打造提升创意和改善效率的工具

Charles Brandt

- 工程师和艺术家的背景差异很大，沟通起来有时不太容易。

- 通过打造非工程师人员使用的工具，某些任务可以不再需要工程师来承担。
- 而且，这些任务本来也更适合由非工程师人员来从事。

实际案例: DinerDash PHPExcel

目标是将游戏的经济变量列到一个excel文件中。

好处

- 游戏设计师可以自己输入这些变量，给工程师节约一些时间。
- 设计师可快速调整变量，从而对游戏进行微调，并设置特别的临时游戏场景。
- 游戏中的变量更易读。

困难

- 必须严格遵守Excel表格的格式。
- 有些文件出现故障，我们最后发现，其中一个文件是用Open Office编辑的，保存下来的文件不兼容。

是否值得?

- 从长期来看，这是一个非常好的想法。但是在我们发现文件故障原因之前，制作工作停止了近一天。

- 最好创建一个全公司的wiki来记录这些错误及其解决方案。另外，还可以张贴标签，以方便查看。

实际案例: Metamorphosis 图像分析冲突侦测

- 按照固定的间隔，对一小段像素进行分析。
- 当侦测到设定的颜色，即激活不同类型的冲突

好处

- 游戏设计师可以很快地测试关卡布局，而不需要程序员为他们进行更新。
- 无需一关一关进行编程。一个冲突侦测引擎就可以完成全部工作。
- 只需通过指定新的颜色，就可以快速添加新的行为。

困难

- 创建这个引擎，并让其顺利运行，可能要花一些时间。
- 在应用于一个小的侦测区域时，效果最佳。
- 在运行缓慢的机器上，可能效果不那么好。
- 两个侦测周期之间的活动，将会被忽略。在这方面需要采取必要的措施，以保证侦测的准确。

出现的问题及一些解决方案

- 在侦测中采用的颜色，你不希望与游戏中出现的颜色相同。解决方案是，为你的关卡制作一张侦测图。保存一张关卡美工图，然后用任意颜色标出你希望对其进行侦测的区域。该侦测图不会显示在屏幕上。

- 侦测运行地太慢，而且你没有引发冲突。试着通过提高侦测率，并进行试错，以找过问题的症结。你可以试着缩小侦测区域的样本规模。加快侦测的另一个方法，是每两个像素进行一次检测，从而能以更少的计算量，覆盖一大片区域。

实际案例: Closet Hero
产生YAML的软件

- 游戏中的物品、其价格、描述等，均在一个YAML文件中予以明确，
- 由于YAML在空间的使用方面有严格规定，因此我们不希望非编程人员对其进行直接修改。

好处

- 大幅提高速度，并让游戏开发人员能够更方便地添加及更新游戏物品。
- 让程序员不再需要无休止地对持续更新的文件进行调试。
- 设计师不需要阅读代码；他们使用的是一个简单易用的图形用户界面。

困难

- 最初创作该软件，花费了一名程序员两周的时间。
- 在创作该软件期间，作出一项决定：该工具应与工作室的所有游戏兼容。每款游戏用不同的风格来安排其YAML文件的格式。这就需要用一个文件来为每款游戏制作一个图解。这样做的一个好处就是，为今后的游戏设定了一个标准化的格式。

是否值得?

- 虽然创作该软件采用了多次迭代，但该软件受到了多个工作室的欢迎。
- 该软件是必需的；该软件被创作出来后，所有相关工作室开发人员的压力得到大大缓解。

实际案例: Fire Horse Gaff工具

- 投币机游戏在行业展会上展示或向客户展示时，我们需要找到一种能快速展示该游戏各种功能的方法，无需等到客户学会熟练操作。
- 我们添加了一项新功能，该功能由添加到现有界面的一个按钮激发。

好处

- 该项新功能在展会上大获成功。
- 通过对一个XML文件进行配置，添加新功能的工作得以轻松完成。
- 该功能被激活后，实践证明，其在缺陷测试中也能发挥作用。

困难

- 采用一个密码来访问旧路径，以强制产生结果。对一些人而言，记住这个密码有些困难。
- 1.0版本创建出来后，引起了众人强烈的兴趣。大家对功能提出了更多更多要求，因此又发布了2.0版。
- 关于配置XML，有一些疑问。但这些疑问都通过发布在办公室wiki上的文件得到了解决。

是否值得?

- 在以前，潜在客户都雇人来玩游戏，直到得出某种结论。这种做法不再必要。实践证明，简单的用户界面易于使用，同时可以将卷轴符号保持在隐藏状态。

实际案例:粒子编辑程序

- 大多数美工师都不知道如何修改粒子特效代码，程序员并非创造视觉效果的最佳人选。因此做出决定：应该编写一个软件，让美工师能控制粒子效果的创建。

好处

- 过去使用过的粒子，可重复利用，并且可作为修订版本的基础。
- 美工师可创建自己喜欢的效果，无需和程序员反复讨论。
- 对于大部分游戏而言，粒子效果比手工动画效果更有效率。

困难

- 已存在一个编辑程序，但不直观，美工师不喜欢该程序。需要对哪方面进行变更？
- 我们如何能给美工师很大的权限，并同时让他们了解每样东西的功能？

是否值得?

- 该项目当前处于开发阶段，但alpha版本大受美工师欢迎。
- 我们计划采用工具提示来解释所有各种选项的功能，从而让其处于隐藏状态，只在需要时出现。
- 最终版本将导出一个效果文件，程序员将在游戏中执行该文件。

须遵循的一些指导方针

- 根据通用性原则来创建。你永远不知道今后需要变更或添加什么。
- 按照一个深思熟虑计划来创建，该计划考虑到今后如何进行修改。
- 对你的工具进行版本控制，这样在必要时，你可以随时取用老版本。

须遵循的一些指导方针（接上页）

- 让你的工具自成一体（self-contained），从而使其能方便地被整合到其它软件中。
- 撰写你自己的文档，并将其保存到易访问的位置。直到完成这一步，你的工作才结束。
- 如果可能，将项目的涉及范围缩小。
- 限制程序员人数。程序员人数少，在初期开发中就更不容易引起混乱。

总是会出现的问题

- 一旦大家看到你制作的工具很好用，他们都会要求从你这里定制。在这个阶段，会有很多变更，因此要确保你的代码组织有序，不至于在项目结束时无法辨认。

谢谢!

- 有何疑问?