

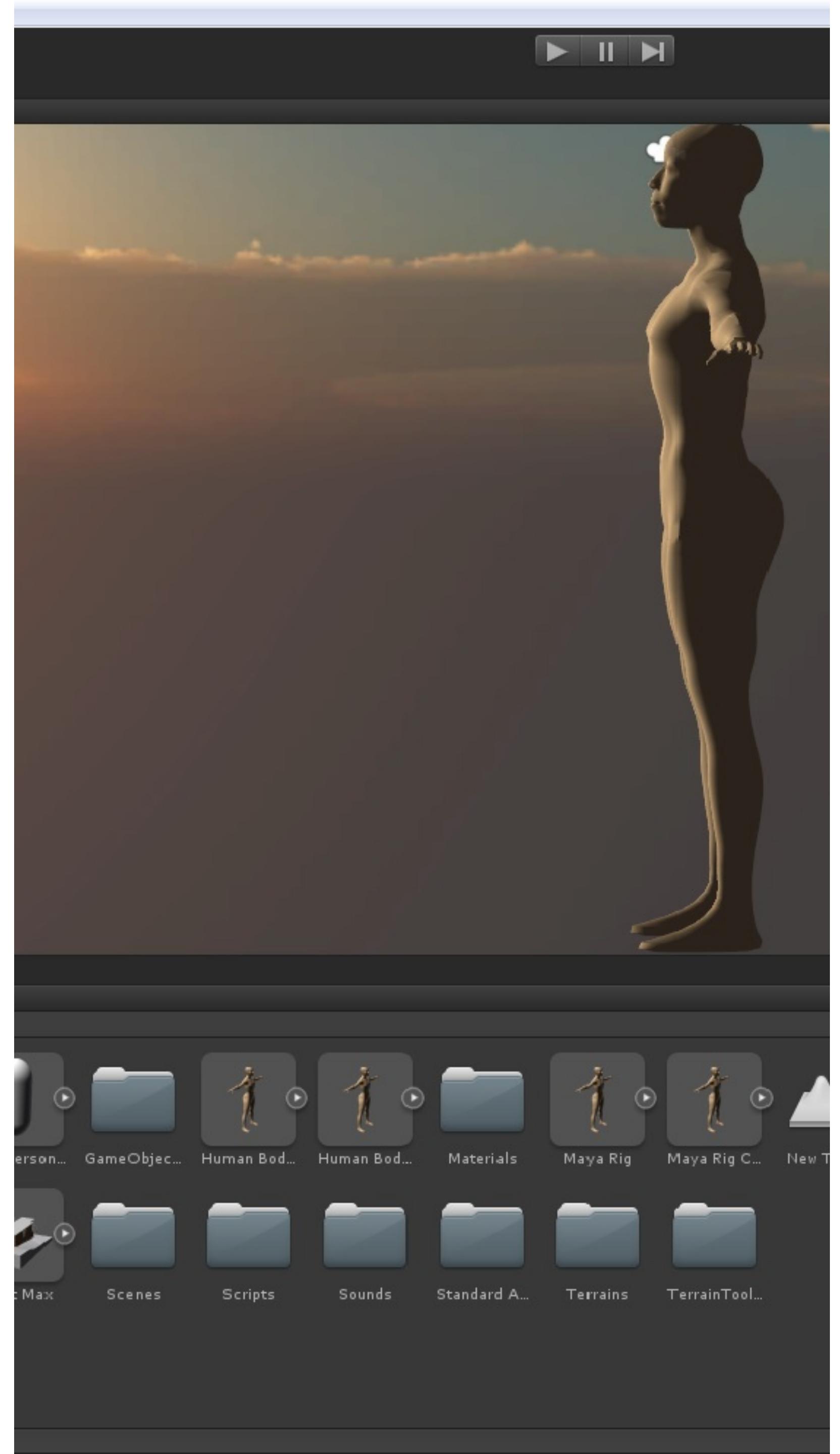
UNITY 5.X

&

GIT

.....

