# **腾讯游戏学院品鉴会路演项目《元能失控》研发经验分享**

作者：柴学梁

链接：http://gad.qq.com/article/detail/287593

导语：腾讯游戏学院品鉴会上，许多优质创意游戏分享自己的研发经验，并通过与现场专家的交流，获得启发，带有Roguelike元素的联机合作冒险游戏《元能失控（Metaverse Keeper）》就是其中一款。目前，《元能失控》已经通过WeGame的发行评测，并在测试区上线，欢迎大家前往体验并提出改进的建议。

**以下为分享内容：**

火花游戏工作室，从2017年初成立至今，逐步从7人发展到目前的十余人。目前的团队成员有参与《生化奇兵》、《XCOM》等游戏制作的主机游戏开发者，也有曾经参与《建党伟业》的影视行业从业者。

在腾讯游戏学院品鉴会上，我们自嘲是一个老年游戏开发团队，不过最近加入不少对游戏制作抱有极大热情的95后，大大减低了团队的平均年龄。



我们团队的部分成员也是《符石守护者》的原创团队成员，从2013年开始进行独立游戏的开发，中间也设计开发过一款Metroidvania类型的横版动作游戏（还未上线）。从2017年初，我们重新成立了新的工作室，开始着手《元能失控》的开发。



这里给大家分享一些我们项目研发过程中遇到的一些问题。

**01**

**联机方式**

《元能失控》作为一个联机游戏，联机是我们立项之初最关心的技术核心点。

我们团队以往的经验都是以开发单机游戏为主，我个人也只有在开发MLB2K11 Wii版本的时候实际接触过帧同步联机，不过这个版本的联机模式也因为种种原因最终没有整合到发售的游戏版本中。

作为一个小团队，我需要一个能够最快速完成，最小人力成本的方案来解决，在进行了一些调研之后，Photon Bolt成为当时的选择，P2P+Steam，非常符合我们的要求，也是一个成熟商业公司推出的解决方案，也有成功的案例。

我们大概从2017年5月份着手联机部分的开发，到同年6月，可玩的DEMO就成型了。当年的China Joy上我们第一次展示了当时的联机版本。



第一个联机版本

不过在这个过程中我们发现就当时的情况，状态同步并不是最适合我们的。作为只有2名程序员的团队，我们需要投入大量的时间在各种功能的开发上（整个开发过程中我们加入过大量的Feature，也砍掉了大量的Feature），如何进行状态同步与游戏的数据结构以及运作逻辑相关，随着游戏的特性越来越多，额外需要花费在联机同步上的时间成本也相应增加，这成为我们快速迭代版本的一个障碍。同时，我们开始有未来开发移动版本的计划，P2P的方式在复杂的移动网络环境下并不适用，如果采用服务器/客户端的方式，我们需要开发自己的状态同步服务器，需要专职的服务器程序来进行，这种方式流量的消耗也较大。

在CJ回来以后，我们权衡利弊，最终决定迁移至帧同步作为《元能失控》的联机方案。

**原因：**

· 前期投入基础开发，后期可以像开发单机游戏一样来进行，游戏特性与联机模块之间几乎没有牵连。

· 较小的消耗的流量，考虑未来可能的移动平台联机。

· 服务器方案较状态同步要简单，不管是P2P的方式还是服务器的方式都较易实现。

当然，过程是痛苦的，大量Unity内置的模块在这个方案下不能使用，我们需要寻找或者自己实现相关的模块用于帧同步。大概花费了一个半月时间完成了以下几个要点，整个过程几乎重写了一遍游戏：

· 逻辑层和渲染层的分离；

· 随机数的统一，保持客户端随机数的一致性；

· 用定点数数学库替代Unity数学库；

· 使用Deterministic的物理库替代Unity物理库；

· 使用A\*寻路算法替代Unity NavMesh。

当然，这里我们不是在对比帧同步和状态同步的优劣，而是分享什么样的方案适合我们的项目和团队。

帧同步的方案也有很大的局限性，譬如很多好用的第三库很难在这个体系下直接应用进来，或需要大量的改造才能使用。当有复杂功能的需求时，难免要造一些轮子。

同时，我们也获得了一些帧同步天然的优势，比如游戏回放，只需要记录玩家的输入，便可以原样还原整个游戏过程，这在后来我们用于Debug定位Bug起到了非常重要的作用。

**02**

**光影效果**

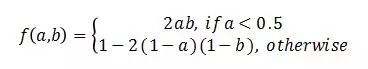
《元能失控》是一个俯视角的2D游戏，整个场景是用Tilemap构建的，在项目的前期我们尝试过几种不同的方式来提高视觉表现：

