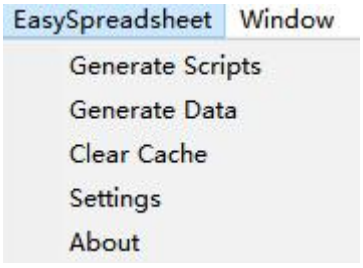
目录

1 简介

游戏开发中，我们经常使用 xlsx 文件来编辑大量的游戏常量数据，例如任务、装备配置，需要使用工具把 xlsx 数据转化成便于游戏运行时使用的数据。
EasySpreadsheet 就是一个帮助你在 Unity 游戏中方便地使用 xlsx 数据的插件，它用法简单、特性丰富、设置灵活，并且运行时不依赖第三方库。
它在 Unity 编辑器中解析 xlsx 文件，然后生成代码和数据，给游戏运行时使用。
它为 excel 文件中每个 sheet 生成一份 ScriptableObject 代码和一份数据文件。
当游戏运行时，EasySpreadsheet 载入生成的数据文件，提供查询数据的 API。

安装之后，它的菜单会出现在 Unity 编辑器菜单栏里：



点击生成代码时，会为 xlsx 文件中的每个 sheet 生成一个 ScriptableObject 类型的代码文件。
点击生成数据时，即为 xlsx 文件中的每个 sheet 生成一个 ScriptableObject 数据文件。

特性：

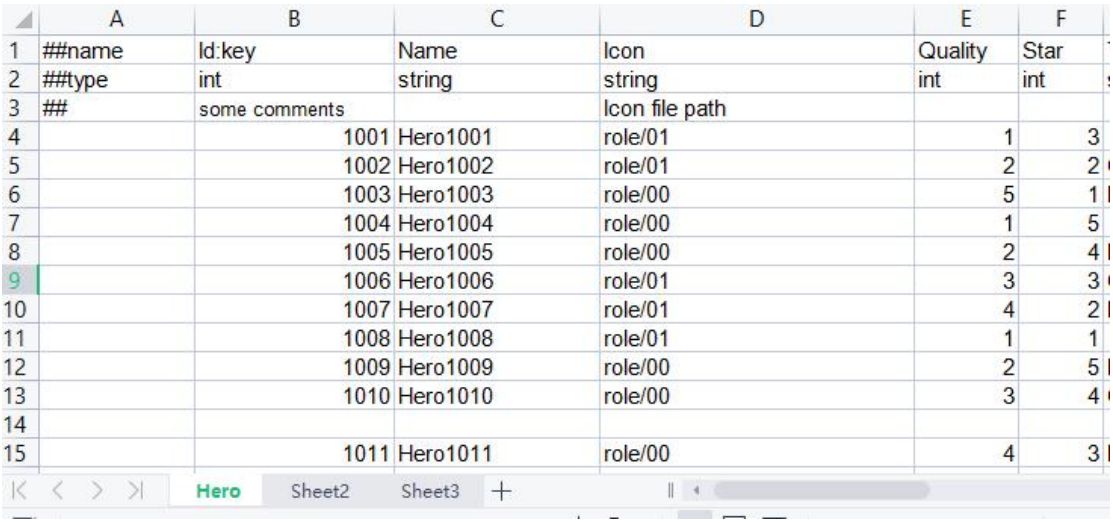
- 一键快速生成代码和数据
- 支持所有 Unity 版本，跨平台
- 支持 Visual Scripting
- 支持类型 int、long、float、string、bool、List、Dictionary
- 支持引擎类型 Vector2、Vector3、Rect...
- 支持自定义类型
- 灵活的设置
- 丰富的示例

2 文件结构

（1）Xlsx 文件

用 Hero.xlsx 作为例子，它的截图如下。
其中第一页 Hero 页的前三行被特殊标记，
第一行是变量名（##name 标记），
第二行是变量类型（##type 标记），
第三行是注释（##标记），
第四行及以下是数据。