How to Git and workflows



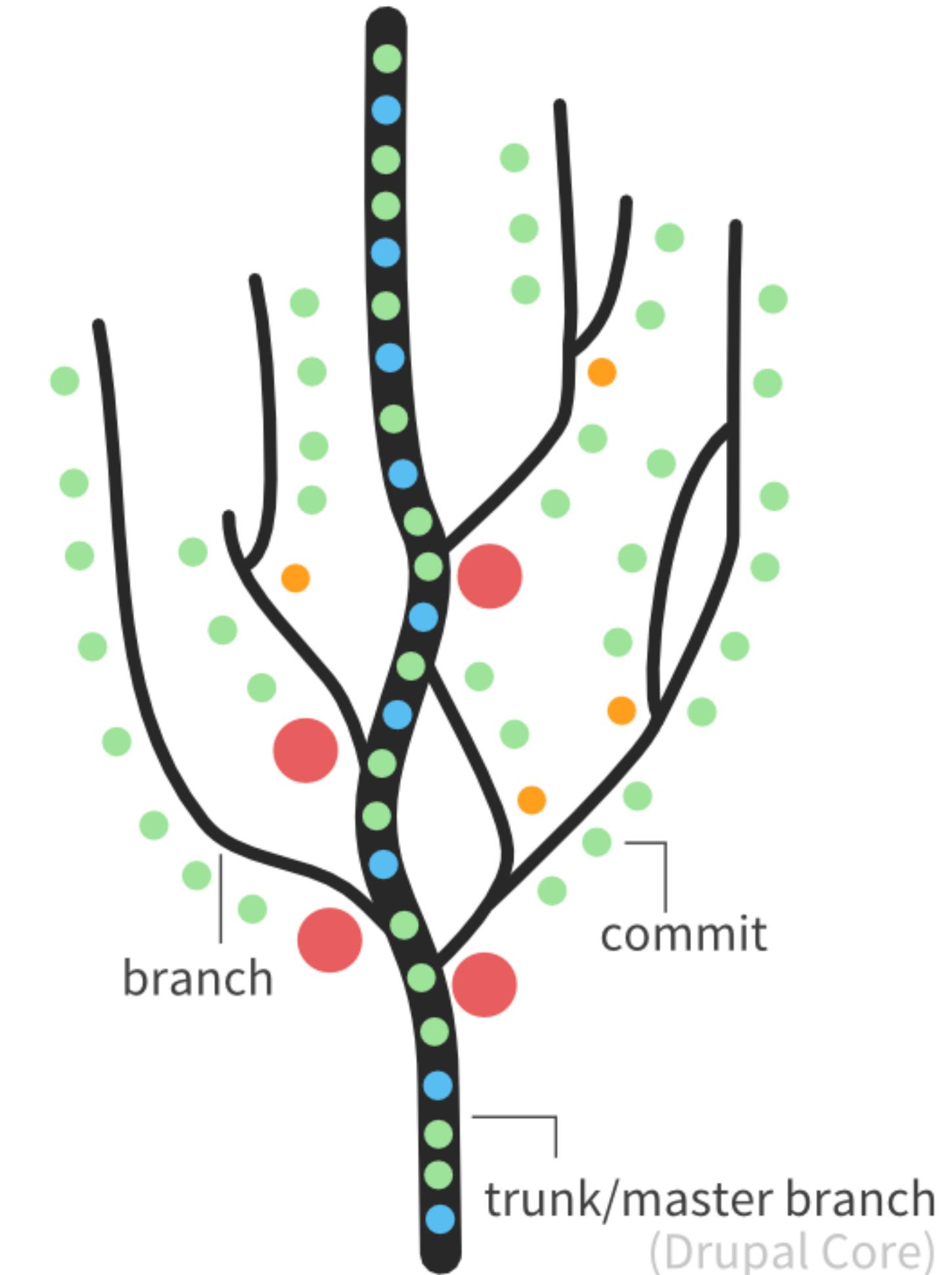
WHAT IS GIT

- A system to control versions of a project
- Enables multiple people to work on files without affecting each other
- Does this by using branches. Branches are independent of each other
- Pull-requests for review and approval before implementation



WHAT IS A BRANCH

- Each branch in a repository represents an independent development path
- Most other Git functions operate using the branch in your working copy
- Branches are used for any change that you want to be implemented into the game. Whether it is a major feature or a single bug fix, every change requires a different branch that most pass through a review process before it is fully implemented.



ON YOUR COMPUTER

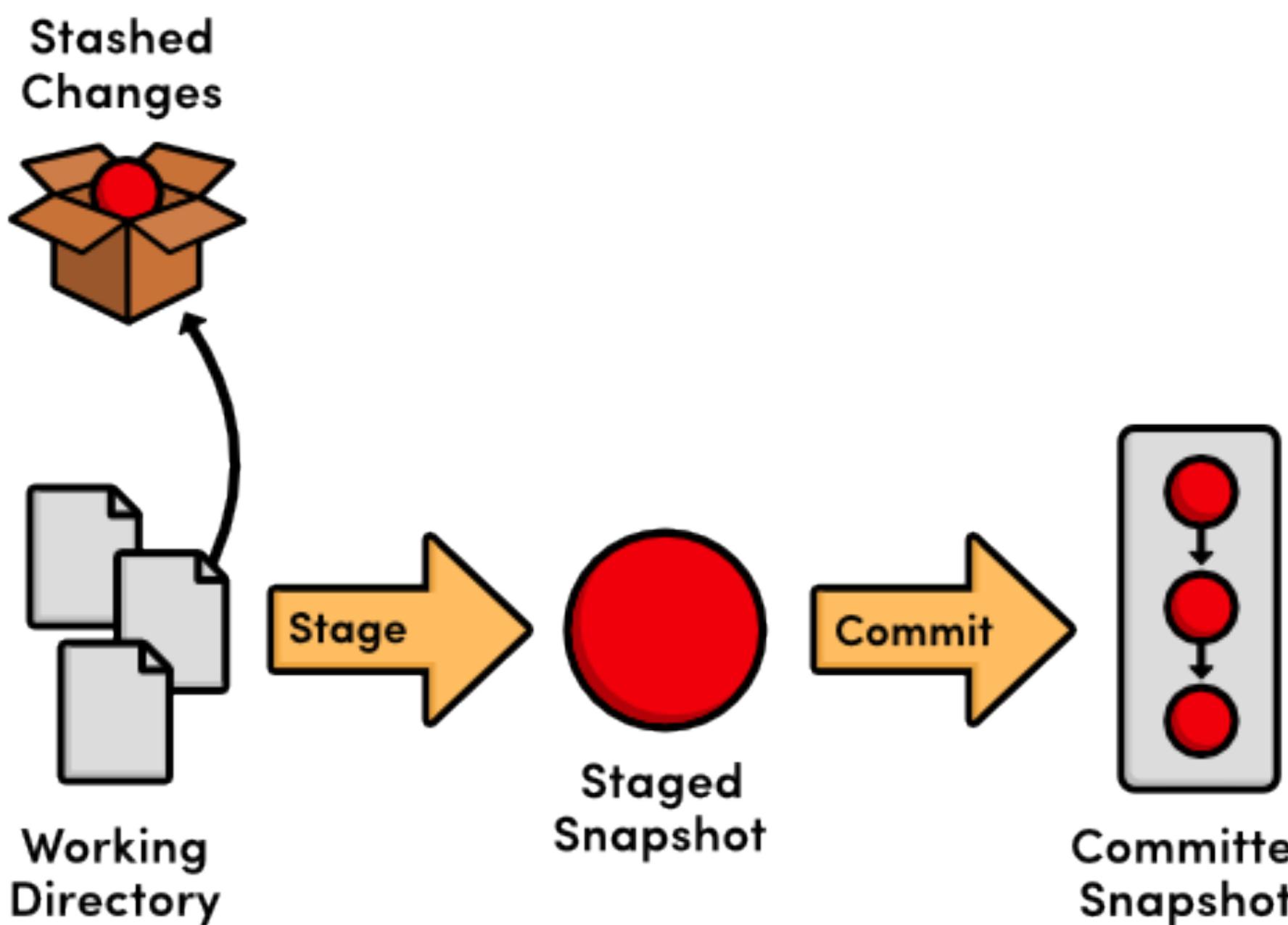
- Working copy is the repository directory on your computer and all its contents
- Git switches between branches by deleting your working copy and replacing it with the selected branch
- If you have uncommitted changes Git will not allow you to switch branches (stashes are a work around)
- Tracked files are files that are currently being used on the remote. Files can be on your computer that aren't actually in the repository