· 用Unity的基本光照来对场景打光；

· 用类似2D Dynamic Lights and Shaders插件这种方式来动态生成阴影。

但尝试下来整体效果并不是特别好，尽管花费了很多时间来调整参数，最终也没有办法达到美术想要表现的效果。

最终我们回归到最适合美术来调整效果的一种简易方式，在引擎中还原Photoshop Overlay的方式来对基础场景层和光影效果层进行叠加。



这样一来，对美术调整场景的效果变得非常友好，场景美术在Photoshop里就可以预览最终能够呈现的效果，所见即所得。配合后处理插件，基本上可以达到满意的效果。

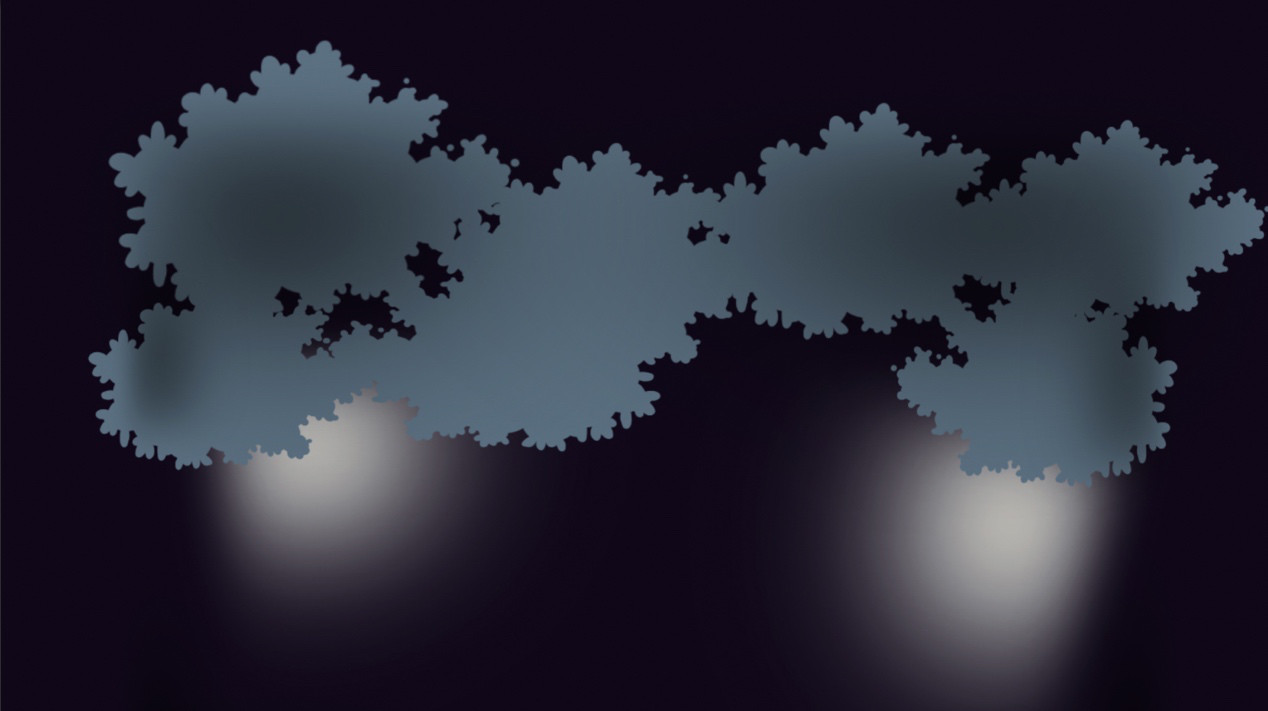
我们把场景分到两个Layer中，光影相关的Sprite分到Overlay层，并且在渲染场景前做一次Render To Texture，把这些贴图渲染到一整张大的纹理中，在场景渲染的Pixel Shader中按上述方式进行混合。

这种方式在需要制作光照效果的地方能够有效提亮，在阴影等需要变暗的地方也能够有效压暗。

如下图所示：



原始场景



光影Overlay贴图



最终效果

03

**2D层级问题**

在2D游戏开发过程中，层级问题是一个常见且头疼的问题。2D游戏不像3D游戏的渲染方式，可以通过深度值自动进行深度测试并决定最终写入FrameBuffer的像素，而是在渲染前就根据渲染顺序决定了最后的渲染结果。