因此，Hero 页符合约定，会被导入，生成代码（Hero.cs 和 HeroTable.cs）和数据（Hero.asset）
而另外两页 Sheet2、Sheet3 前三行没有标记，所以会被忽略。
注意：数据页名不必和文件名相同，且可包含多个数据页。



	A	B	C	D	E	F
1	##name	Id:key	Name	Icon	Quality	Star
2	##type	int	string	string	int	int
3	##	some comments		Icon file path		
4		1001	Hero1001	role/01	1	3
5		1002	Hero1002	role/01	2	2
6		1003	Hero1003	role/00	5	1
7		1004	Hero1004	role/00	1	5
8		1005	Hero1005	role/00	2	4
9		1006	Hero1006	role/01	3	3
10		1007	Hero1007	role/01	4	2
11		1008	Hero1008	role/01	1	1
12		1009	Hero1009	role/00	2	5
13		1010	Hero1010	role/00	3	4
14						
15		1011	Hero1011	role/00	4	3

Hero Sheet2 Sheet3

（2）生成的代码文件

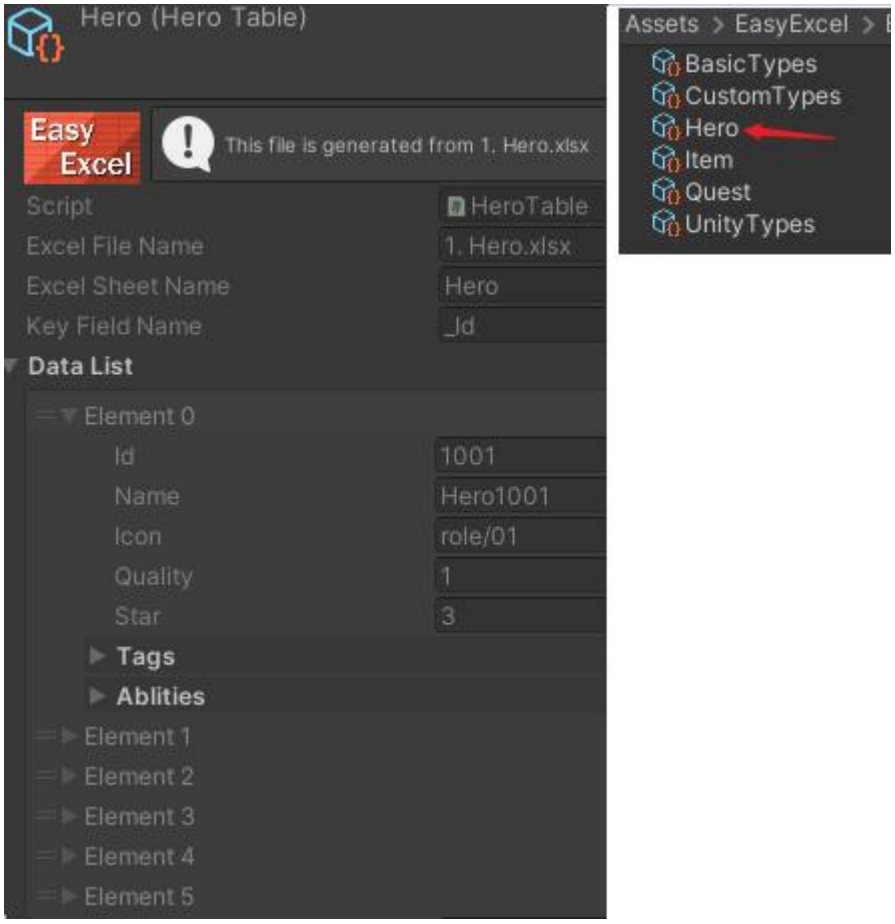
因为 Hero.xlsx 中只有 Hero 页有数据，所以只生成了 Hero.cs 和 HeroTable.cs。
Hero 类表示 Hero 页中的一行数据。
HeroTable 是 Hero 数据合集，它包含所有从 Hero 页导入的数据。
HeroTable 是 ScriptableObject 子类。

```
[Serializable]
public sealed partial class Hero : EERowData
{
    [EEKeyField]
    [SerializeField]
    private int _Id;
    public int Id => _Id;

    [SerializeField]
    private string _Name;
    public string Name => _Name;

    [SerializeField]
    private string _Icon;
    public string Icon => _Icon;

    [SerializeField]
    private int _Quality;
    public int Quality => _Quality;

    [SerializeField]
    private int _Star;
    public int Star => _Star;
```

```
public sealed partial class HeroTable : EERowDataTable
{
    [SerializeField]
    private List<Hero> _dataList = new List<Hero>();

    private Dictionary<int, Hero> _dataMap;
    public Dictionary<int, Hero> DataMap => _dataMap;
    public List<Hero> DataList => _dataList;

    public Hero Get(int key) => _dataMap.TryGetValue(ke
    public Hero this[int key] => _dataMap.TryGetValue(k

    protected override void OnDeserialized()
    {
        _dataMap = new Dictionary<int, Hero>(_dataList.
        foreach (var data in _dataList)
        {
            _dataMap.Add(data.Id, data);
            (data as ISerializationCallbackReceiver).On
        }
        PostInit();
    }
```