GIT COMMANDS

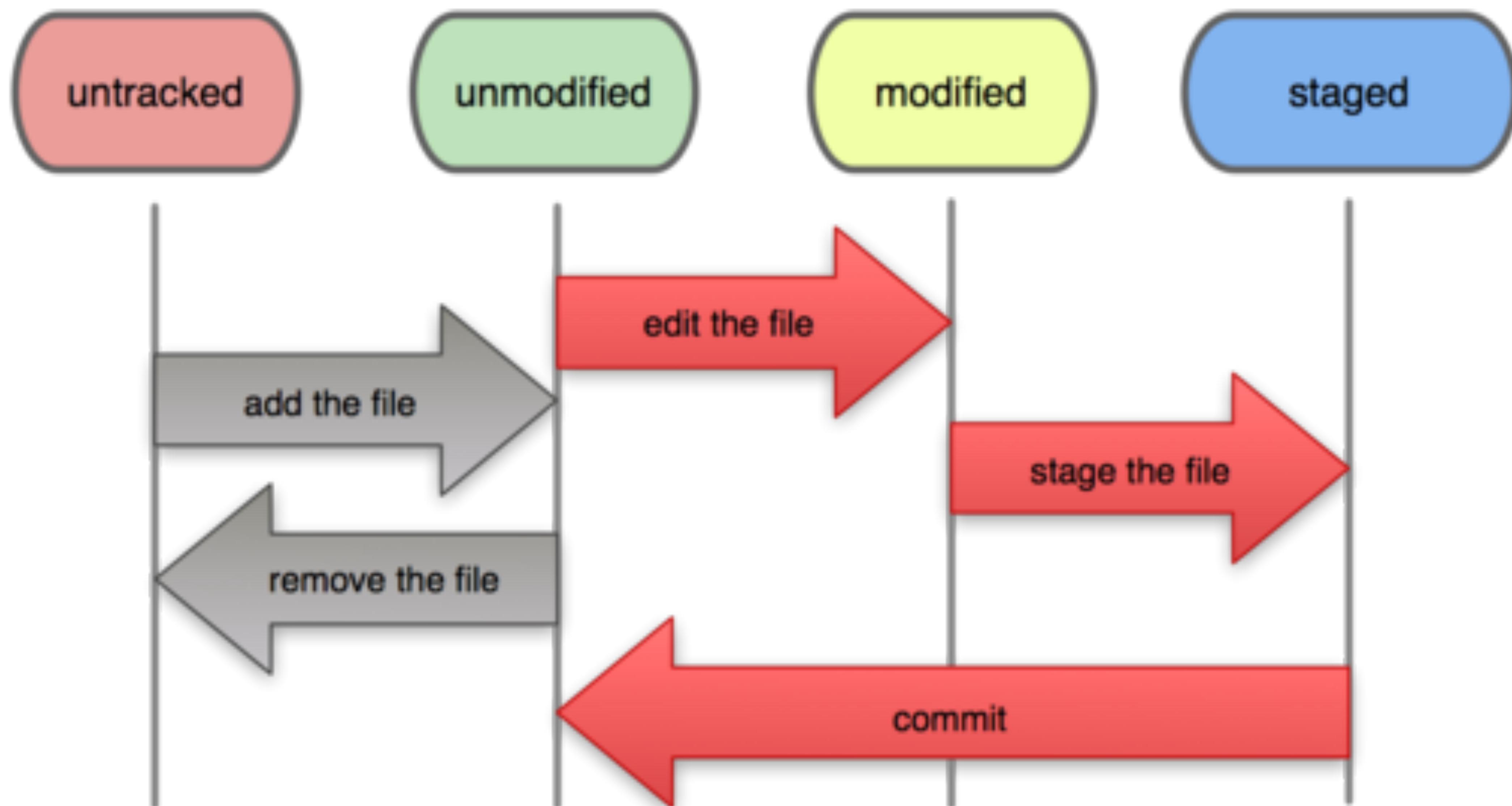
- Commit - The equivalent of a save. Commits will save the staged files with a comment. This will not be sent to the remote until a push.
- Push - Sends changes of a branch to the remote.
- Pull - Replaces your out of date working copy with the latest version of it from the remote
- Checkout - Adds a previously unknown branch to the working copy
- Fetch - Checks the remote to see if any of the branches on your working copy are out of date

GIT COMMANDS

- Stash - A temporary storage of the working copy. Useful for switching branches or applying changes from one branch to another as it doesn't require a commit.
- Reset - Deletes your working copy and replaces it with the remote of the branch

