分数

GameObject.Find("grade\_value").GetComponent<Text>().text = score.ToString();

//剩余时间

currentTime = currentTime - 1;

if (koo == 1) //照明弹计时

{

timer = timer - 1;

}

if (timer <= 0)

{

sd.SetTexture("\_Mask", mask1);

timer = 600f;

koo = 0;

}

int currentTime\_minute = (int)currentTime / 3600;

int currentTime\_second = (int)(currentTime - currentTime\_minute \* 3600) / 60;

string currentTime\_minute\_str = currentTime\_minute.ToString();

string currentTime\_second\_str;

string currentTime\_str;

//秒数<10，前面+0

if (currentTime\_second < 10)

currentTime\_second\_str = "0" + currentTime\_second.ToString();

else

currentTime\_second\_str = currentTime\_second.ToString();

//分钟数<10，前面+0

if (currentTime\_minute < 10)

currentTime\_str = "0" + currentTime\_minute\_str + ":" + currentTime\_second\_str;

else

currentTime\_str = currentTime\_minute\_str + ":" + currentTime\_second\_str;

GameObject.Find("剩余时间").GetComponent<Text>().text = currentTime\_str;

if (currentTime <= 0)

{

//时间结束，你没有被狼人碰到，游戏结束，并且你在这个关卡中获胜

isOver = true;

youWin = true;

}

else

{

if (currentTime % 3600 == 0)

{

state = Random.Range(0, 3);

if (state == 0)

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "宿 夜";

}

else if (state == 1)

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "普通之夜";

}

else

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "月圆之夜";

}

}

}

//装备剩余使用时间（>0的-1）

for (int i = 0; i < 10; i++)

{

if (item\_remainedTime[i] > 0)

{

item\_remainedTime[i] = item\_remainedTime[i] - 1;

if (i == 5 && item\_remainedTime[i] <= 0)

{

GetComponent<Renderer>().material.color = new Color(1.0f, 1.0f, 1.0f, 1.0f);

}

else if(i == 6 && item\_remainedTime[i] <= 0)

{

GameObject obj = GameObject.FindGameObjectWithTag("dummy");

if (obj != null)

{

Destroy(obj);

}

}

}

}

}

else

{

this.gameObject.GetComponent<AudioSource>().Stop();

}

}

private bool Valid(Vector2 dir)

{

Vector2 pos = transform.position;

RaycastHit2D hit = Physics2D.Linecast(pos + dir, pos);

return (hit.collider == GetComponent<Collider2D>());

}

public void showCharacterProperties()

{

//初始化游戏玩家的属性

//角色等级

GameObject.Find("rank\_value").GetComponent<Text>().text = level.ToString();

//游戏分数

GameObject.Find("grade\_value").GetComponent<Text>().text = score.ToString();

//角色速度

GameObject.Find("speed\_value").GetComponent<Text>().text = speed.ToString();

//离地方距离

//distance = 0f;

//角色位置

//dest = Vector2.zero;

//角色体力

GameObject.Find("blood\_value").GetComponent<Text>().text = healthPoint.ToString() + " / " + max\_healthPoint.ToString();

//视野范围

//visibleRange;

//当前状态（state=0(宿夜),state=1(普通之夜),state=2(月圆之夜)）

if (state == 0)

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "宿夜";

}

else if (state == 1)

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "普通之夜";

}

else

{

GameObject.Find("夜晚状态").GetComponent<Text>().text = "月圆之夜";

}

//关卡时间

//levelTime;

//剩余时间

//currentTime;

//装备箱（已有装备，最大容量为8，每个数组存放装备编号）

itemBox = new int[8];

for (int i = 0; i < 8; i++)

{

itemBox[i] = 0;

}

}

//碰撞函数

//这里碰撞函数的设置主要是防止角色碰到碰撞块一直加分并且改变角色属性

void OnCollisionEnter2D(Collision2D collision)

{

canAddScore = false;

}

private void OnCollisionExit2D(Collision2D collision)

{

canAddScore = true;

}

// 开始接触

void OnTriggerEnter2D(Collider2D collider)

{

//玩家碰到了出口（人类碰到了出口），那么游戏胜利，运行终止

if (collider.tag.Equals("exit") == true)

{

isOver = true;

youWin = true;

}

//（玩家选择人类）碰到了狼人

else if (collider.tag.Equals("wolf") == true)

{

//隐身衣道具没有起作用

if (item\_remainedTime[5] <= 0)

{

//直接结束游戏

isOver = true;