（3）生成的数据文件

生成了 Hero.asset 文件，它是 HeroTable 类型的 ScriptableObject 文件，包含了所有 Hero 数据。



（4）生成的 DataTables 类

每次生成代码时，还会生成一个数据管理类 DataTables.cs。
它负责加载数据和查询数据。
DataTables 中包含同步和异步两种 Load 方法
DataTables 中为每种数据都生成了静态方法，来获取数据，如：
DataTables.GetHero(1001)
DataTables.GetHeroList()
注意，在调用这些静态方法前，一定要先完成初始化加载数据。

```
public partial class DataTables
{
    public static DataTables Instance => s_instance;

    private Dictionary<Type, EERowDataTable> _cfgs;
    public Dictionary<Type, EERowDataTable> Cfgs => _
    public BasicTypesTable BasicTypesTable {get; priv
    public CustomTypesTable CustomTypesTable {get; pr
    public HeroTable HeroTable {get; private set;}
    public ItemTable ItemTable {get; private set;}
    public QuestTable QuestTable {get; private set;}
    public UnityTypesTable UnityTypesTable {get; priv


    public void Load(IEEDataLoader loader)
    {

    public IEnumerator LoadAsync(IEEDataLoader loader
    {

    public static Hero GetHero(int id)
    {
        return s_instance.HeroTable.Get(id);
    }

    public static List<Hero> GetHeroList()
    {
        return s_instance.HeroTable.DataList;
    }
```

3 运行时 API

（1）初始化加载数据

创建 DataTables 实例，并调用 Load 完成初始化。

```
public class InitializeEasyExcel : MonoBehaviour
{
    private readonly EE.DataTables _dataTables = new EE.DataTables();

    private void Awake()
    {
        // Load all data when app starts.
        _dataTables.Load(new EasyExcel.EEDataLoaderResources());
    }
}
```

（2）获取数据

Hero 示例，取得 id 为 1001 的 hero 配置

```
// Get hero with id 1001.
EE.Hero cfg1 = EE.DataTables.GetHero(1001);

Debug.Log("Hero's name is " + cfg1.Name);
```

