GVR for Unity 实例

1、Unity 界面介绍

在使用 Unity 开发项目时，为了设置对象的不同属性，经常需要在多个窗口跳转，对于初学者，有必要知道每个窗口的具体作用(详细介绍，请参看官方文档)，这里为了便于后续文档的描述，用 A、B、C、D、E 五个区域来简化原来窗口的名称，如下图所示：

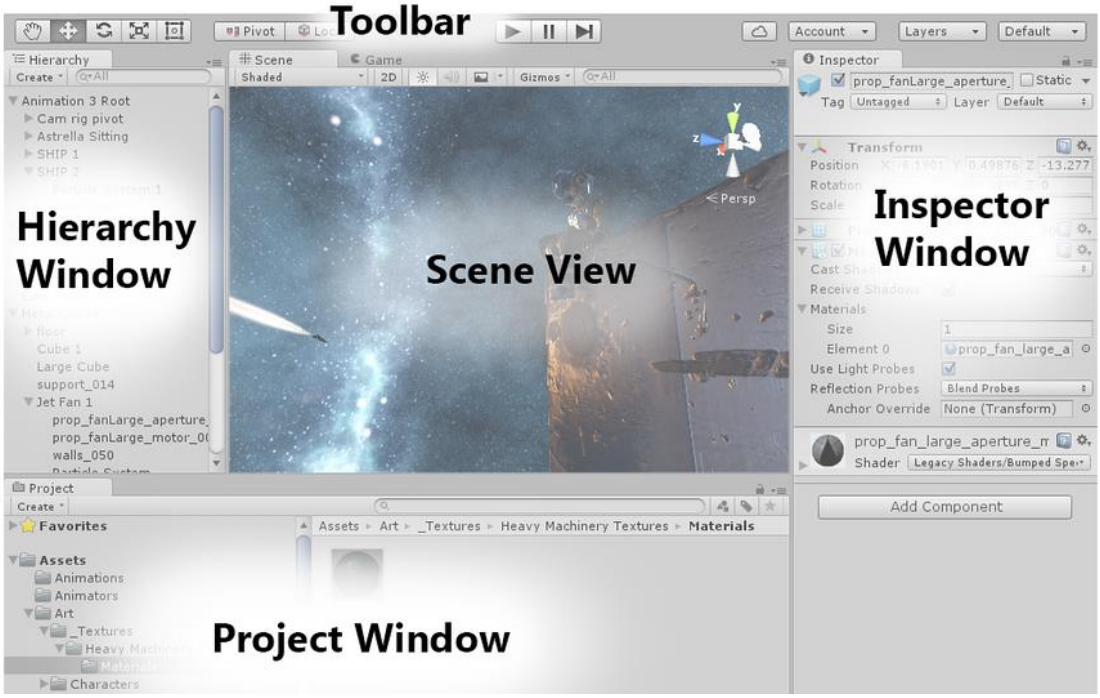


图 1　Unity 主界面介绍

A-Hierarchy； B-Toolbar；C-Scene；D-Inspector；E-Project Window

2、gvr-unity-sdk 的环境搭建

- 安装 Unity —— Version 5.4.1f1 personal
- gvr-unity-sdk —— https://github.com/googlevr/gvr-unity-sdk
- 打开 Unity->新建->命名->选择 3D 场景->Create，在主界面菜单栏中，通过 Assets->Import Package->Custom Package，导入解压后的 gvr-unity-sdk
- 在 E 区 Assets 目录下，新建“_Scene”的空文件夹，保存当前场景到“_Scene”目录下，File->Save Scene，重命名为“MainUI”

3、为“MainUI”添加 VR 摄像头

- 删除场景“MainUI”中的默认摄像头，在 A 区，选中 MainCamera，右键“Delete”
- 添加 VR 摄像头，在 E 区，展开 Assets->GoogleVR->Legacy->Prefabs->GvrMain，选中 GvrMain 并拖动到 A 区空白位置，并在 D 区设置其位置为{x:20,y:2,z:-35}
- 添加 VR 摄像头焦点，在 E 区，展开 Assets->GoogleVR->Prefabs->UI->GvrReticle，拖动“GvrReticle”到 A 区的 GvrMain->Head->MainCamera 对象下
- 设置可获得焦点的对象类型，在 A 区依次展开 GvrMain->Head->Main Camera，选中 Main Camera，在 D 区，单击“Add Component”按钮，添加 Physics Raycaster
- 设置对象获得焦点的方式，单击 A 区空白区域，右键选择 UI->Event System，在 D