//可能要加特效

}

}

//（玩家选择狼人）碰到了人类

else if (collider.tag.Equals("people") == true)

{

score = score + 100;

//销毁人类

Destroy(collider.gameObject);

//增加狼人体力

healthPoint = healthPoint + 10;

//超过最大体力

if (healthPoint > max\_healthPoint)

{

healthPoint = max\_healthPoint;

}

}

//（玩家选择狼人）碰到猎人

else if (collider.tag.Equals("hunter") == true)

{

if(MapCreation.gameChoice == 0)

{

//扣掉80%的体力

int declineHealthPoint = (int)(max\_healthPoint \* 0.8f);

//体力不足，游戏结束，失败

if(healthPoint < declineHealthPoint)

{

healthPoint = 0;

isOver = true;

youWin = false;

string ss = healthPoint.ToString() + " / " + max\_healthPoint.ToString();

GameObject.Find("character\_value").GetComponent<Text>().text = ss;

}

else

{

healthPoint = healthPoint - declineHealthPoint;

}

}

}

//没有碰到人类和狼人（碰到了道具）

else if (collider.tag.Contains("item") == true)

{

//Collider2D clld2 = collider;

//记录，用来标记拿到的道具

int item\_flag = 0;

//根据道具箱的容量判断该装备是否消失，若道具箱满了随机装备不能消失

int can\_destroy = 1;

//寻找道具箱中可以替代的位置

int place\_item = 9;

//寻找道具箱中对应的位置字符串

string item\_flagstr = null;

//在道具箱中寻找，寻找是否有空能够容纳下该道具

int i;

for (i = 0; i < 8; i++)

{

if (itemBox[i] == 0)

{

place\_item = i;

break;

}

}

if (i == 8)

{

can\_destroy = 0;

}

else

{

switch (place\_item)

{

case 0:

item\_flagstr = "1";

break;

case 1:

item\_flagstr = "2";

break;

case 2:

item\_flagstr = "3";

break;

case 3:

item\_flagstr = "4";

break;

case 4:

item\_flagstr = "5";

break;

case 5:

item\_flagstr = "6";

break;

case 6:

item\_flagstr = "7";

break;

case 7:

item\_flagstr = "8";

break;

default: break;

}

}

//可以拾起道具，假如不能拾起道具，不理会它

if (can\_destroy == 1)

{

string str = null;

Sprite s = null;

//Destroy(collider.gameObject);

switch (collider.tag)

{

case "item\_1":

//猎枪

//score = score + 200;

item\_flag = 1;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_2":

//防狼喷雾

//score = score + 100;

item\_flag = 2;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_3":

//铲子

//score = score + 100;

item\_flag = 3;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_4":

//照明弹

//score = score + 100;

item\_flag = 4;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_5":

//隐身

//score = score + 100;

item\_flag = 5;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_6":

//傀儡

//score = score + 200;

item\_flag = 6;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_7":

//时光药水

//score = score + 50;

item\_flag = 7;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

case "item\_8":

//体力药水

//score = score + 50;

item\_flag = 8;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

Destroy(collider.gameObject);

break;

case "item\_9":

//手电筒

//score = score + 50;

item\_flag = 9;

itemBox[place\_item] = item\_flag;

//item\_flagstr = place\_item.ToString();

str = "bag" + item\_flagstr + "\_image";

//在装备箱中放上道具

s = itemList[item\_flag].GetComponent<SpriteRenderer>().sprite;

GameObject.Find(str).GetComponent<Image>().sprite = s;

break;

default:

break;

}

Destroy(collider.gameObject);

}

}

}

/\*

// 接触结束

void OnTriggerExit(Collider collider)

{

Debug.Log("主角接触结束");

print("主角接触结束");

}

// 接触持续中

void OnTriggerStay(Collider collider)

{

Debug.Log("主角接触持续中");

print("主角接触持续中");

}\*/

//寻找游戏地图中离玩家最近的物体（可以是敌人，也可以是墙体）（传入的参数为：附近的障碍--要求的距离）

public GameObject FindClosestObject(string tag, float distance)

{

//查找标签为Enemy的全部游戏物体

GameObject closest = GameObject.FindWithTag(tag);

if (GameObject.FindGameObjectsWithTag(tag) != null)

{

GameObject[] gos;

gos = GameObject.FindGameObjectsWithTag(tag);

//差不多为一个方块大小的距离

var position = transform.position;

//设定一个变量，先暂存distance的数值

float temp = distance;

var curDistance = 0f;

foreach (GameObject go in gos)

