Demo10-有限状态机的使用

2018年4月27日 20:50

最近有朋友需要Game Framework状态机的使用教程，所以我就简单说说状态机的用法吧。

本Demo只介绍Game Framework框架状态机的使用方式，至于什么是有限状态机，以及状态机的最佳使用方式，大家自行学习。

1.FsmState（有限状态机基类）

状态机包括角色的各个状态，只要让我们的状态类继承FsmState即可。
在继承FsmState的时候需要指定一个类型，这个类型就是状态的持有者。比如我们有一个英雄，英雄有行走和站立的状态，那么，这些状态的持有者就是这个英雄。

本Demo演示一个英雄的行走和站立状态切换，其中英雄是GameFramework中的Entity，如果大家还记得Demo6的话，我们创建实体的时候需要一个实体逻辑处理类，这个逻辑处理类就可以作为状态的持有者。

我们来看代码：

```
public class Demo10_ProcedureLaunch : ProcedureBase {
    protected override void OnEnter(ProcedureOwner procedureOwner)
    {
        base.OnEnter(procedureOwner);
        // 获取框架实体组件
        EntityComponent entityComponent
            = UnityGameFramework.Runtime.GameEntry.GetComponent<EntityComponent>();
        // 创建实体
        entityComponent.ShowEntity<Demo10_HeroLogic>(1, "Assets/Demo10/CubeEntity.prefab", "EntityGroup");
    }
}
```

在初始场景里加载一个实体，实体的预制体大家自己去源码下载，我就不演示怎么创建这个预制体了。
其中Demo10_HeroLogic是这个实体的逻辑处理类。

于是，站立状态类和行走状态类要这么创建：

```
public class Demo10_HeroIdleState : FsmState<Demo10_HeroLogic>
public class Demo10_HeroWalkState : FsmState<Demo10_HeroLogic>
```

和之前说的一样，状态类继承了FsmState，并且需要指定状态的持有者。

2.站立状态类（HeroIdleState）

我们来看看站立状态类的具体实现：

```
public class Demo10_HeroIdleState : FsmState<Demo10_HeroLogic> {
    protected override void OnInit (IFsm<Demo10_HeroLogic> fsm) { }

    protected override void OnEnter (IFsm<Demo10_HeroLogic> fsm) {
        Log.Info("进入站立状态");
    }

    protected override void OnUpdate (IFsm<Demo10_HeroLogic> fsm, float elapseSeconds, float
    realElapseSeconds) {
        /* 按W、S或者上下方向键移动 */
        float inputVertical = Input.GetAxis ("Vertical");
```

```
        if (inputVertical != 0) {
            /* 移动 */
            ChangeState<Demo10_HeroWalkState>(fsm);
        }
    }

    protected override void OnLeave (IFsm<Demo10_HeroLogic> fsm, bool isShutdown) {
    }

    protected override void OnDestroy (IFsm<Demo10_HeroLogic> fsm) {
        base.OnDestroy (fsm);
    }
}
```

是不是角色状态类似曾相识？（旁白：并没有）
没错，你们猜对了，这和流程的生命周期简直一模一样。（旁白：不是的，我压根没有猜过）

因为...流程就是状态类！流程就是状态类！流程就是，状态类。

```
namespace GameFramework.Procedure
{
    /// <summary>
    /// 流程基类。
    /// </summary>
    44 references | You, 15 days ago | 1 author (You)
    public abstract class ProcedureBase : FsmState<IProcedureManager>
```

所以，大家应该和容易理解状态类的用法。
如果我没有瞎猜错的话，对于状态类的各个生命周期，我们用的最多的会是OnEnter（进入状态时调用）和OnUpdate（轮询）。
比如，在进入站立状态时，切换到站立动画，在OnUpdate中判断是否切换到其它状态。这个大家根据实际情况使用。

Demo里的站立状态会判断是否按下了移动键，如果是，则切换到行走状态。

3.行走状态类（HeroWalkState）

至于行走状态，基本上和站立状态一样，只是OnUpdate稍有不同：

```
/// <summary>
/// 有限状态机状态轮询时调用。
/// </summary>
/// <param name="fsm">有限状态机引用。</param>
/// <param name="elapseSeconds">逻辑流逝时间，以秒为单位。</param>
/// <param name="realElapseSeconds">真实流逝时间，以秒为单位。</param>
6 references
protected override void OnUpdate (IFsm<Demo10_HeroLogic> fsm, float elapseSeconds, float realElapseSeconds) {
    float inputVertical = Input.GetAxis ("Vertical");
    if (inputVertical != 0) {
        /* 移动 */
        fsm.Owner.Forward(elapseSeconds * inputVertical);
    } else {
        /* 站立 */
        ChangeState<Demo10_HeroIdleState>(fsm);
    }
}
```