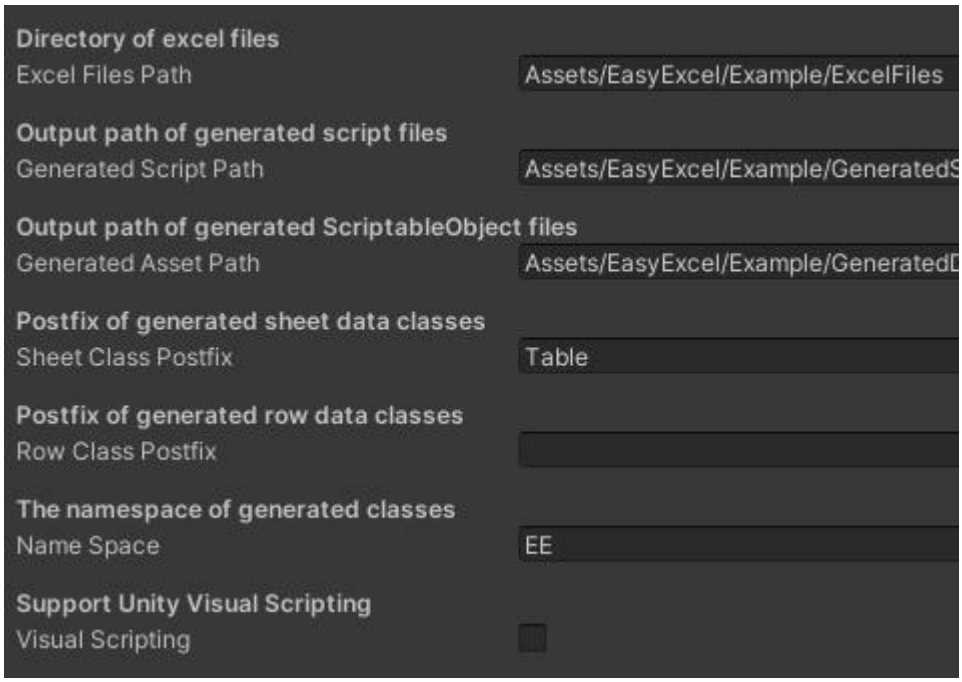
取得全部 hero 配置

```
// Get all heros'data.
List<EE.Hero> heros = EE.DataTables.GetHeroList();
```