{

//计算player与Enemy的向量距离差

var diff = (go.transform.position - position);

//将向量距离平方（防止有负数产生）

curDistance = diff.sqrMagnitude;

if (curDistance < distance)

{

//找出比distance更短的距离

//print(curDistance);

//更新最近距离敌人

closest = go;

//更新最近距离

distance = curDistance;

}

}

//没找到比它短的

if(temp == distance)

{

closest = null;

}

}

else

{

closest = null;

}

return closest;

}

public float getNearestDistance(string tag)

{

GameObject obj = FindClosestObject(tag, Mathf.Infinity);

return Vector2.Distance(this.gameObject.transform.position, obj.gameObject.transform.position);

}

}

（4）游戏菜单操作代码

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class meancrl : MonoBehaviour {

public GameObject pa1;

// Use this for initialization

void Start () {

pa1.SetActive(false);

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClicks);

}

public void OnClicks()

{

pa1.SetActive(true);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class meancrl1 : MonoBehaviour {

public GameObject pa1;

// Use this for initialization

void Start()

{

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClicks);

}

public void OnClicks()

{

pa1.SetActive(false);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class returnmean : MonoBehaviour {

void Start()

{

Button btn = this.GetComponent<Button>();

btn.onClick.AddListener(OnClicks);

}

public void OnClicks()

{

if (MapCreation.gameChoice == 1)

{

SceneManager.LoadScene("people");

}

else if (MapCreation.gameChoice == 0)

{

SceneManager.LoadScene("wolf");

}

}

}

### 2.3游戏效果控制代码

（1）游戏遮罩

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class zhezhaoctrl : MonoBehaviour {

public GameObject player;

public Material sd;

public float le = 1280f;

public float wi = 720f;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

float offsetx = -(player.transform.position.x) / (le \* 2f);

float offsety = -(player.transform.position.y) / (le \* 2f);

sd.SetTextureOffset("Mask", new Vector2(offsetx, offsety));

//GetComponent<Renderer>().material.SetTextureOffset(sd., new Vector2(offsetx, offsety));

}

}

（2）摄像机移动

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class yidong : MonoBehaviour {

//摄像机随着人物移动的脚本按钮

private Vector3 offset;

public Transform player;

void Start()

{

int startx = MapCreation.startx;

int starty = MapCreation.starty;

if (startx == 0 && starty == 0)

this.transform.position = new Vector3(-14f, 10f, -10.0f);

else if (startx == 3 && starty == 0)

this.transform.position = new Vector3(-14f, -12f, -10.0f);

else if (startx == 0 && starty == 3)

this.transform.position = new Vector3(14f, 10f, -10.0f);

else

this.transform.position = new Vector3(14f, -12f, -10.0f);

//player = GameObject.Find("MainCharacter").transform;

offset = player.position - transform.position;

}

void Update()

{

//print(transform.position);

transform.position = Vector3.Lerp(transform.position, player.position - offset, Time.deltaTime \* 5);

//Quaternion rotation = Quaternion.LookRotation(player.position - transform.position);

//transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime \* 3f);

}

}

（3）游戏音效控制

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Audio : MonoBehaviour {

//心跳声音

public AudioClip heartClip;

//用于控制声音的AudioSource组件

private AudioSource heartAudioSource;

float distance;

private void Awake()

{

//添加心跳声音

//在添加此脚本的对象中添加AudioSource组件，此处为摄像机

heartAudioSource = this.gameObject.AddComponent<AudioSource>();

//设置循环播放

heartAudioSource.loop = true;

//设置音量为最大，区间在0-1之间

heartAudioSource.volume = 1.0f;

//设置audioClip

heartAudioSource.clip = heartClip;

//设置静音

heartAudioSource.mute = false;

}

private void FixedUpdate()

{

//print(distance);

distance = MainCharacter.distance;

if (distance <= 3.5f)

{

if (heartAudioSource.isPlaying == false)

heartAudioSource.Play();

}

else

{

if (heartAudioSource.isPlaying == true)

heartAudioSource.Stop();

}

}

}

## 三、模块3（排行榜操作模块）

### 3.1排行榜下载显示代码

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

public class HiScoreApp : MonoBehaviour {

public GUISkin menuSkin;

// 按钮位置

public Rect m\_uploadBut;

public Rect m\_downLoadBut;

// 输入框位置

public Rect m\_nameLablePos;

public Rect m\_scoreLablePos;

public Rect m\_nameTxtField;

public Rect m\_scoreTxtField;

