准备工作

`Git`是一个命令行工具，对于不习惯敲命令的人来说过于繁琐了，所以我推荐先选择一个图形化的软件。

软件推荐

1. SourceTree 免费且z跨平台(win, osx, linux)，界面简洁易懂
2. GitKraken 界面很是炫目，使用web技术开发，用起来也很舒服
3. Tower 价格较贵，个人认为相比`SourceTree`仅仅有性能上的优势

理论

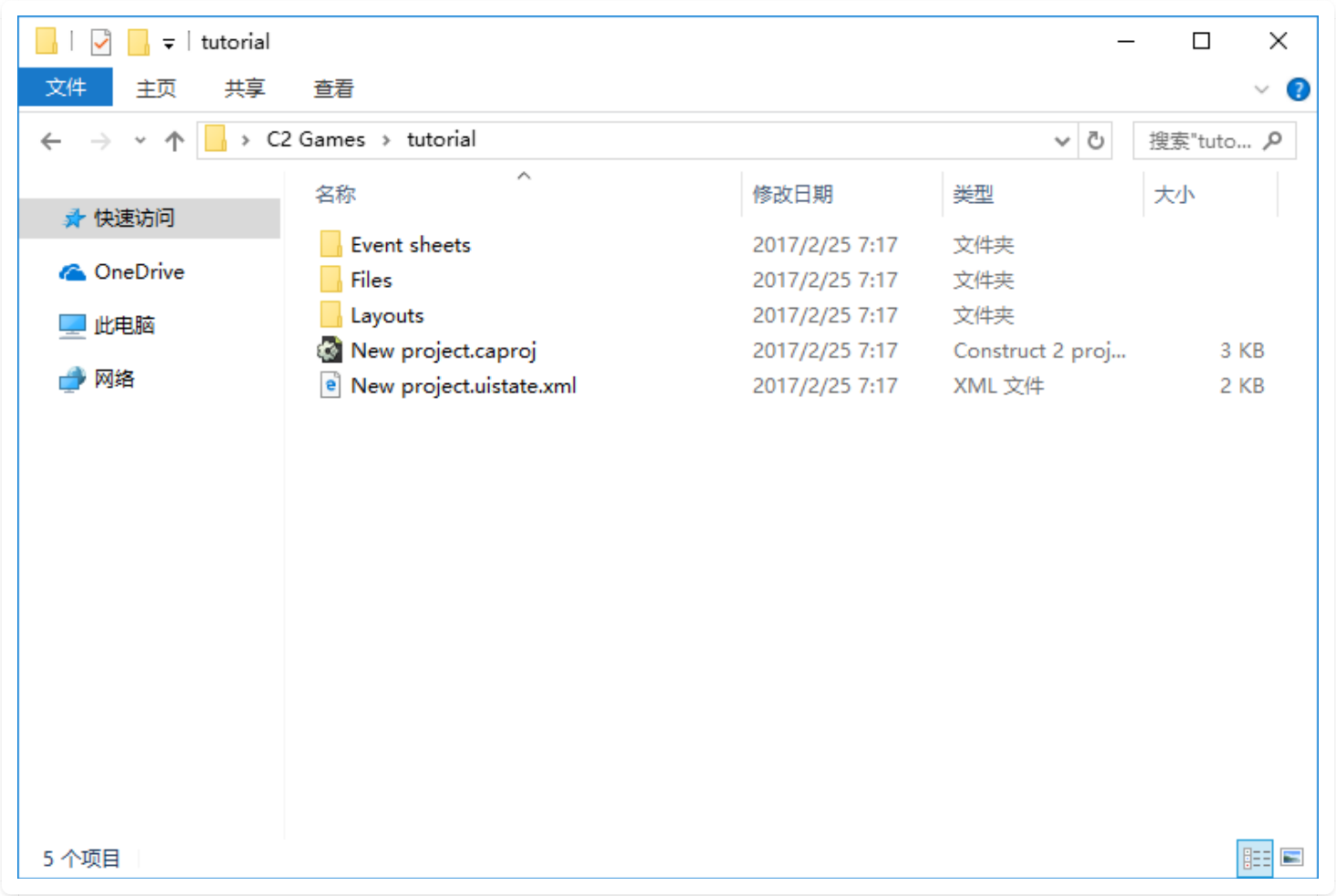
简单来讲，版本控制就是一个可以保存下来的无限UNDO/REDO，并且还附带了一条简单的说明信息。 例如给某个`sprite`加上动画可以算作是一次修改，调整主角的跳跃速度也可以算做一次修改。许许多多次修改就组成了一个可供随意返回的历史，而这个历史会以文本的形式保留下来，这样就不用担心关闭软件之后UNDO的历史也随之消失。