如果按下了移动键，则调用状态持有者的Forward函数进行移动（这是自己写的函数，大家看源码即可）。
如果没有按下移动键，则切换回站立状态。

于是，在按下移动键和松开移动键的时候，英雄就会在行走和站立之间不断切换。

4.Owner（持有者）

状态类的每一个生命周期函数都会有一个fsm对象，通过fsm.Owner可以获得状态机持有者对象，这里的Owner就是我们的Demo10_HeroLogic类了。

因此，在各个状态里，可以对持有者进行操作。

5.让状态机启动

状态类有了，持有者有了，但是，它们怎么运作起来?

这个就要看Demo10_HeroLogic类了：

```
/// <summary>
/// 英雄逻辑处理
/// </summary>
18 references | You, 4 hours ago | 1 author (You)
public class Demo10_HeroLogic : EntityLogic {
    2 references
    private GameFramework.Fsm.IFsm<Demo10_HeroLogic> m_HeroFsm;
    3 references
    private FsmComponent Fsm = null;

    6 references
    protected override void OnInit (object userData) {
        base.OnInit (userData);

        Fsm = UnityGameFramework.Runtime.GameEntry.GetComponent<FsmComponent>();

        /* 英雄的所有状态类 */
        FsmState<Demo10_HeroLogic>[] heroStates = new FsmState<Demo10_HeroLogic>[] {
            new Demo10_HeroIdleState (),
            new Demo10_HeroWalkState (),
        };

        /* 创建状态机 */
        m_HeroFsm = Fsm.CreateFsm<Demo10_HeroLogic> (this, heroStates);

        /* 启动站立状态 */
        m_HeroFsm.Start<Demo10_HeroIdleState> ();
    }
```

启动状态注意有以下步骤：

a.创建状态数组，有多少个状态类，都加到数组里

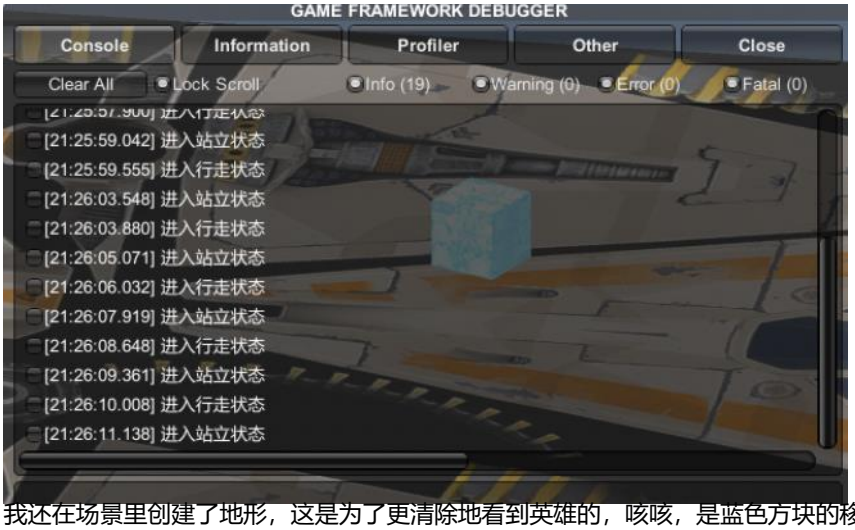
b.通过框架的FsmComponent组件的CreateFsm函数创建状态机，需要指定持有者类型、持有者对象、状态数组

c.调用状态机的Start函数启动初始状态，需要指定状态类型

6.运行

OK，现在一切都准备好，下载源码，运行游戏，看看效果吧。

按下键盘的w、s或者上下方向键控制英雄（就是，那个蓝色的方块）移动，如果一切顺利的话，应该要看到以下输出：



我还在场景里创建了地形，这是为了更清除地看到英雄的，咳咳，是蓝色方块的移动。
另外，按A和D还能旋转角度，很厉害吧（所以有个屁用啊）。

好了，状态机的使用就介绍这么多，只要理清了FsmState和持有者，其它的看看API文档就行了。