4 设置

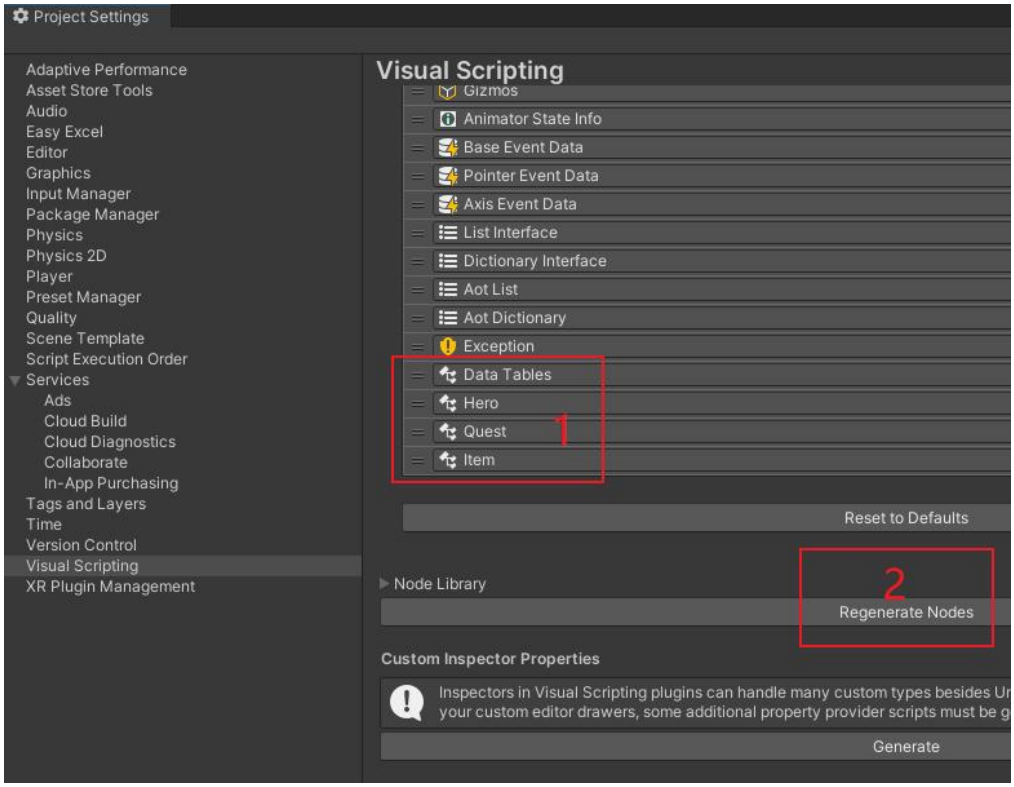
（1）界面

点击 EasySpreadsheet/Settings 可打开界面。
这里可以设置输入和输出路径，类名规则，Unity 的 Visual Scripting。



（2）如何开启 Visual Scripting

勾选界面中的 Visual Scripting
点击 Generate Scripts 重新生成代码
在 ProjectSettings/VisualScripting，把生成的类型加入到 TypeOptions 中
点击 Regenerate Nodes



在 Example/Scene/VisualScripting 中可以查看示例，
其中示范了调用静态

5 示例

（1）预定义类型 PredefinedTypes.xlsx

_PredefinedTypes.xlsx 是一个固定存在的文件，用它来添加自定义类型。
注意不可以删掉它。

A	B	C	D	E	F
##var	full_name	separator	comment	fields	
##var				name	type
##	全名(包含模块和名字)	分割符		字段名	类型
	UnityEngine.Vector2	-		x	float
				y	float
	UnityEngine.Vector3	-		x	float
				y	float
				z	float
	UnityEngine.Rect	-		x	float
				y	float
				width	float
				height	float
	Int3	-		x	int
				y	int
				z	int
	Int2	-		x	int
				y	int
	Int4	-		x	int

（2）基础类型 BasicTypes.xlsx

BasicTypes.xlsx 中的 BasicTypes 页，演示了支持的基础类型：
int、long、float、string、bool、List、Dictionary。
对于 List、Dictionary，你可以在类型后写上(sep=;)来指定元素分割符，
如(sep=;)、(sep=#)、(sep=,)。
如果省略(sep=;)，则使用默认分隔符 ‘;’。

D	E	F	G	H	I	J	K
num1	num2	num3	bool1	nums	strs	nums2	dict1
int	float	long	bool	int[](sep=;)	string[](sep=;)	float[](sep=;)	< int , int >
20	3.1	1	TRUE	1;2;3;4;5	a123;b456;c78	3.1;3.2;3.3	1 : 11 ; 2 : 22; 3: 33 ;
80	4.2	2	FALSE	1;2;3;4	eeee	4.2;4.3	1:11;2:22;3:33;
100	5	3	TRUE	1;2;3	a;bbbb	5	1:11;2:22
1000	50.8	4	FALSE	1;2	cccc	50.8	1:11
120	25	5	TRUE	1	ddddd	25	
160		6	FALSE				
200		7	TRUE				

（3）自定义类型 CustomTypes.xlsx

CustomTypes.xlsx 中的 CustomTypes 页，演示了如何使用自定义类型，这些类型需要首先在 PredefinedTypes.xlsx 中定义。

C	D	E	F
point2d	point3d	point3ds	point2dDict
Int2	Int3	Int3[]	<int,Int2>
5-6	5-6-7	5-6-7;2-3-4	111:5-6
3-9	2-3-4	2-3-4;	222:3-9
1-2	3-4-5		333:1-2
4-7	4-5	4-5-6	444:4-7
			222:3-9 ; 333:1-2 ; 501:6-7