一般来说使用`Git`分为两个部分：管理保存在自己电脑上的项目，和上传去某个项目存储网站。

使用

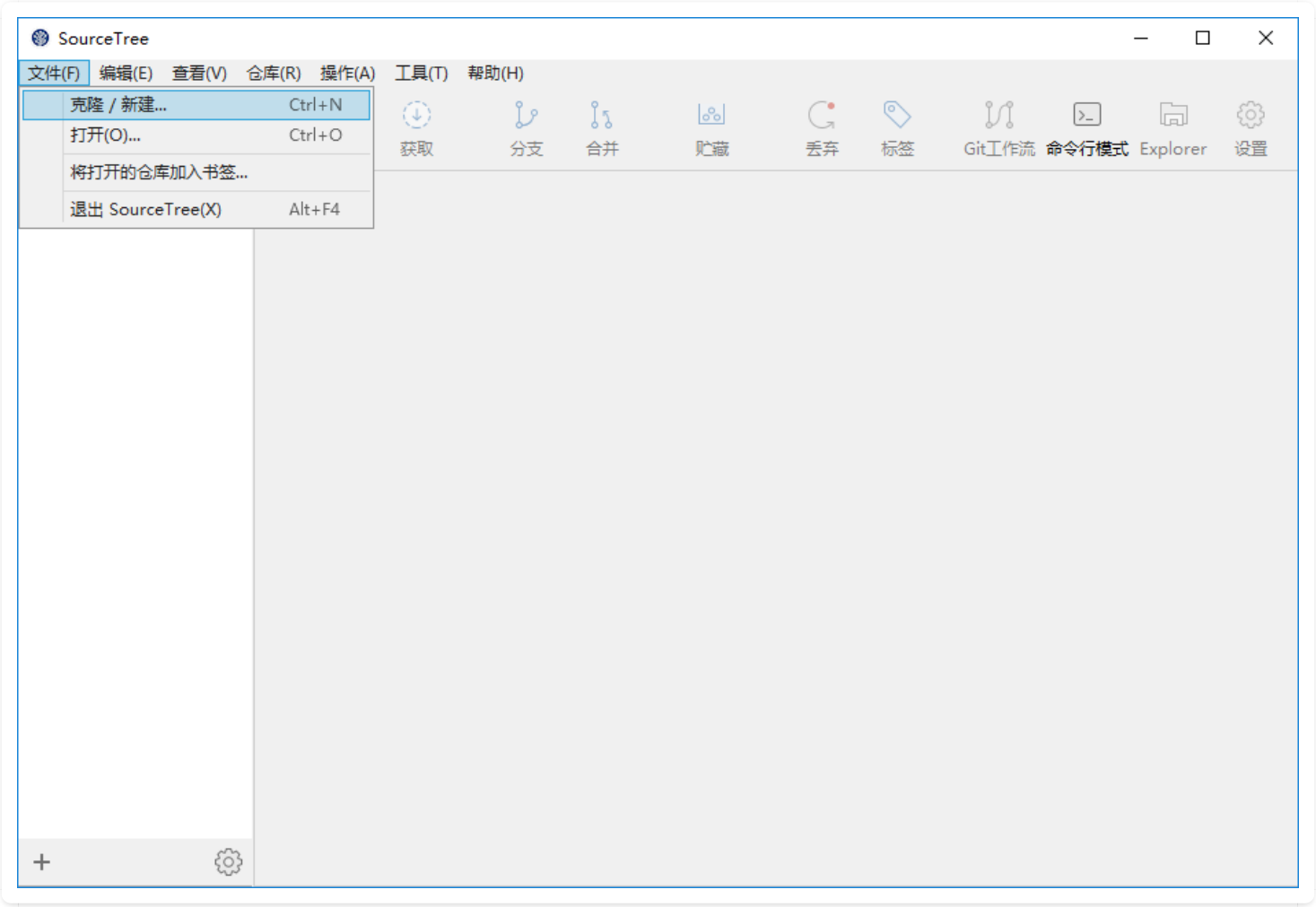
下面就以`SourceTree`(以下称"ST")为主，来说说怎么使用`Git`来管理你的C2游戏。

1. 新建C2工程并保存 (File > Save As Project...)