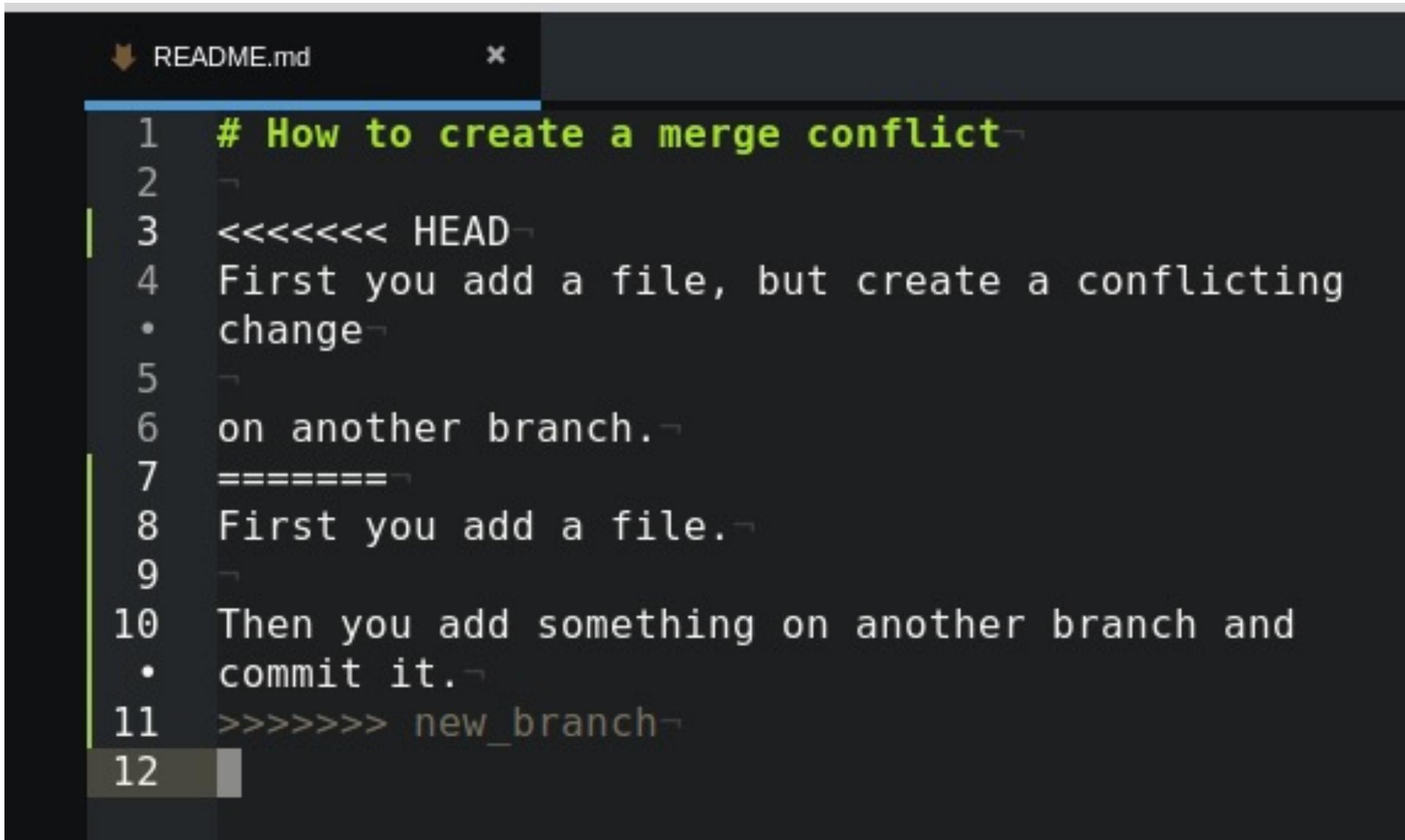


File Status Lifecycle



MERGE

- Merging is used to merge one branch in to another. This will essentially combine both branches and their functionality.
- Merging can create conflicts when the same lines data in a file has been edited by you and edited on the targeted merge branch.



A screenshot of a terminal window titled "README.md". The file content shows a merge conflict between two branches. The conflict is indicated by markers: '<<<<< HEAD' on line 3 and '>>>>> new_branch' on line 11. The text describes the steps to create a merge conflict, mentioning adding a file, creating a conflicting change, and then adding something else on another branch and committing it.

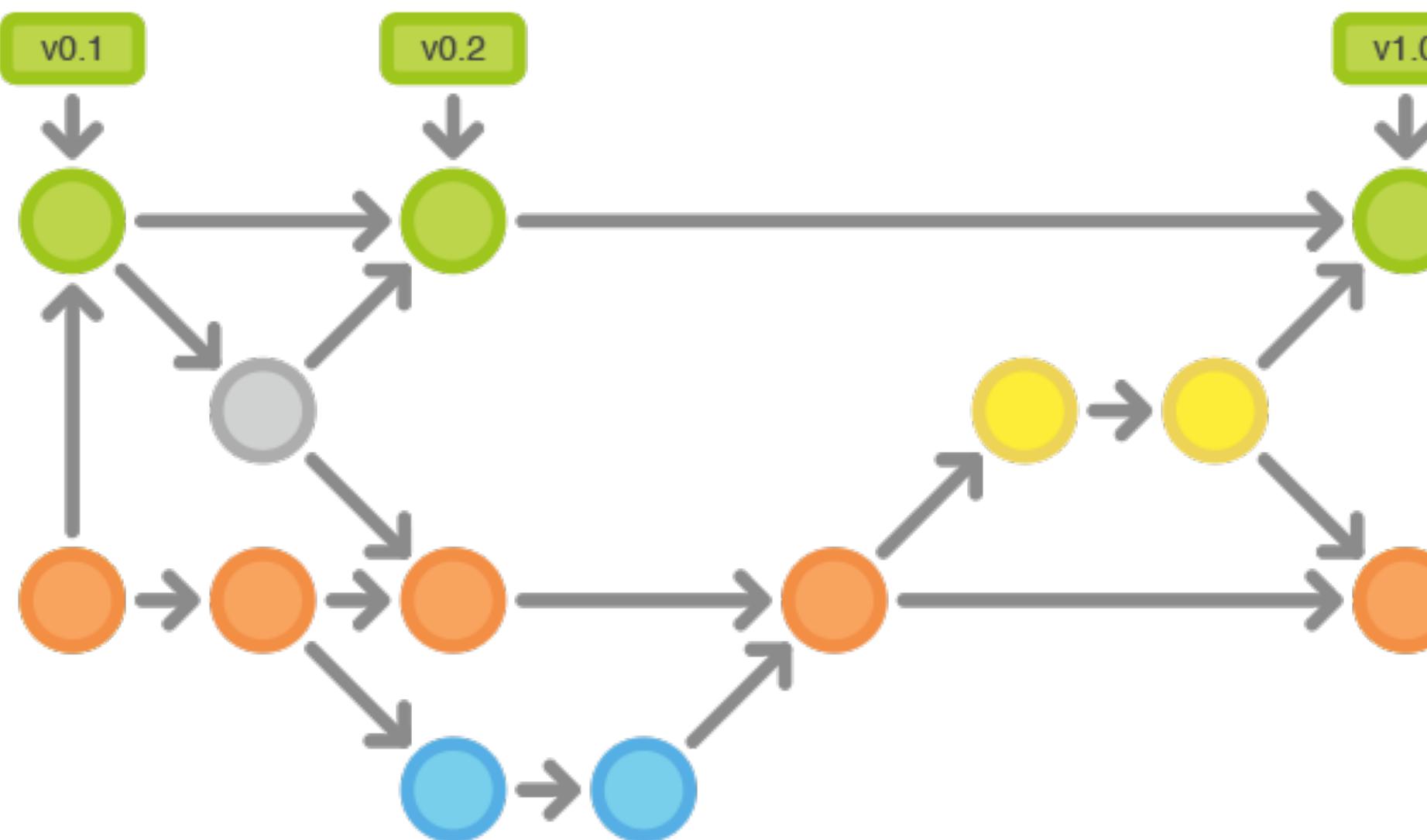
```
1 # How to create a merge conflict
2
3 <<<<< HEAD
4 First you add a file, but create a conflicting
• change
5
6 on another branch.
7 =====
8 First you add a file.
9
10 Then you add something on another branch and
• commit it.
11 >>>>> new_branch
12
```