（4）Unity 引擎类型 UnityEngineTypes.xlsx

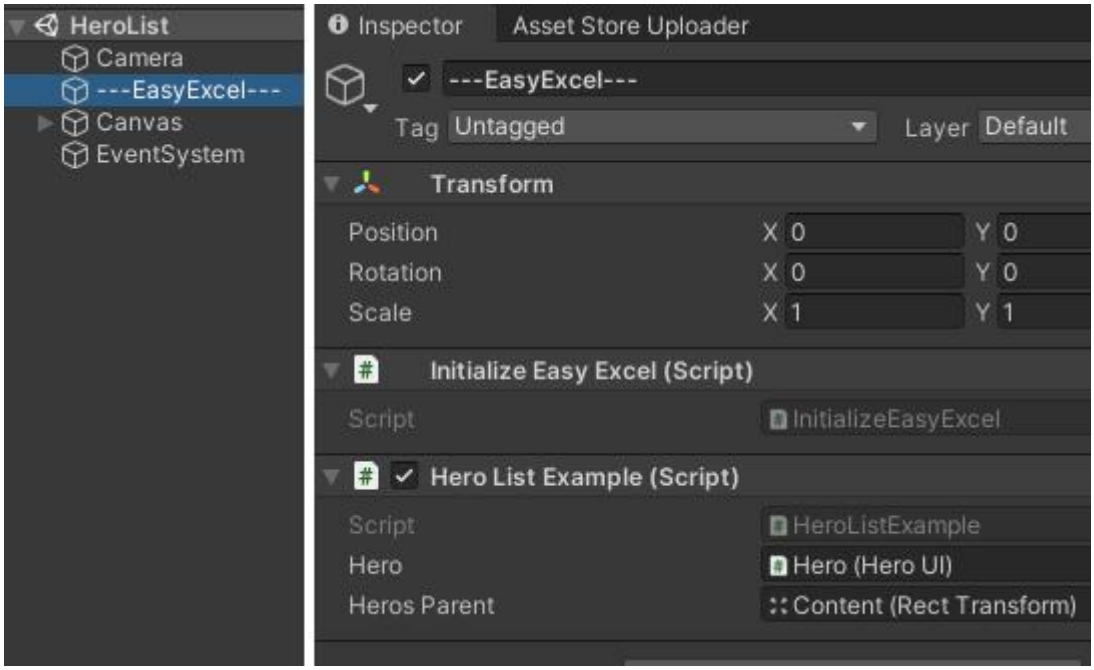
UnityEngineTypes.xlsx 中的 UnityEngineTypes 页，演示了如何使用 Unity 内置类型，这些类型需要首先在 PredefinedTypes.xlsx 中定义。

C	D	E	F
vec2	vec3	vec3s	vec3Dict
UnityEngine.Vector2	UnityEngine.Vector3	UnityEngine.Vector3[]	<int,UnityEngine.Vector3>
5.1-6.2	5.1-6.2-7.9	5.1-6.2-7.9 ; 2-3-4	111 : 5.1-6.2-7.9 ; 402 : 2-3-4
1.3-2.5	3.2-4.2-5.7		
4.3-7.1	4-5	4-5-6	402 : 2-3-4