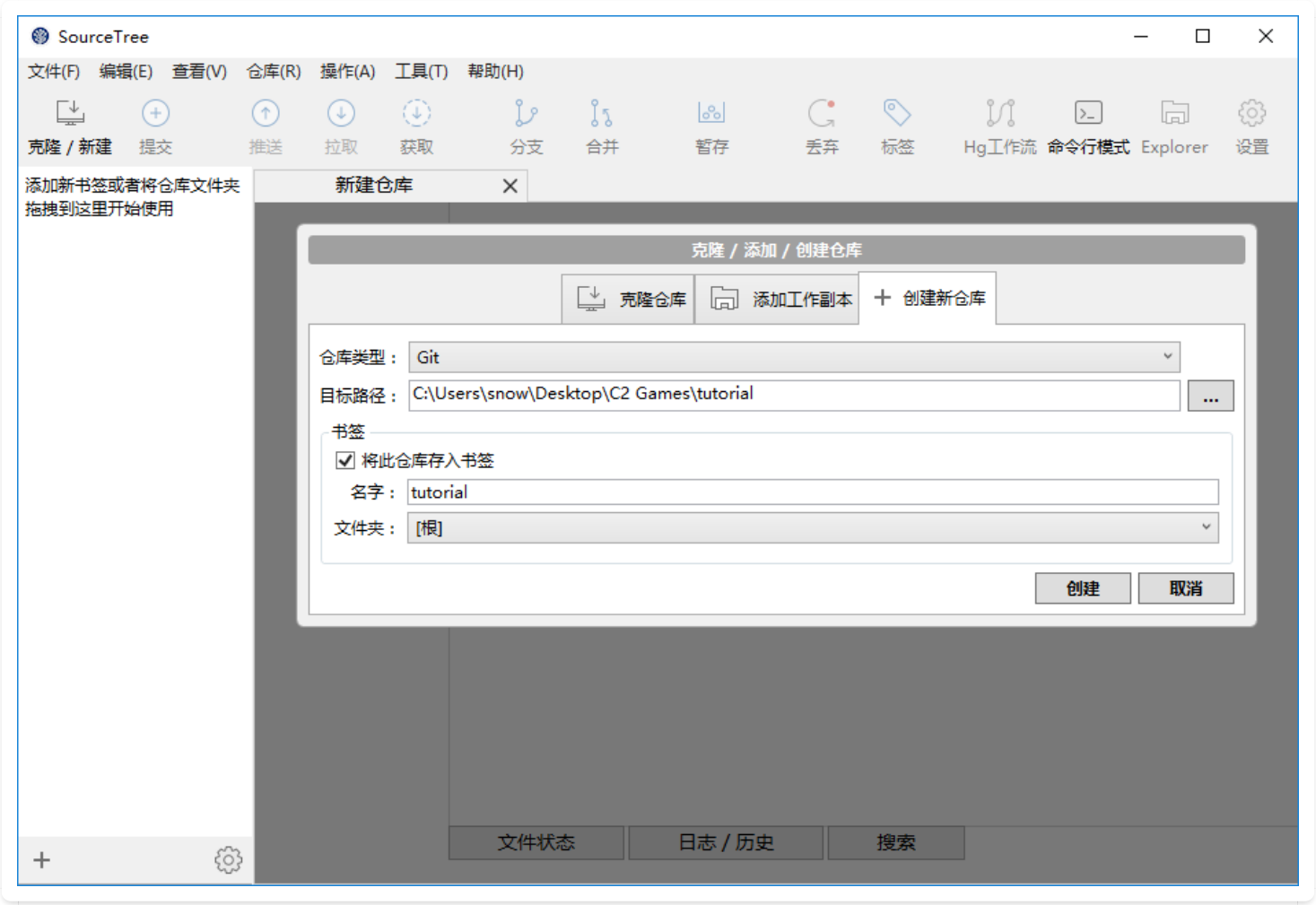


需要注意的是，因为`Git`是被设计来管理程序源代码的工具，所以最适合管理的其实是文本文件。 这就需要你将C2项目保存成为一个文件夹的形式。

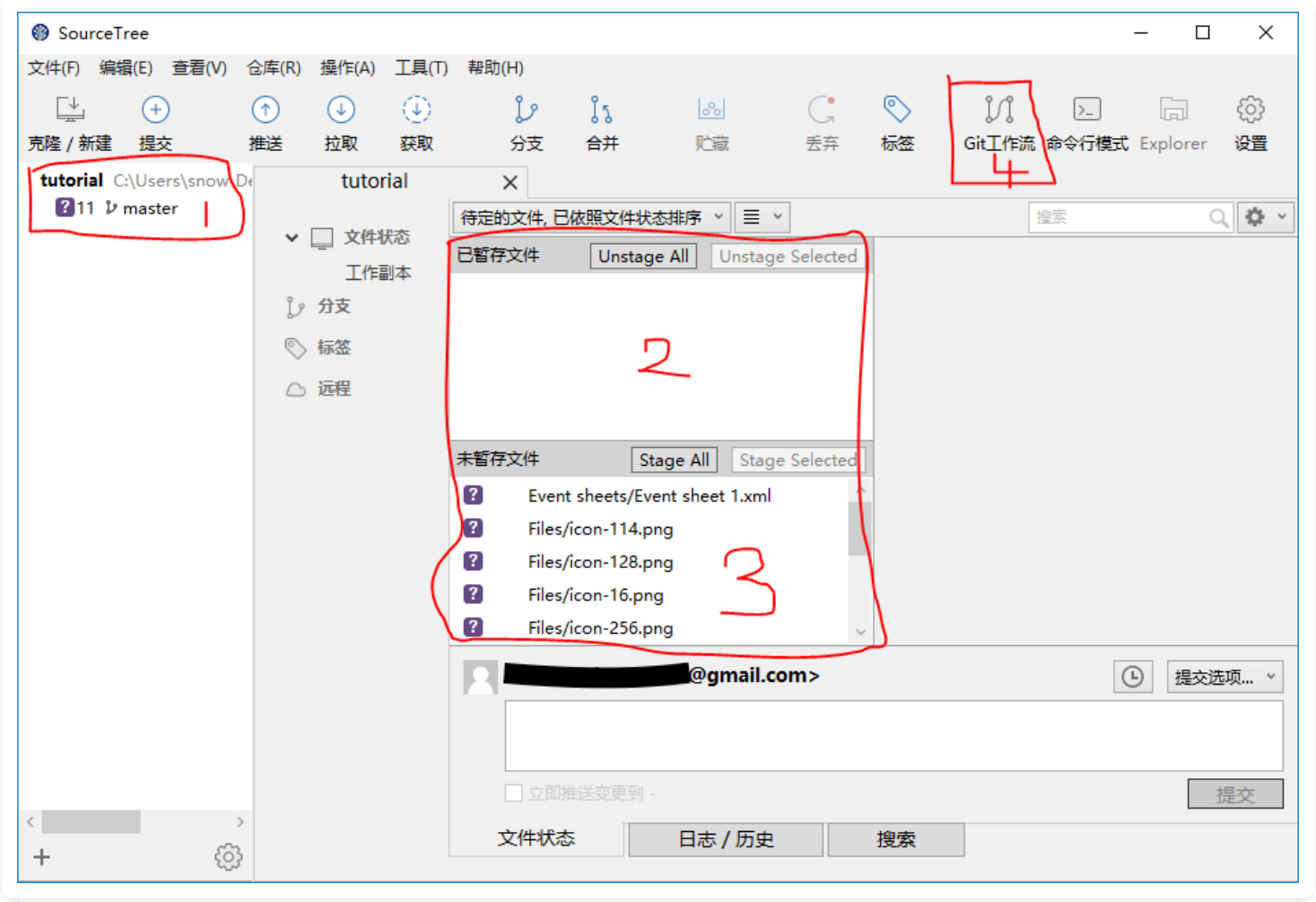
1. 打开`ST`新建一个库



路径选择到刚才保存的那个文件夹


SourceTree
文件(F) 编辑(E) 查看(V) 仓库(R) 操作(A) 工具(T) 帮助(H)
克隆 / 新建
提交
推送
拉取
获取
分支
合并
暂存
丢弃
标签
Hg工作流
命令行模式
Explorer
设置
添加新书签或者将仓库文件夹拖拽到这里开始使用
新建仓库
克隆 / 添加 / 创建仓库
克隆仓库
添加工作副本
创建新仓库
仓库类型：
Git
目标路径：
C:\Users\snow\Desktop\C2 Games\tutorial
...
书签
将此仓库存入书签
名字：
tutorial
文件夹：
[根]
创建
取消
文件状态
日志 / 历史
搜索

