Git要点：

1. 分为工作区，本地仓库，远端仓库，工作区是开发的地方，会产生修改；本地仓库是远端仓库的完整克隆，commit操作是将工作区的修改内容应用于本地仓库；远端仓库相当于工程的终端，所有的开发者最终的提交汇总之地，对应于pull/push操作，pull将远端仓库内容更新到本地仓库，push将本地仓库内容同步至远端仓库。由于pull/push只是本地/远端仓库内容的合并或者说是同步，所以冲突，stage啥的一些概念可以基于工作区和本地仓库理解，不需要考虑远程仓库。
2. 三种状态：
   1. modified: 已修改表示修改了文件，但还没保存到数据库中。可以理解工作区的修改，数据库就是本地仓库
   2. staged：已暂存表示对一个已修改文件的当前版本做了标记，使之包含在下次提交的快照中。什么情况下回出现staged状态，staged状态的文件暂存在哪里？
   3. committet: 已提交表示数据已经安全地保存在本地数据库中。只要是提交到仓库，无论是本地还是远程，都是可以找回的，除非相关的物理设备损坏。

前项目工作流程总结：

个人没用过SVN，但是用过p4，之前项目选用git的两个原因其一是免费，其二是相较svn来说，git提交/更新更快。

git问题主要围绕两方面。

1. git概念比较复杂，学习成本较高。
2. git无法锁定文件，所以更容易产生冲突。对于代码来说，冲突双方沟通合并就可以，但是对于资源就没有这么简单，很有可能导致一方工作内容丢失。

基本分支划分

develop：主干，开发阶段的核心分支，几乎所有功能都会最终合并到此分支上。

Feature：功能开发分支，用于开发新的功能，以及bug的修复。

其他branch：比如各个版本的发布分支，基于某个时间节点创建，其相当于是develop的一个子集，最终还会合并到develop。这里有个特例，如果做区域相关本地化的内容，需要酌情考虑。

项目中遇到的问题：

1. 操作不当，导致工作内容丢失：
2. 如果长时间呆在一个分支上开发：
3. 策划原始表格数据的合并(excel)：
4. 多个分支同步进行产生的资源冲突：
5. merge/rebase:
6. 美术转用svn：