区，先右键删除 StandralInputMethod，再通过“Add Component”选项，添加 GazeInputMethod

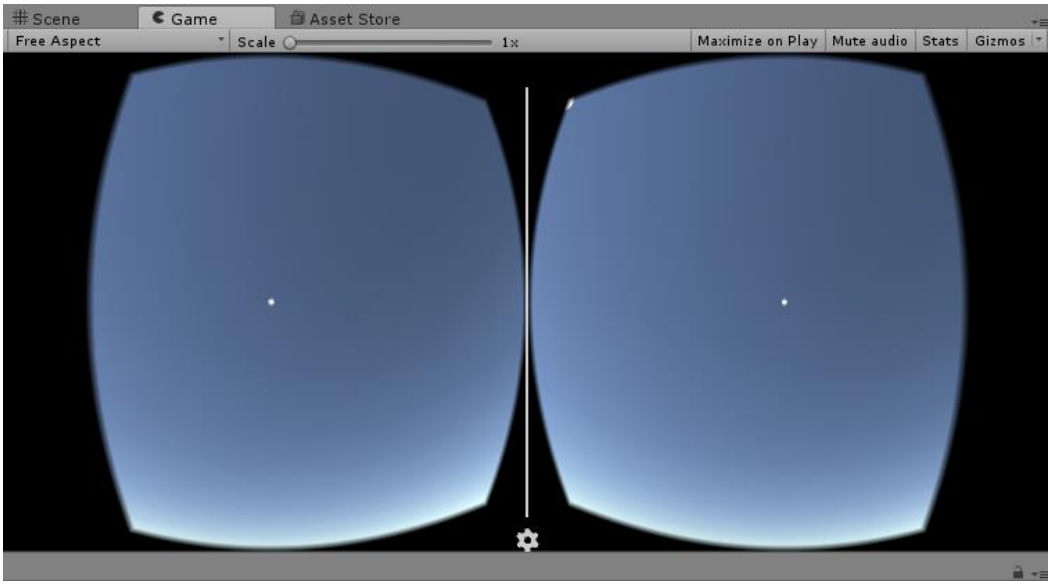


图 2 MainUI 运行预览图

4、创建 Menu 选择界面

- 添加 Menu 界面的整体框架，单击 A 区空白区域，右键添加 UI->Canvas，重命名为“Menu”，在 D 区设置其相关基本参数，如图 3 所示

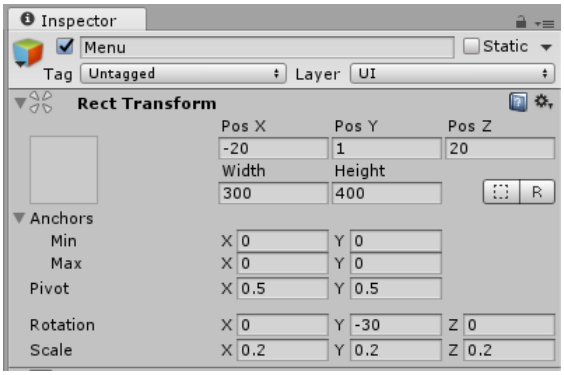


图 3 “Menu”相关参数

- 在 A 区右键“Menu”，通过 UI->Panel，添加两个 Menu 子界面，分别命名为“Item1”、“Item2”，基本参数设置如图 4 所示

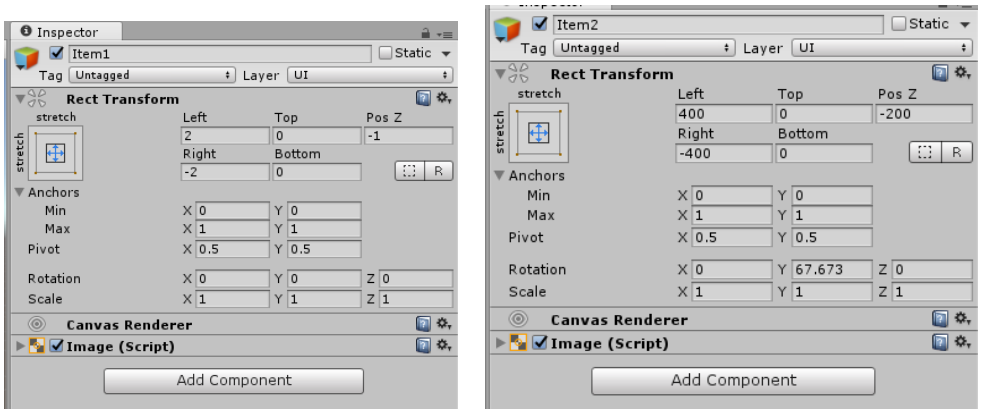


图 4 Item1、Item2 相关参数设置

- 在 A 区右键“Item1”，通过 UI->Button，为 Item1 添加两个按钮事件，分别命名为“ClickEvent”和“GazeEvent”，基本参数设置如图 5 所示