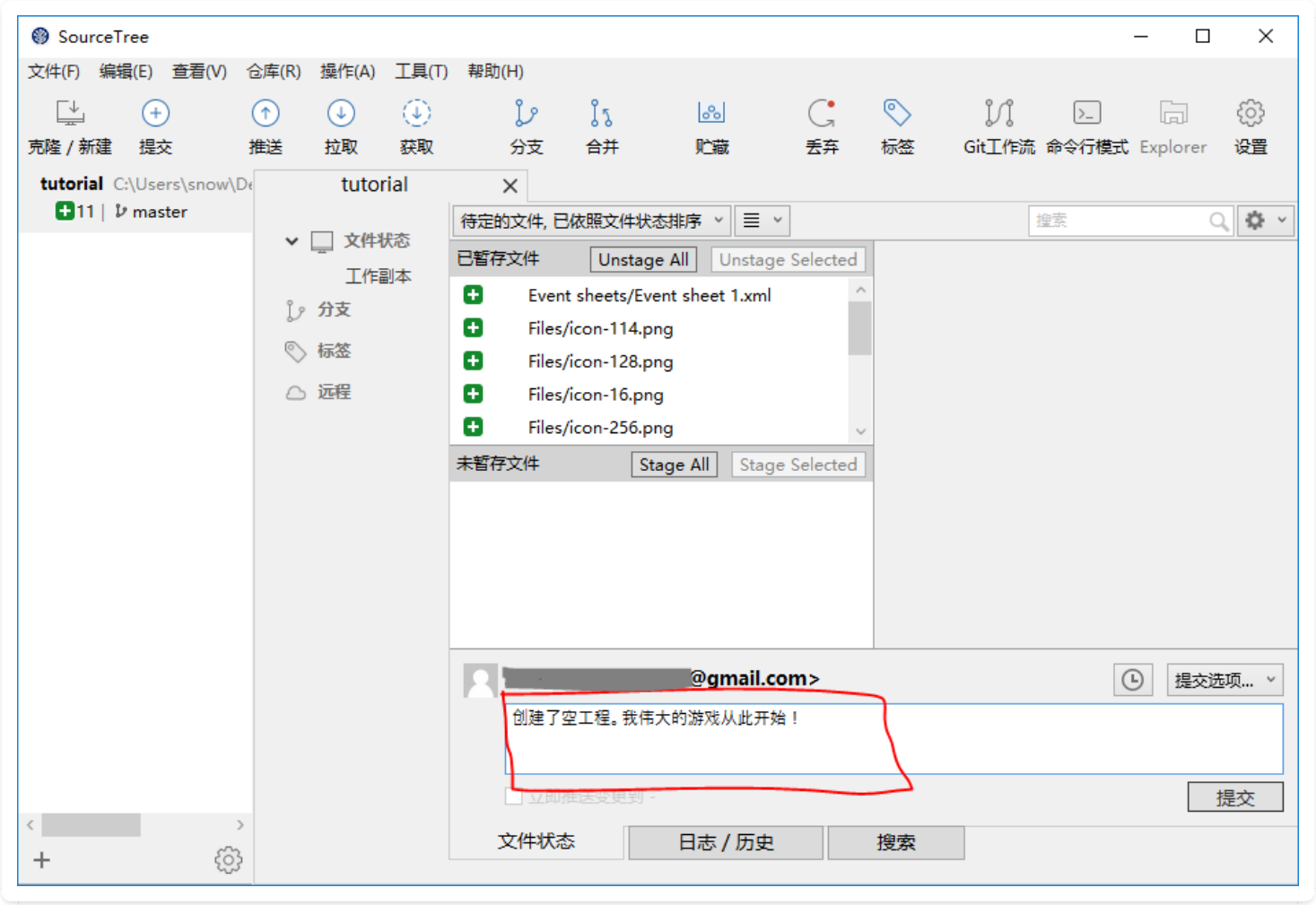
1. 查看库现在的情况



- 左边1号应该是刚刚创建出来的库，包含名称tutorial、11个未记录下的修改、所在分支是master
- 中间的2号是用来显示当前**暂存**的修改，后面会详细讲解，现在应该是空的
- 3号现实的则是已经在C2里面作出的修改，但是还没有被**暂存**。因为我们新建这个库，这个文件夹里面所有的文件都还处在没有被**保存进入**`Git`的状态

1. 把C2空工程全部保存进`Git`

现在我们新建了C2工程，但是Git并不知道，所以我们需要把工程里面的所有文件都丢给他，让它帮助我们管理。



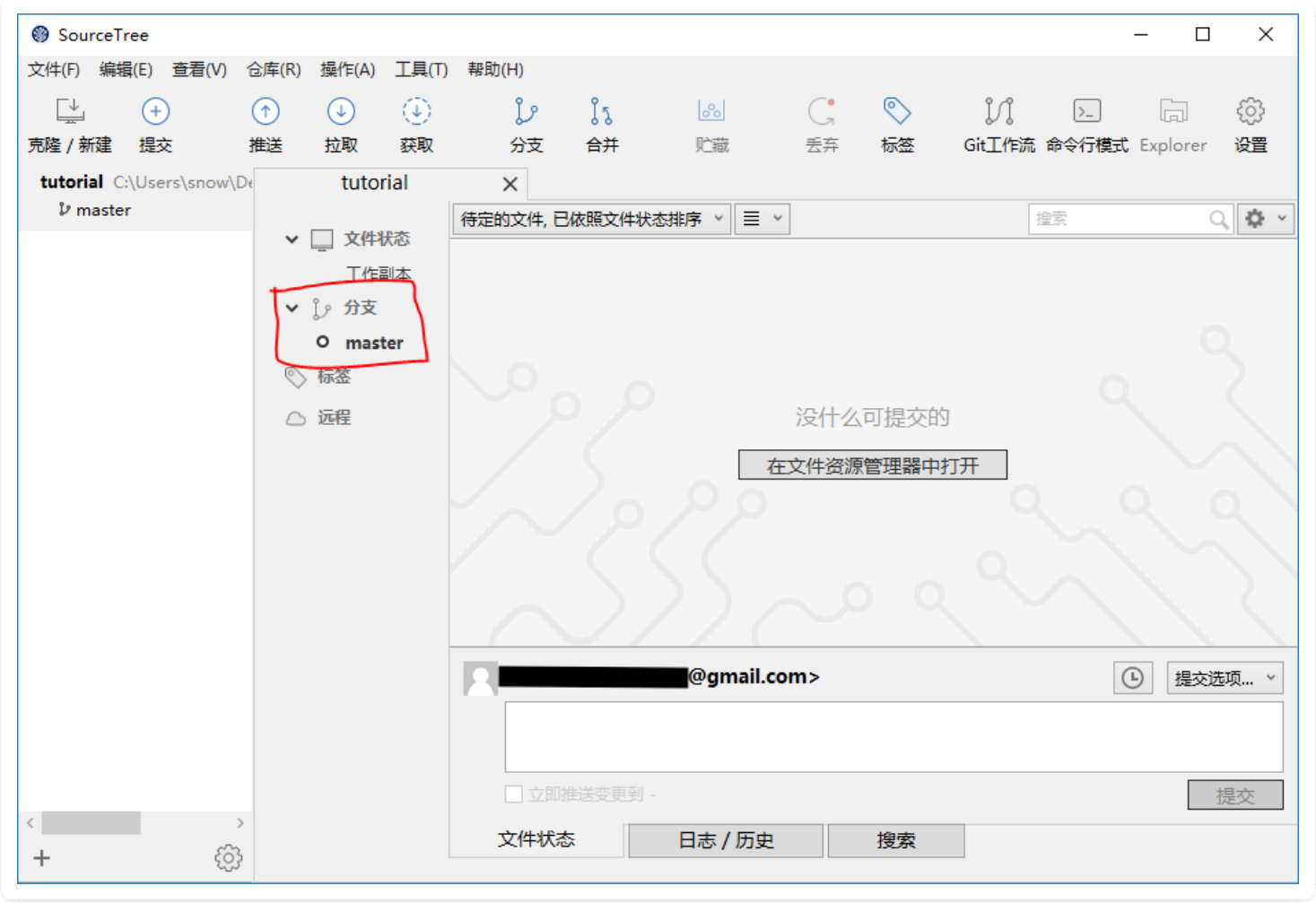
这里你有很多种选择：

- 可以在3号中选择想要保存的文件，然后点击Stage Selected
- 也可以直接点击Stage All，把所有文件保存进去

之后你会看见所有文件都出现在了2号窗口，并且左边显示一个绿色的+号。绿色+号的意思是这文件 还不曾被它记录过，所以属于新加入的条目。

然后在下面的文本框里写上**你这一次保存都做了那些事**，这里面的内容非常重要，要做到以后 回看历史记录的时候，通过这个信息便能清楚的知道这一步都做了什么事。

点击提交就保存完成了，成功之后如果没有剩余未暂存的文件的话，2和3号窗口应该都已经消失。



这时候注意一个很重要的地方，库左边栏的分支下出现了一个`master`。点击这个`master`就会打开这一个分支的历史界面。`master`是最重要的分支，它一般保存的就是你游戏最成熟的功能。在我们这次要使用的方法里面，这一个分支总是保存着完整的功能而没有任何试验的成分。你可以在任何时候切换到这一个分支，然后编译导出游戏。我们会尽量在某个新功能完成的情况下才保存进来，这样可以确保你的游戏在这一条历史上总是可以发给别人试玩的。