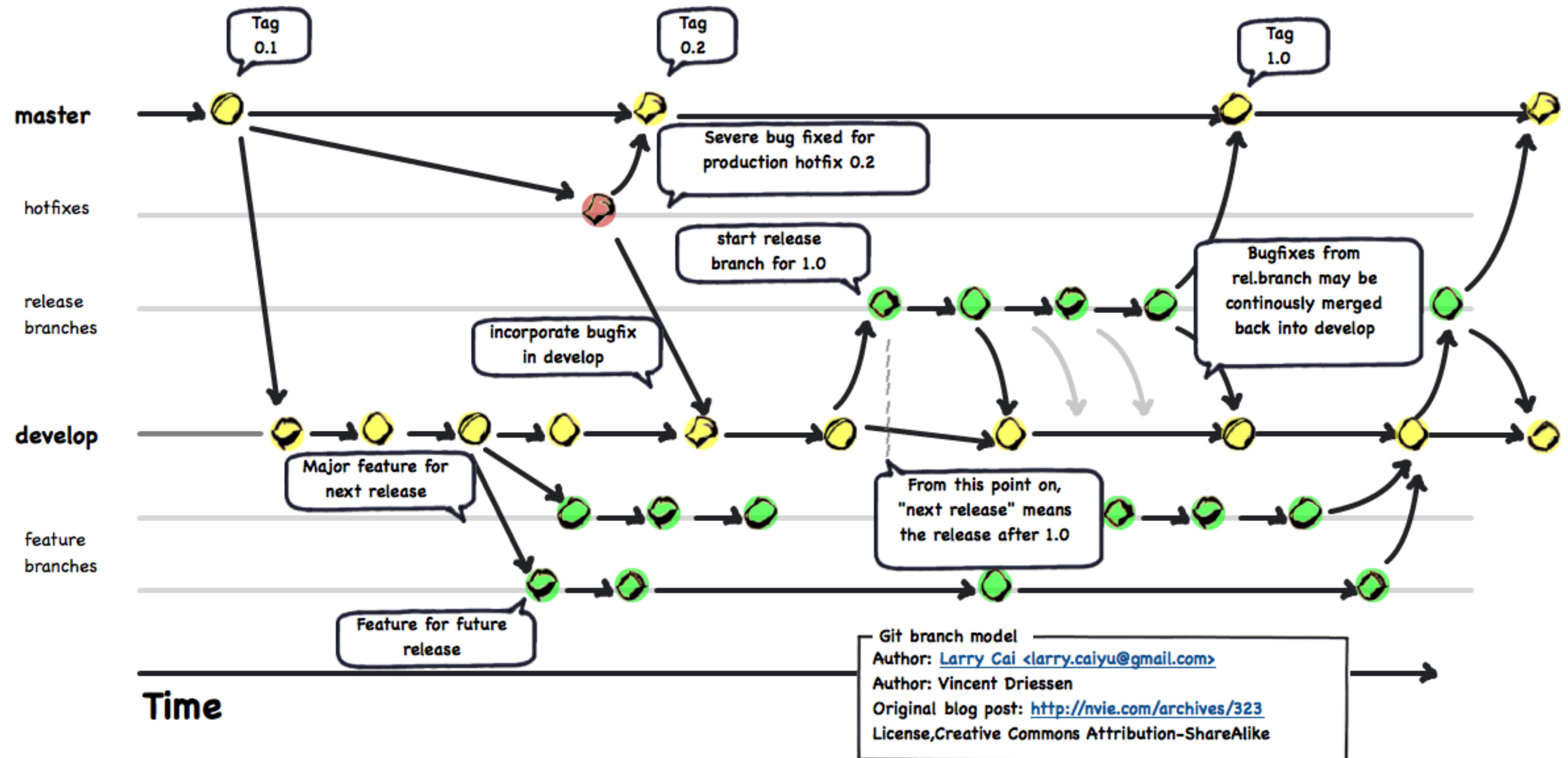
BRANCHES

- master - The branch that represents the version of the project which is released. It will be tagged with a version for every commit made. Only directly editable by an admin.
- develop - The branch where all the features combine and represents what will be eventually turned into a release. Only admins can edit it directly but developers can through pull-requests.
- feature/feature-name - Branch which represents edits that's want to be implemented into the develop branch. Editable by everyone on the project. This is where most of the development actually happens.

BRANCHES

- release-1.13.1 - The intermediate branch between the develop and master branches. Used to prepare the project for release by fixing bugs and building artifacts.
 - hotfix - Branch that becomes active if a patch needs to be issued before the next release. Copies the master and applies the fixes before merging back in with a new version tag.