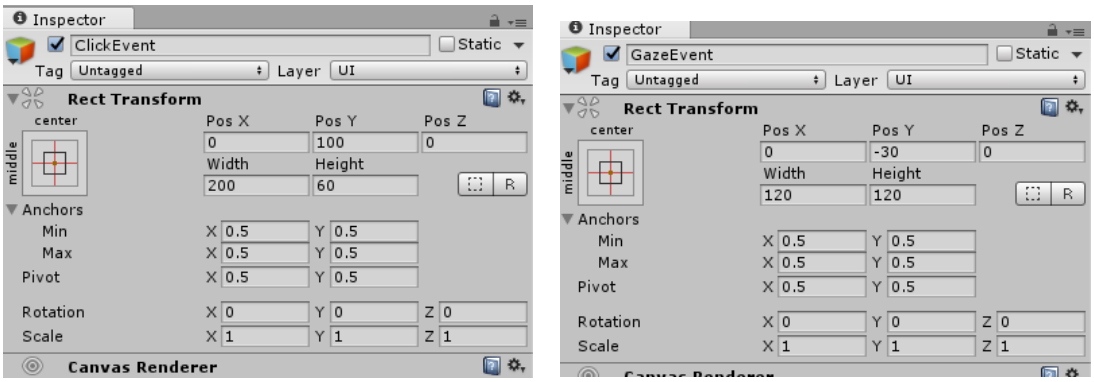


图 5 ClickEvent、GazeEvent 相关参数

- 分别设置 ClickEvent->Text 和 GazeEvent->Text 的基本参数，如图 6 所示

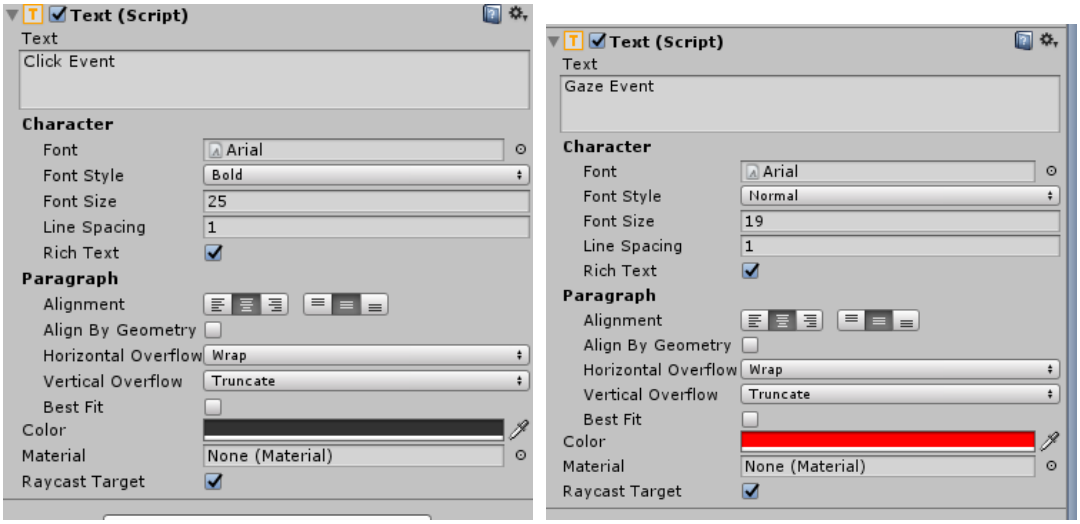


图 6 Text 相关参数

- 在 A 区单击“ClickEvent”，并在 D 区点击“Add Component”按钮，添加“New Script”文件，重命名为“ChangeText”，双击打开，并复制如下代码

```
using UnityEngine;
using UnityEngine.UI;
namespace VRStandardAssets.Utils

{
    public class ChangeText : MonoBehaviour
    {

        [SerializeField]
        private Text buttonText;

        public void Change()
        {
            if (buttonText.text.Equals("Change"))
                buttonText.text = "Click Event";
```

```
            else
                buttonText.text = "Change";
        }
    }
}
```

- 保存“ChangeText”文件，在 Unity 界面中，刷新“ChangeText”，将 A 区的“ClickEvent->Text”拖动到 D 区“ChangeText”下的 Button Text 栏
- 在 A 区右键“GazeEvent”，添加 UI->Image，基本参数设置如图 7 所示

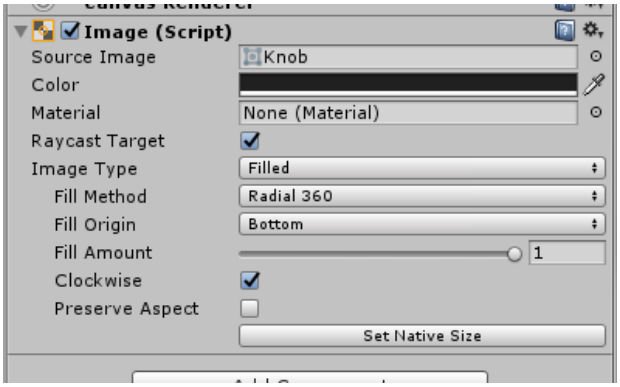


图 7 Image 相关参数

- 在 A 区单击“GazeEvent”，并在 D 区通过“Add Component”按钮添加名为“Button3Function”的“New Script”文件，复制以下代码

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class Button3Function : MonoBehaviour, IPointerEnterHandler,
IPointerExitHandler{

    // Add an image as child to your button object and set its image type to Filled.