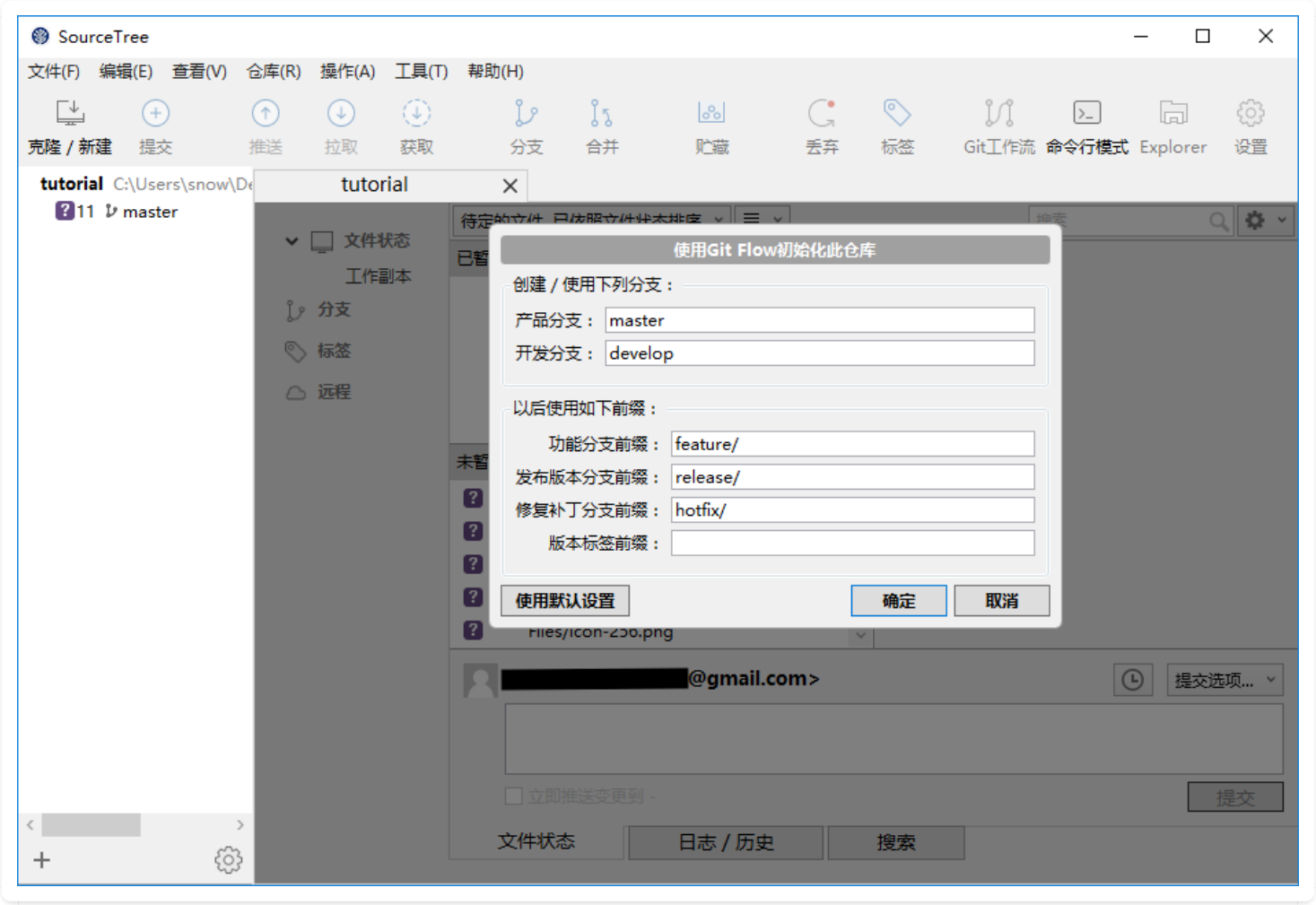
什么是分支?

首先，分支 = 历史。分支保存了这一条历史线上的所有修改，所以它本身就是历史。那会有多个分支吗？答案是确定的，因为这个历史被记录在了电脑上，你就可以从任意一个时间点分支出去，最终改变历史！！！但分支之间是互不影响的，就算以后你开了新的分支并且做了和之前完全相反的事情，那一段历史也不会改变，而改变的只有新创建出来的这个分支。相当于穿越之后其实没有改变自己原来的历史，而是创造了平行宇宙。

因为我们使用一个很特殊的方法，所以不会像其他教程一样深入这个概念，只要大概了解就行。

1. 开启**Git工作流**

大多数教程都不曾涉及过这一概念，但是我觉得这是个非常新手向而且非常值得信赖的解决方案。无论你是一个人开发还是一个大团队，使用这样一个设计的非常好的工作流只会事半功倍。



这里一般保留默认的设置，直接确定即可。等待ST初始化完成，你会发现分支下出现了新条目：`develop`。而且仔细看这一个`develop`还被加粗显示了，其实是ST已经帮助你把现在所在的分支换成了它。

在这一工作流的设计中，`develop`是专门创建出来给你记录开发过程用，等到你的开发完成了一个系列或者完成了一个你计划中的里程碑，就可以发布进`master`分支。