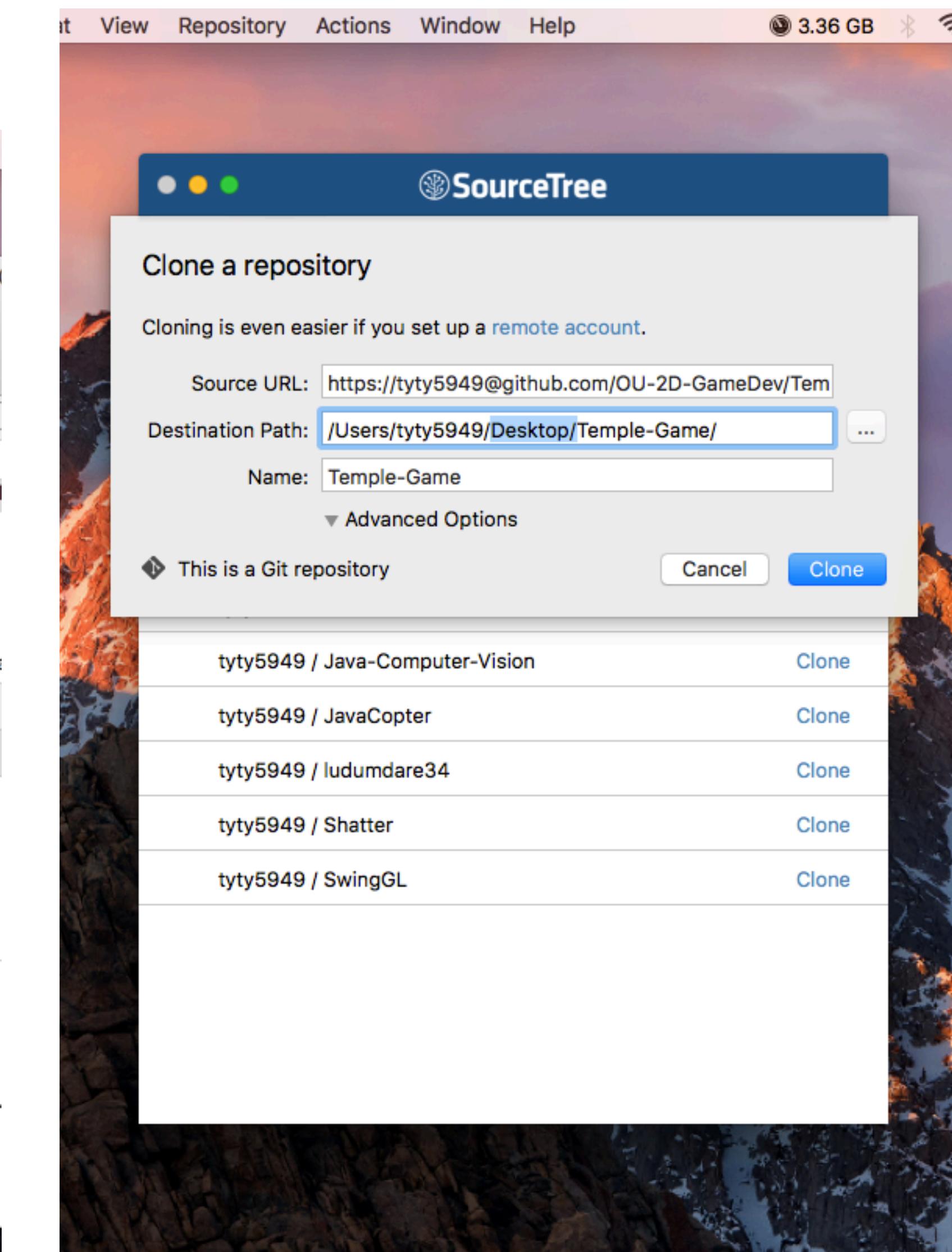
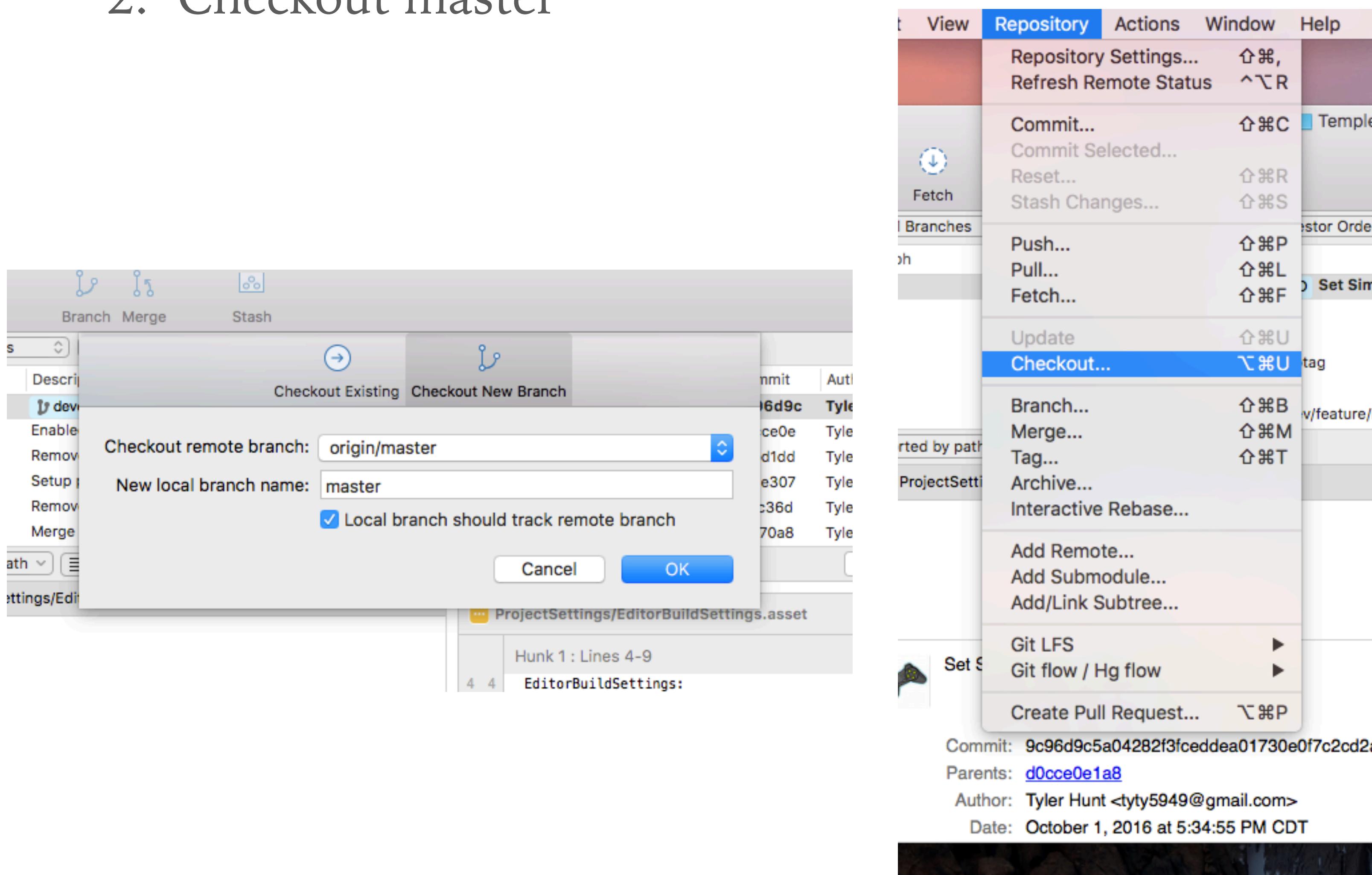