    // Assign it to this field in inspector.
    public Image progressImage;

    bool isEntered = false;

    float timeElapsed = 0;
    float GazeActivationTime = 3;

    // Update is called once per frame
    void Update () {

        if (isEntered)
        {
            timeElapsed += Time.deltaTime;
```

```
            progressImage.fillAmount = Mathf.Clamp(timeElapsed / GazeActivationTime,
0, 1);
            if (timeElapsed >= GazeActivationTime)
            {
                timeElapsed = 0;
                isEntered = false;
            }
        }
        else
        {
            timeElapsed = 0;
        }
    }

    public void OnPointerEnter(PointerEventData eventData)
    {
        isEntered = true;
        progressImage.fillAmount = 0;
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        isEntered = false;
        progressImage.fillAmount = 100;
    }
}
```

- 保存“Button3Function”文件，在 Unity 界面中，刷新 Button3Function，将 A 区的“GazeEvent->Image”拖动到“Button3Function”下的 Progress Image 栏
- 在 A 区右键“Item2”，通过 UI->Button，创建名为“NextScene”的按钮，基本参数设置如图 8 所示

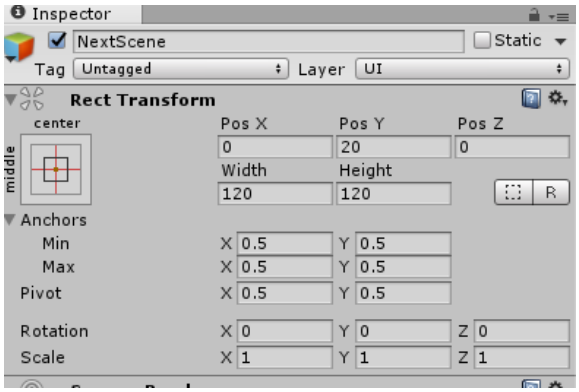


图 8 “NextScene”基本参数

- 在 A 区单击“NextScene”，并在 D 区通过“Add Component”添加名为“LoadScene”的“New Script”文件，复制如下代码