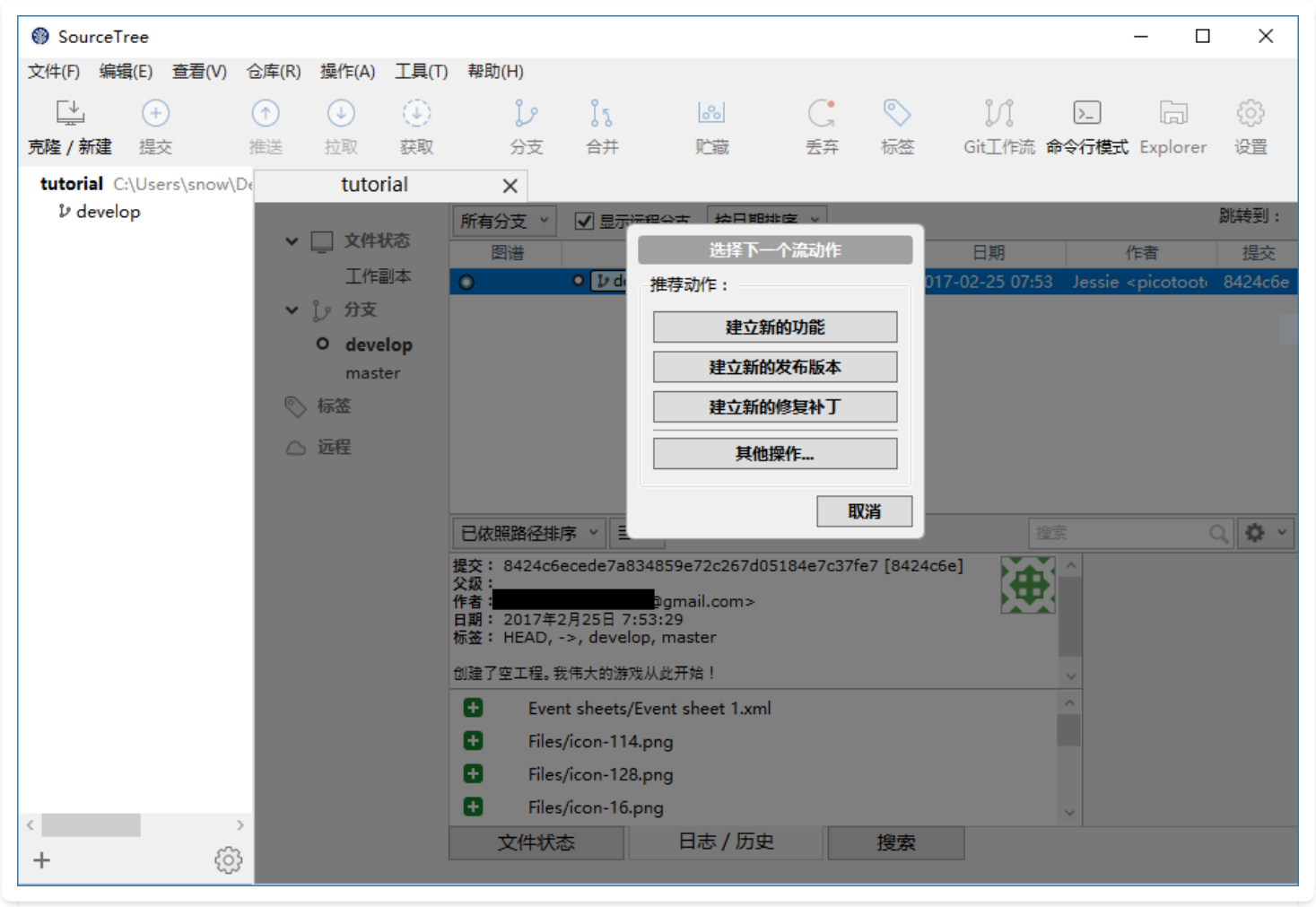
对于C2这样每一样功能都非常“实际”的引擎来说，`develop`分支甚至很少需要直接使用，我们可以单靠不断累积“新功能”的方式来制作游戏。

下面我就来用例子演示Git工作流的这一伟大之处。

1. 新功能，“创建一个主角可以站立的平台”

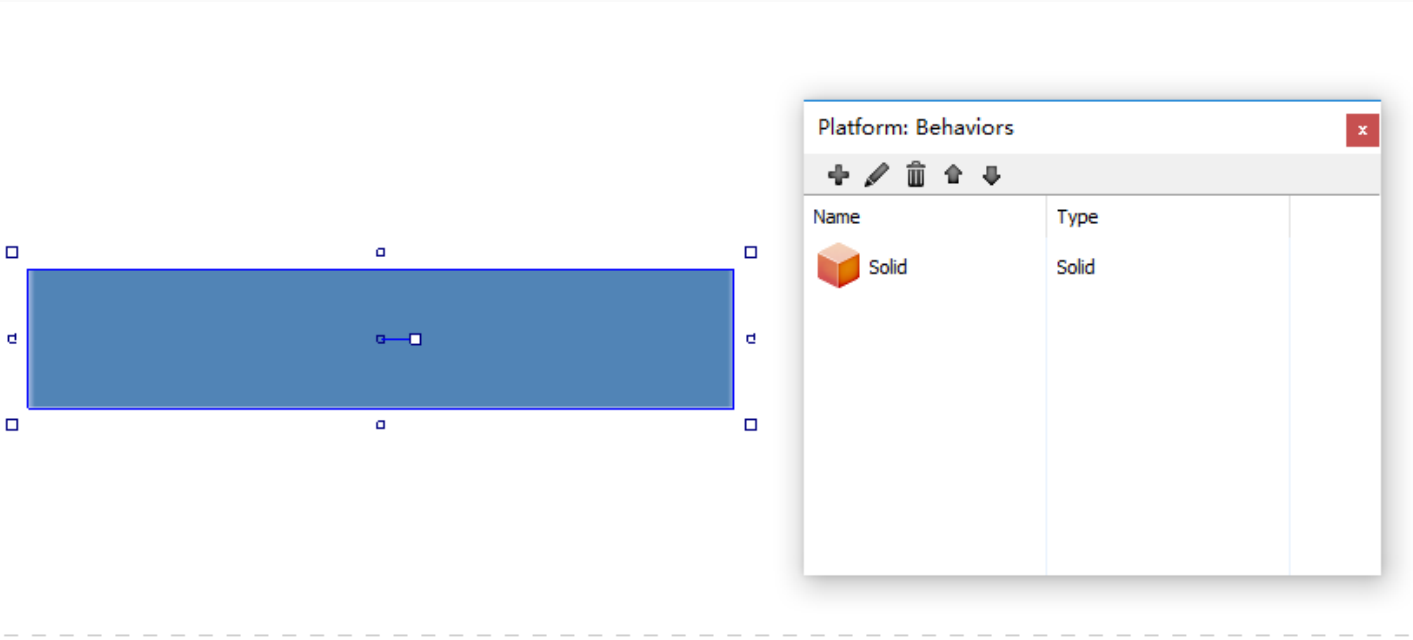
在开始着手制作每一个新功能之前，想好你要做的事情。一般来讲一个词或者一句话能解释的功能比较容易完成。