（5）英雄、任务、物品示例

Hero.xlsx 中的 Hero 页，演示了如何配置英雄角色数据，

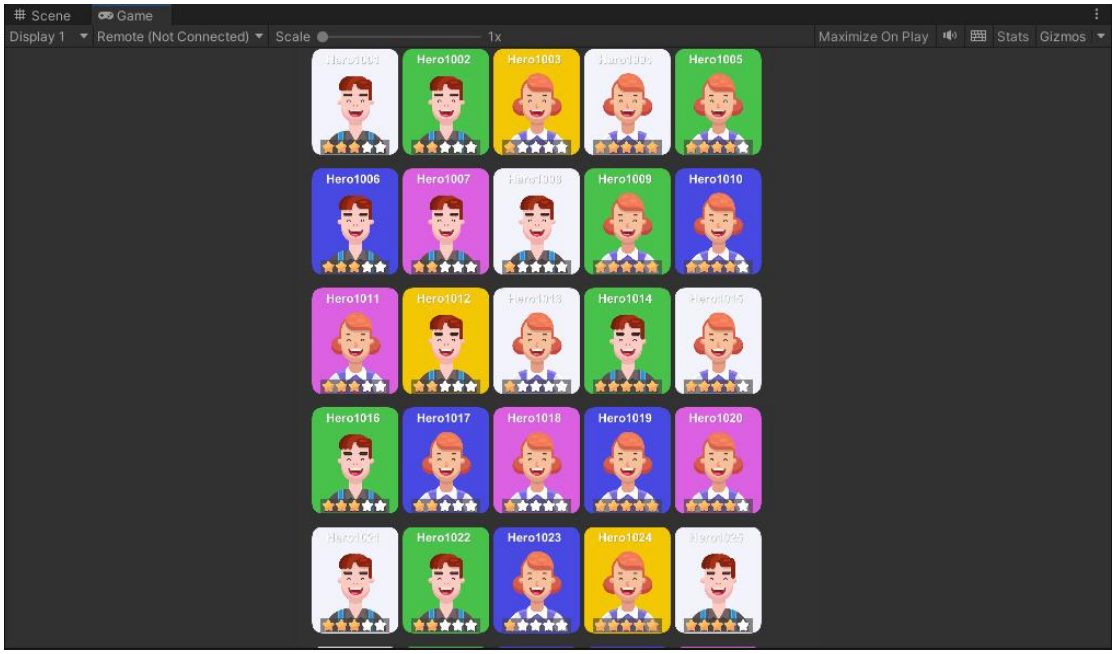
查看场景文件 EasySpreadsheet\Example\Scenes\HeroList.unity



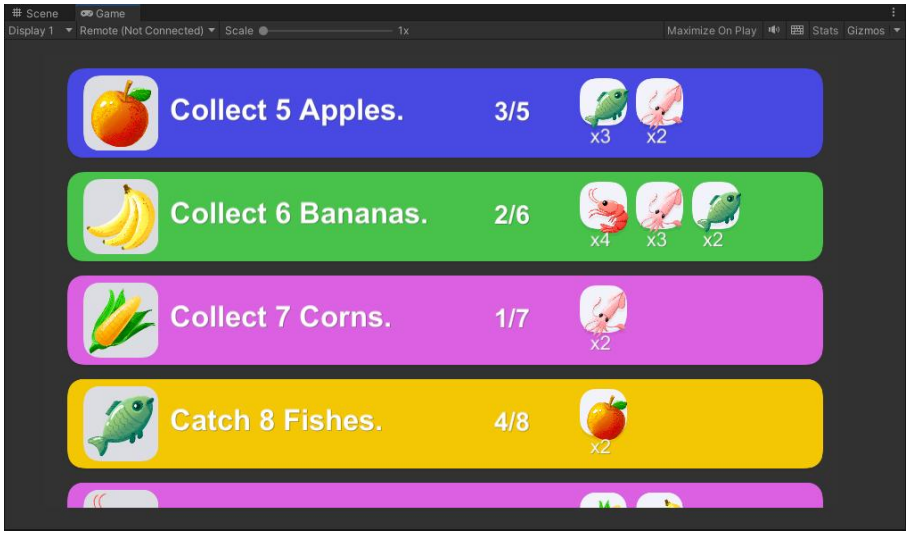
InitializeEasySpreadsheet.cs 脚本示例如何初始化和加载数据。

HeroListExample.cs 脚本示例获取 hero 列表并显示出来。

运行时截图如下：



Quest.xlsx 中的 Quest 页，演示了如何配置任务数据，Item.xlsx 中的 Item 页，演示了如何配置道具。其中 Quest 引用了 Item 表，查看场景文件 EasySpreadsheet\Example\Scenes\QuestList.unity。



（6）可视化编程

场景文件 EasySpreadsheet\Example\Scenes\VisualScripting.unity
参照设置（2）中开启 Visual Scripting 选项。

6 技术支持

QQ：1534921818

Mail：1534921818@qq.com