```
using UnityEngine;
using UnityEditor.SceneManagement;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using System;

public class LoadScene : MonoBehaviour,IPointerEnterHandler,IPointerExitHandler {

    public Text timeText;
    public string sceneName;

    private bool isEnter = false;
    private float time = 0;

    public void OnPointerEnter(PointerEventData eventData)
    {
        isEnter = true;
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        timeText.text = "LoadScene";
        isEnter = false;
        time = 0;
    }

     // Update is called once per frame
     void Update () {

        if (isEnter)
        {
            time += Time.deltaTime;
            if (0 <= time && time < 1)
                timeText.text = "5s";
            else if (1 <= time && time < 2)
                timeText.text = "4s";
            else if (2 <= time && time < 3)
                timeText.text = "3s";
            else if (3 <= time && time < 4)
                timeText.text = "2s";
            else if (4 <= time && time < 5)
                timeText.text = "1s";
            else
            {
```

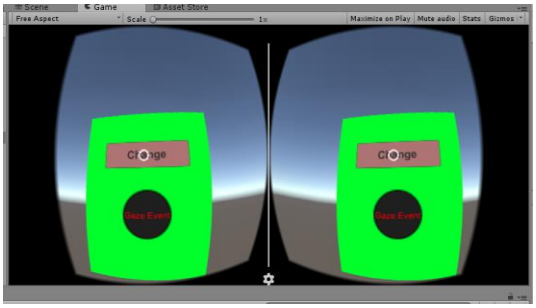
```
                timeText.text = "LoadScene";
                EditorSceneManager.LoadScene(sceneName);
                isEnter = false;
                time = 0;
            }
        }

    }
}
```

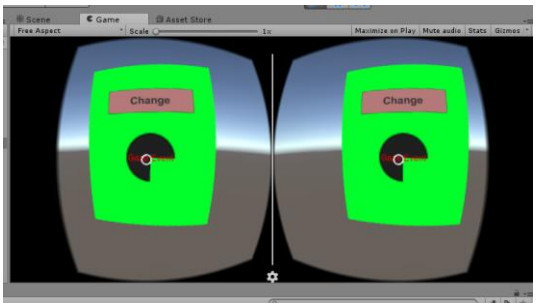
- 保存“LoadScene”文件，在 Unity 界面中，刷新 LoadScene，将 A 区的 NextScene->Text 拖动到“LoadScene”下的 Time Text 栏，在 Scene Name 栏输入“LoadScene”，保存“MainUI”
- 通过 File->New Scene 新建名为“LoadScene”的场景文件，保存在“_Scene”目录下，并为“LoadScene”添加 VR 摄像头(参考步骤 3)
- 返回“MainUI”，运行程序，测试各个按钮功能

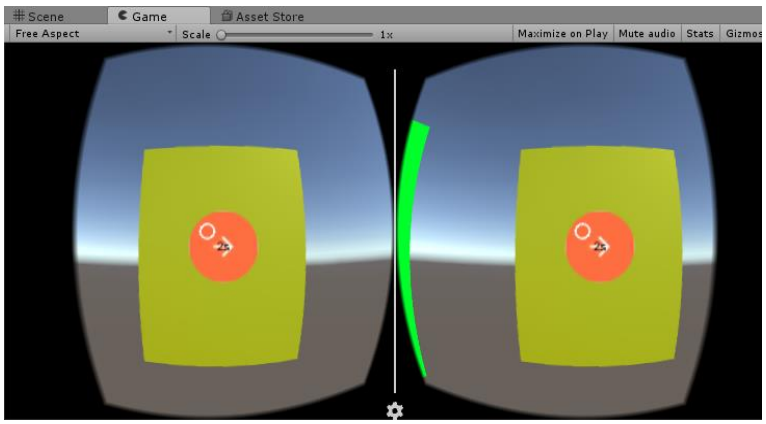
5、测试结果

如要转动视角，需按住“Alt”按钮并同事转动鼠标

- 当焦点处于“ClickEvent”按钮时，此时点击鼠标左键，“ClickEvent”按钮提示内容为“Change”，再次点击返回初始状态
- 当焦点处于“GazeEvent”按钮时，设置在“GazeEvent”按钮上的图片自动顺时针在 3s 内填满
- 当焦点处于“NextScene”按钮时，倒计时开始，当倒计时为 0 时，跳转到 LoadScene 界面







效果图