WORKFLOW

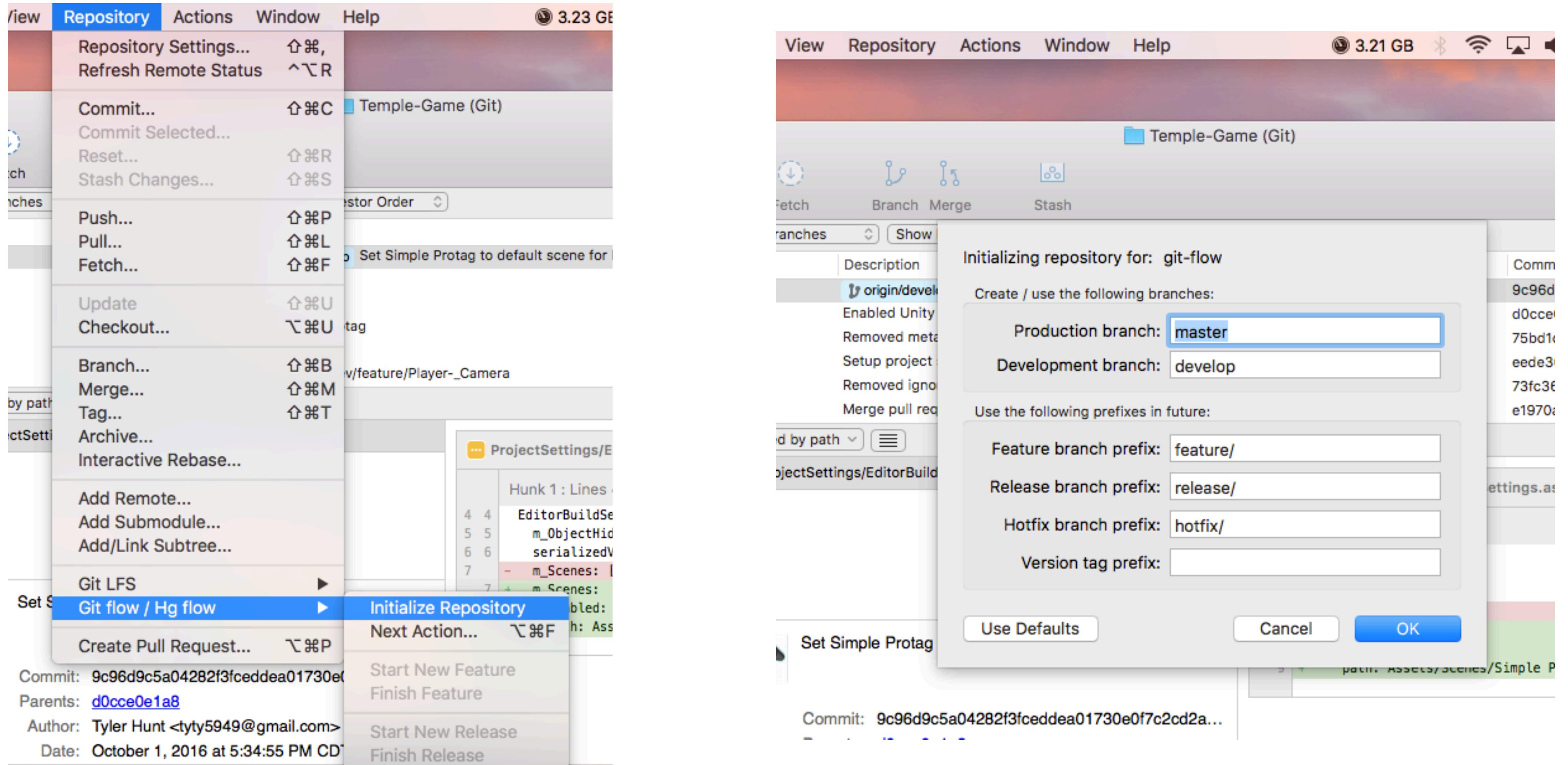
1. Clone into repository
2. Initialize Git Flow
3. Create feature branch
4. Make changes
5. Commit and push changes
6. (When ready to add feature to game)
7. Merge from develop and fix conflicts
8. Create pull-request