2D渲染的物体，Unity提供了Sorting Layer和Sorting Order来最终决定渲染的顺序。

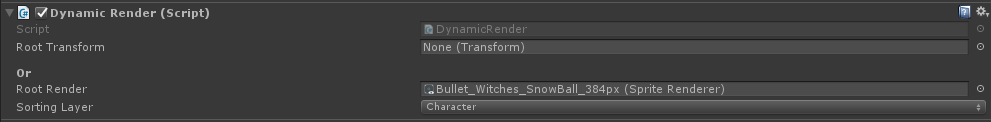
首先，对于静态的物体以及能够明确场景中层级的问题，我们在Sorting Group中设定好对应的Sorting Layer，来确保总体层级上不出错。比如关卡房间的地面，设定为RoomBG, 角色设定为Character, 像地面上的岩浆、毒液等，它们应该处于地面和角色之间，暂且定义为UnderCharacter，使得在最终渲染时 RoomBG > Character > UnderCharacter。



场景内的Sorting Layer

当然，根据实际需求，往往还需要更多的Layer来处理复杂的层级关系，这里仅作示意。

在同一层次内，Sorting Order通常不是固定不变的，角色、怪物、子弹等场景中的物体需要根据他们实时的位置进行动态调整。在这些物体上，我们加入了一个Dynamic Render的脚本，在游戏的运行过程中根据物体的Y值实时更新Sorting Order。



这里需要注意的是，2D物体在视觉上也是有体积的，需要选定2D物体中正确的点并获得它的Y值，才能得到视觉上正确的结果。所以脚本中我们开放了一个Transform结点，供美术在需要进行调整的时候使用。

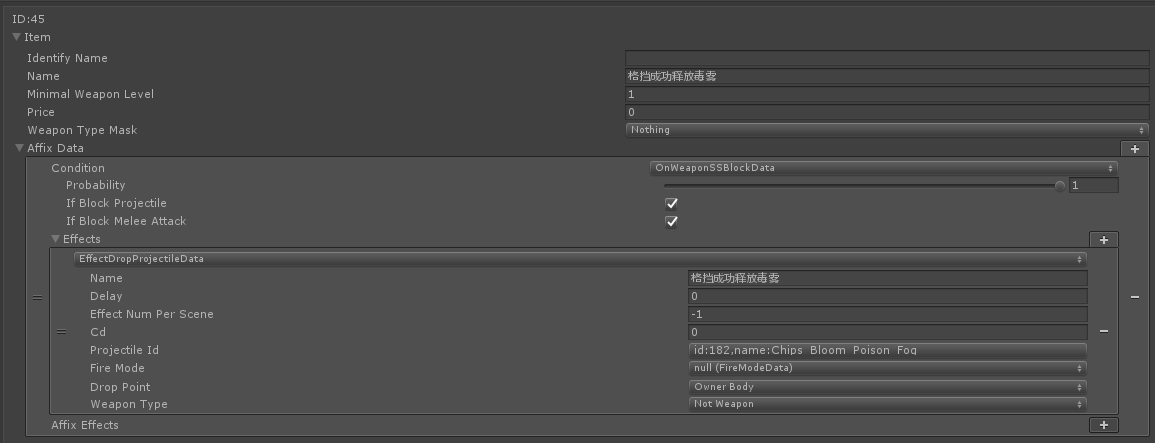
在很多复杂情况下，这种方式也不能完美解决显示层级的问题，通常需要很多小的技巧来解决，特别是《元能失控》的场景做了很强的3D透视感，经常会遇到一些棘手的问题，让我们的程序感慨下一个游戏一定要做3D的。（我们团队大部分美术和技术都是3D背景的。）

04

其它

在整个研发过程中，我们通常把游戏的内容开放给策划来进行配置，比如怪物的AI状态机逻辑、武器道具芯片的各种效果等。

比如下图是一个武器词缀的编辑器，我们制作了大量的触发条件以及词缀效果，策划可以在编辑器中自由组合，产生出全新的词缀效果，条件和效果中的参数开放给策划来调整。



总得来说，不同的游戏类型和团队的实际情况，都会遇到不同的实际问题需要去解决，没有哪一个方法是放之四海而皆准的。在《元能失控》的研发过程中，我们遇到的问题大多并非解决难度上的障碍，而是哪种方法更适合现在这个项目和团队。在遇到一些问题的时候，通常没有完美的方案，我们需要做一些取舍，用合理的代价去解决。

由腾讯游戏学院联合极光计划共同主办，旨在发掘和扶持优质创意手游的第四期“腾讯游戏学院品鉴会” 极光计划专场正式启动报名，11月10日将落地深圳，大力欢迎国内外开发者们提交游戏项目，前来交流。

**电脑端报名：**

<http://gameinstitute.qq.com/gameshow>