打开ST，点击Git工作流，选择第一个“建立新的功能”。给这个功能取个名字，然后点击确定。你会发现分支下多出来一个文件夹feature，这个文件夹下就是你刚刚取的那个名字。

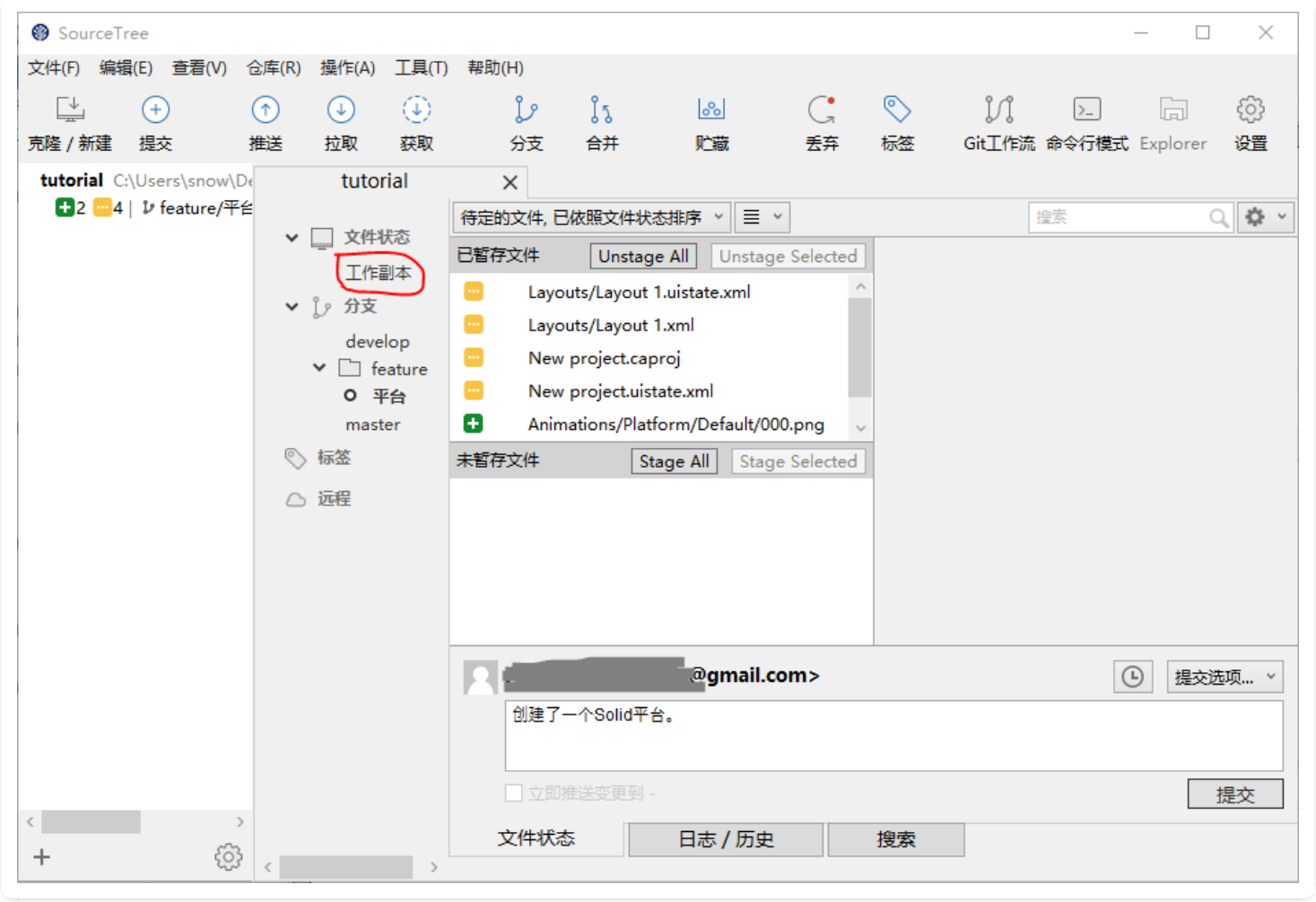


这时候你正处在一个完全独立的环境中工作，无论做什么，就算是把游戏毁了，也不会影响`master`和`develop`这两个关键的分支。所以放心大胆去做吧！

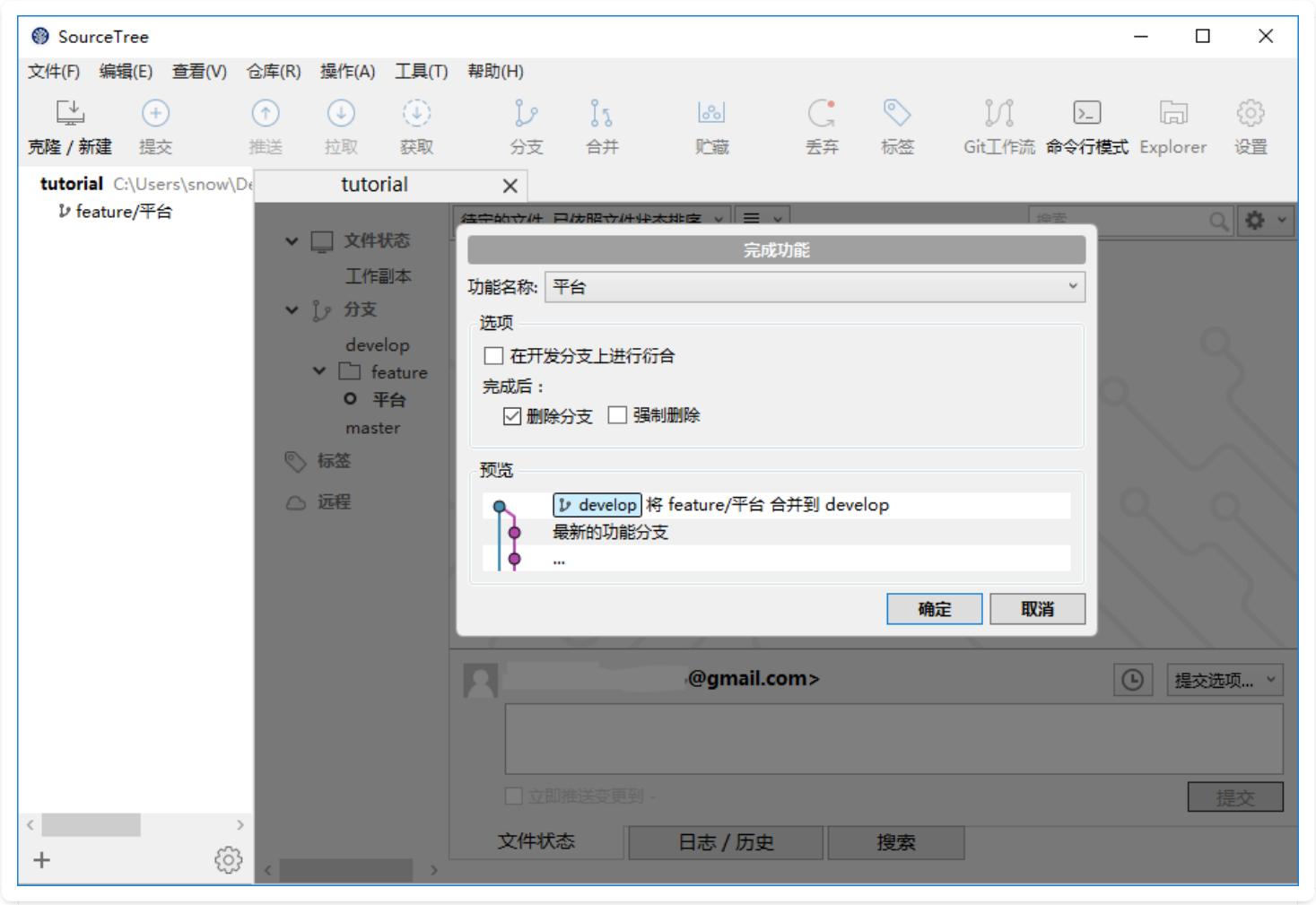
在C2中新建Sprite，并且给它一个Solid行为。这样我们的新功能就完成啦，保存一下，我们可以回去ST把新功能提交给`Git`。