// 滑动框位置

public Rect m\_scrollViewPosition;

public Vector2 m\_scrollPos;

public Rect m\_scrollView;

// 网格位置

public Rect m\_gridPos;

public string[] m\_hiscores;

// 用户名

protected string m\_name="";

// 得分

protected string m\_score = "";

public bool useRedis = false;

// Use this for initialization

private void Start () {

// 列出20个排行榜名单

m\_hiscores = new string[15];

StartCoroutine(DownloadScores(m\_name, m\_score));

}

// 创建UI

private void OnGUI()

{

GUI.skin = menuSkin;

// 设置文字对齐

GUI.skin.button.alignment = TextAnchor.MiddleCenter;

m\_gridPos.width = 500;

m\_gridPos.height = 800;

m\_scrollViewPosition = new Rect(220, 200, m\_gridPos.width, m\_gridPos.height+25);

m\_scrollView = new Rect(0, 0, m\_gridPos.width, m\_gridPos.height);

// 设置文字对齐

GUI.skin.button.alignment = TextAnchor.MiddleLeft;

m\_scrollPos =GUI.BeginScrollView(m\_scrollViewPosition, m\_scrollPos, m\_scrollView, true, true);

m\_gridPos.height = m\_hiscores.Length \* 15;

// 显示分数排行榜

GUI.SelectionGrid(m\_gridPos, 0, m\_hiscores, 1);

GUI.EndScrollView();

}

//下载排行榜

private IEnumerator DownloadScores(string name, string score)

{

WWW www = new WWW("http://127.0.0.1/downloadscores.php");

yield return www;

if (www.error != null)

{

Debug.LogError(www.error);

}

else

{

Debug.Log(www.text);

// 将PHP返回的数据解析为字典格式

var dict = MiniJSON.Json.Deserialize(www.text) as Dictionary<string, object>;

int index = 0;

foreach (object v in dict.Values)

{

UserData user = new UserData();

// 将字典中的值反序列化为UserData

MiniJSON.Json.ToObject(user, v);

// 更新UI上的文字

m\_hiscores[index] = string.Format("{0}\t{1}\t{2}",user.name,user.score,user.time);

m\_hiscores[index+1] = string.Format("\n\n");

index +=2;

}

}

}

// 定义一个类，它的字段名称一定要与服务器返回的JSON格式数据的键名一致

public class UserData

{

public int id;

public string name;

public int score;

public string time;

}

}

### 3.2排行榜更新查看代码

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class downloadlocalscore : MonoBehaviour {

// 用户名

protected string m\_name = "";

// 得分

protected string m\_score = "";

public GameObject grade;

public GUISkin menuSkin;

// Use this for initialization

void Start () {

}

private void OnGUI()

{

GUI.skin = menuSkin;

StartCoroutine(downloadScore(m\_score));

GUI.Label(new Rect(245, 530, 800, 200), m\_score);

}

private IEnumerator downloadScore(string score)

{

WWW www = new WWW("http://127.0.0.1/downloadlocalscore.php");

yield return www;

if (www.error != null)

{

Debug.LogError(www.error);

}

else

{

Debug.Log(www.text);

var dict = MiniJSON.Json.Deserialize(www.text) as Dictionary<string, object>;

foreach (object v in dict.Values)

{

UserData user = new UserData();

// 将字典中的值反序列化为UserData

MiniJSON.Json.ToObject(user, v);

m\_score=string.Format("{0}\t{1}\t{2}",user.name,user.score, user.time);

}

}

}

// Update is called once per frame

void Update () {

}

public class UserData

{

public string name;

public int score;

public string time;

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class uploadloaclscore : MonoBehaviour {

MainCharacter a = new MainCharacter();

// Use this for initialization

void Start () {

GameObject btnObj = GameObject.Find("queding");

Button btn = btnObj.GetComponent<Button>();

btn.onClick.AddListener(onchick);

}

public void onchick()

{

int score = a.getscore();

StartCoroutine(UploadScore(score));

}

private IEnumerator UploadScore(int score)

{

WWWForm form = new WWWForm();

form.AddField("score", score);

WWW www = new WWW("http://127.0.0.1/uploadlocalscore.php", form);

yield return www;

if (www.error != null)

{

Debug.LogError(www.error);

}

else

Debug.Log(www.text);

}

// Update is called once per frame

void Update () {

}

public class UserData

{

public int id;

public string name;

}

}