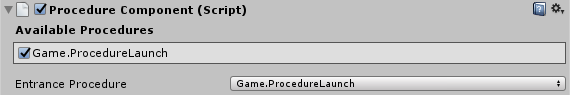
# 流程使用手册

流程是贯穿游戏运行时整个生命周期的有限状态机。通过流程，将不同的游戏状态进行解耦将是一个非常好的习惯。

对于网络游戏，你可能需要如检查资源流程、更新资源流程、检查服务器列表流程、选择服务器流程、登录服务器流程、创建角色流程等流程，而对于单机游戏，你可能需要在游戏选择菜单流程和游戏实际玩法流程之间做切换。

如果想增加流程，只要派生自 [ProcedureBase](http://gameframework.cn/api/class_game_framework_1_1_procedure_1_1_procedure_base.html) 类并实现自己的流程类即可使用。

在导入插件后，查看 Procedure 的 Inspector 窗口。

[](http://gameframework.cn/wp-content/uploads/2016/08/Procedure-Inspector.png)

此时，有一个默认的流程 ProcedureLaunch，且这个流程作为 Entrance Procedure，在游戏运行时，将会从这个流程启动。

在这里，我们创建一个新的流程 ProcedureMain，当游戏从 ProcedureLaunch 启动完成后，再直接进入 ProcedureMain。

using GameFramework.Procedure;

using ProcedureOwner = GameFramework.Fsm.IFsm<GameFramework.Procedure.IProcedureManager>;

namespace Game

{

    public class ProcedureMain : ProcedureBase

    {

        // 游戏初始化时执行。

        protected override void OnInit(ProcedureOwner procedureOwner)

        {

            base.OnInit(procedureOwner);

        }

        // 每次进入这个流程时执行。

        protected override void OnEnter(ProcedureOwner procedureOwner)

        {

            base.OnEnter(procedureOwner);

        }

        // 每次轮询执行。

        protected override void OnUpdate(ProcedureOwner procedureOwner, float elapseSeconds, float realElapseSeconds)

        {

            base.OnUpdate(procedureOwner, elapseSeconds, realElapseSeconds);

        }

        // 每次离开这个流程时执行。

        protected override void OnLeave(ProcedureOwner procedureOwner, bool isShutdown)

        {

            base.OnLeave(procedureOwner, isShutdown);

        }

        // 游戏退出时执行。

        protected override void OnDestroy(ProcedureOwner procedureOwner)

        {

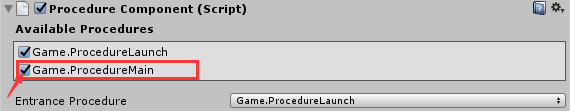
            base.OnDestroy(procedureOwner);

        }

    }

}

通过编译后，Inspector 中会出现这个新增加的流程，勾选上这个流程，使之成为可用流程之一。

[](http://gameframework.cn/wp-content/uploads/2016/08/Procedure-Inspector-2.png)

修改 ProcedureLaunch 的 OnUpdate 方法，使用 ChangeState 方法在 ProcedureLaunch 首次轮询时就切入 ProcedureMain 流程。

        protected override void OnUpdate(ProcedureOwner procedureOwner, float elapseSeconds, float realElapseSeconds)

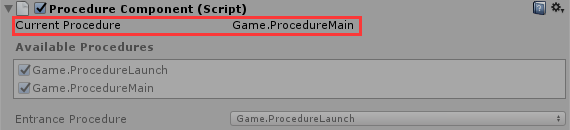
        {

            base.OnUpdate(procedureOwner, elapseSeconds, realElapseSeconds);

            ChangeState<ProcedureMain>(procedureOwner); // 切换流程到 ProcedureMain

        }

运行游戏，观察 Inspector 变化，会发现从 ProcedureLaunch 快速切换到 ProcedureMain。

[](http://gameframework.cn/wp-content/uploads/2016/08/Procedure-Inspector-3.png)

流程的创建和切换就这样简单的完成了。

**相关 API 参考手册**

* [ProcedureComponent（流程组件）](http://gameframework.cn/api/class_unity_game_framework_1_1_runtime_1_1_procedure_component.html)
* [ProcedureBase（流程基类）](http://gameframework.cn/api/class_game_framework_1_1_procedure_1_1_procedure_base.html)
* [FsmState（有限状态机状态基类）](http://gameframework.cn/api/class_game_framework_1_1_fsm_1_1_fsm_state.html)
* 本文固定链接: [http://gameframework.cn/archives/198](http://gameframework.cn/archives/198" \o "流程使用手册)
* 转载请注明: [Ellan](http://gameframework.cn/archives/author/jiangyin" \o "由Ellan发布) 2016年08月26日 于 [Game Framework](http://gameframework.cn/" \o "访问Game Framework) 发表