点击文件状态下的“工作副本”，然后保存所有的修改，写上信息提交。



提交完成之后再点击Git工作流，选择完成功能，不用做任何修改直接点击确认。



于是分支下的那个feature和刚创建的功能分支都一并消失了，并且我们又回到了`develop`。

这就是新功能的全部步骤，简单来讲就是：

1. 进ST，用Git工作流“建立新功能”
2. 回到C2开始工作
3. 如果已经做好了一些内容但是还没有完成整个功能，你可以先保存，之后回到ST提交一次修改。
4. 整个功能都完成之后保存，然后回到ST，用Git工作流“完成功能”，写好信息提交之后就可以开始下一个功能啦！

需要注意的是：每次提交之前，记得先保存C2项目，否则ST不会知道你做了什么修改。

制作某个功能的时候，可以另外去做别的功能吗？

当然可以。就算是正在某个功能的分支，也可以开始新功能。但要注意的是，这时候会从`develop`分支 开始，所以你在制作的另一个功能的内容都是不会出现的哦～

你可能想问如果我想包含另一个功能里的内容怎么办呢？我的建议是，虽然可以但请尽量不要这么做。 时时刻刻记着把功能和功能区别开，不要有交叉的地方，这样可以让你做的更好，也可以避免很多麻烦。 用过版本控制工具的话你肯定知道“合并分支”的时候会出现冲突，这会非常麻

烦。尤其是对于C2这样 都是通过图形界面来制作游戏的工具，我们并不知道文本里冲突的到底是什么。

为了避免节外生枝，我强烈建议每个功能都从`develop`开始（也就是默认选项），并且不要在开发 新功能的时候使用其它尚未完成的功能。

这样还有一个好处，那就是多个人合作的时候，你们可以专注于完成自己的那个功能，而互不冲突。

> *“每次只专心做一件事并不意味着你很笨，恰恰相反这是极聪明的做法”*

我觉得游戏的功能差不多可以算做一个阶段，可以导出来给别人试玩啦

非常好，现在要做的很简单：

1. 点击Git工作流，选择“建立新的发布版本”
2. 给你这个版本取个名字，比如“又蹦又跳还有敌人”，然后确定
3. 回去C2修改下你的版本号，还有说明信息什么的，保存！
4. 到ST里保存修改提交
5. 点击Git工作流“完成发布版本”
6. 在“此信息的标签”里面写上一些对于这个版本的说明，可以不写但是建议还是写上吧
7. 确认

完成之后，你会发现库的左边栏“标签”下面出现了你刚刚发布的那个版本，而库本身又自动回归到了develop状态。

但是不碍事，因为现在的develop和master还有发布的那个标签都是完完全全一样的。你可以回到C2去发布游戏啦～

上传到网络上，以免项目因为电脑故障之类的原因而丢失

有很多的版本库保存完站，其中也有很多免费的。下面我推荐几个：

- Github 最具知名度，但是私有库是要付费的
- BitBucket 仅次于Github的流行程度，私有库也免费哦
- Gitlab 这东西是开源的，而且也提供免费的私有库，你还可以安装到自己的服务器上去

如果你想保存一些公开的项目首选Github，自己私有的游戏的话，如果不想付费就选另外两个吧。 Bitbucket拥有很强大的其他服务，非常适合很多人以及大型项目，但速度稍微有些慢。Gitlab 是最年轻的成员，用起来可能有些生涩，但以后的潜力很大。最主要是可以自己安装进自己的服务器。

个人推荐使用Github，非常好用多人合作也非常容易，一个月只要$5。但是如果怕麻烦，可以直接用 Bitbucket，原因很简单，ST就是他们开发的你要下载来用就已经有了他们的账号。登陆之后

新建repo， 然后页面上有一个地址，类似`ssh://bitbucket.org/hello/my_proj.git`，你把它复制下来， 然后打开ST > 仓库 > 项目设置 > 添加。

1. 远端名称建议取做“origin”，方便多人合作。
2. 把地址粘贴进URL栏
3. 确定
4. 点击ST界面上端的“获取”，也就是和服务器同步看看
5. 点击ST界面上端的“推送”，也就是上传

要注意的只有一件事：每次上传之前先“获取”，如果多人合作的话，获取之后可能会显示网络上有新条目可以下载，这时候再点“拉取”。确保和网络同步之后再“推送”。这点很重要哦！

忽略那些不需要保存的文件

最后要说的是，记得忽略掉那些不用保存或者不想保留下来的文件，例如C2自己的backup1、backup2...。 当你看到“未暂存”窗口里出现了这些文件，只需要做一件事，选中之后右击忽略。