CLONING IN SOURCE TREE

1. Clone the repository (Note: The directory)
2. Checkout master

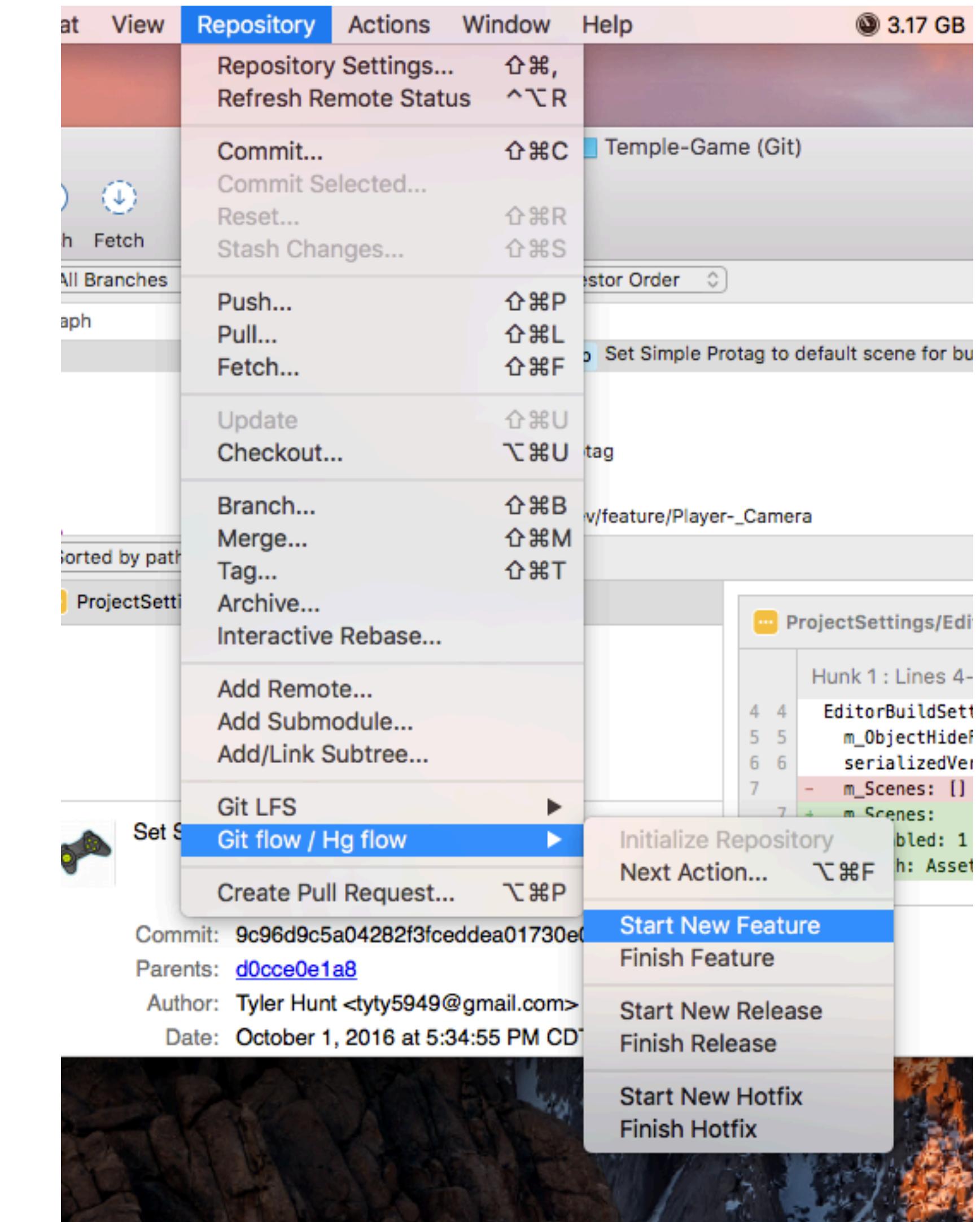
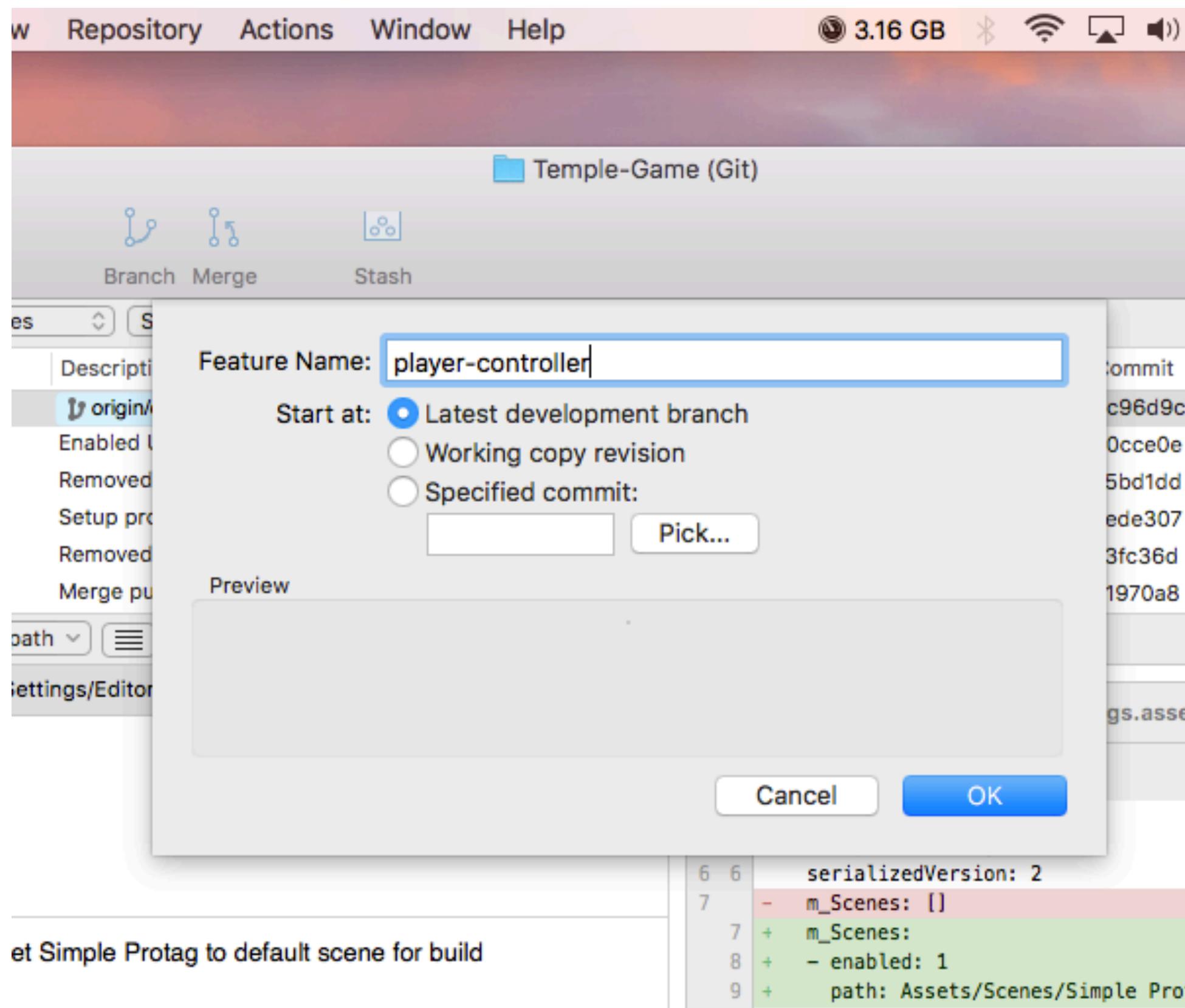
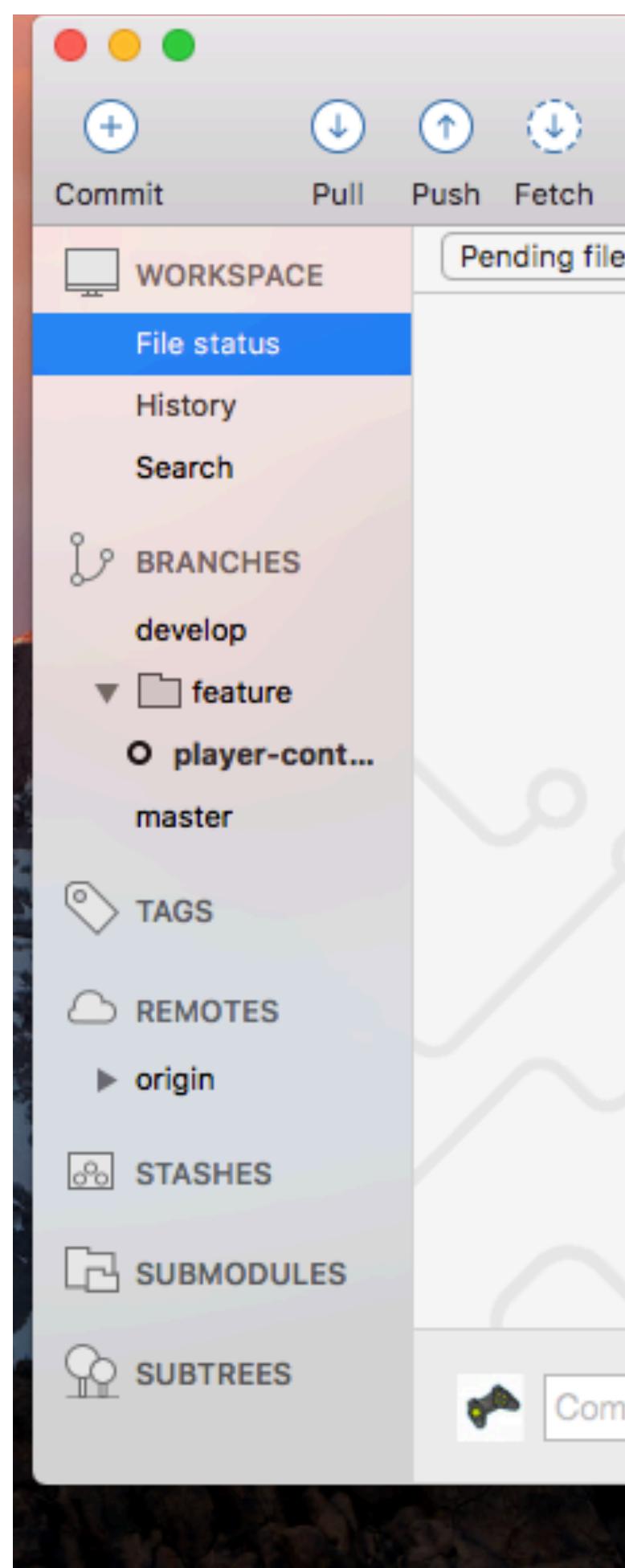


INITIALIZE GITFLOW IN SOURCE TREE



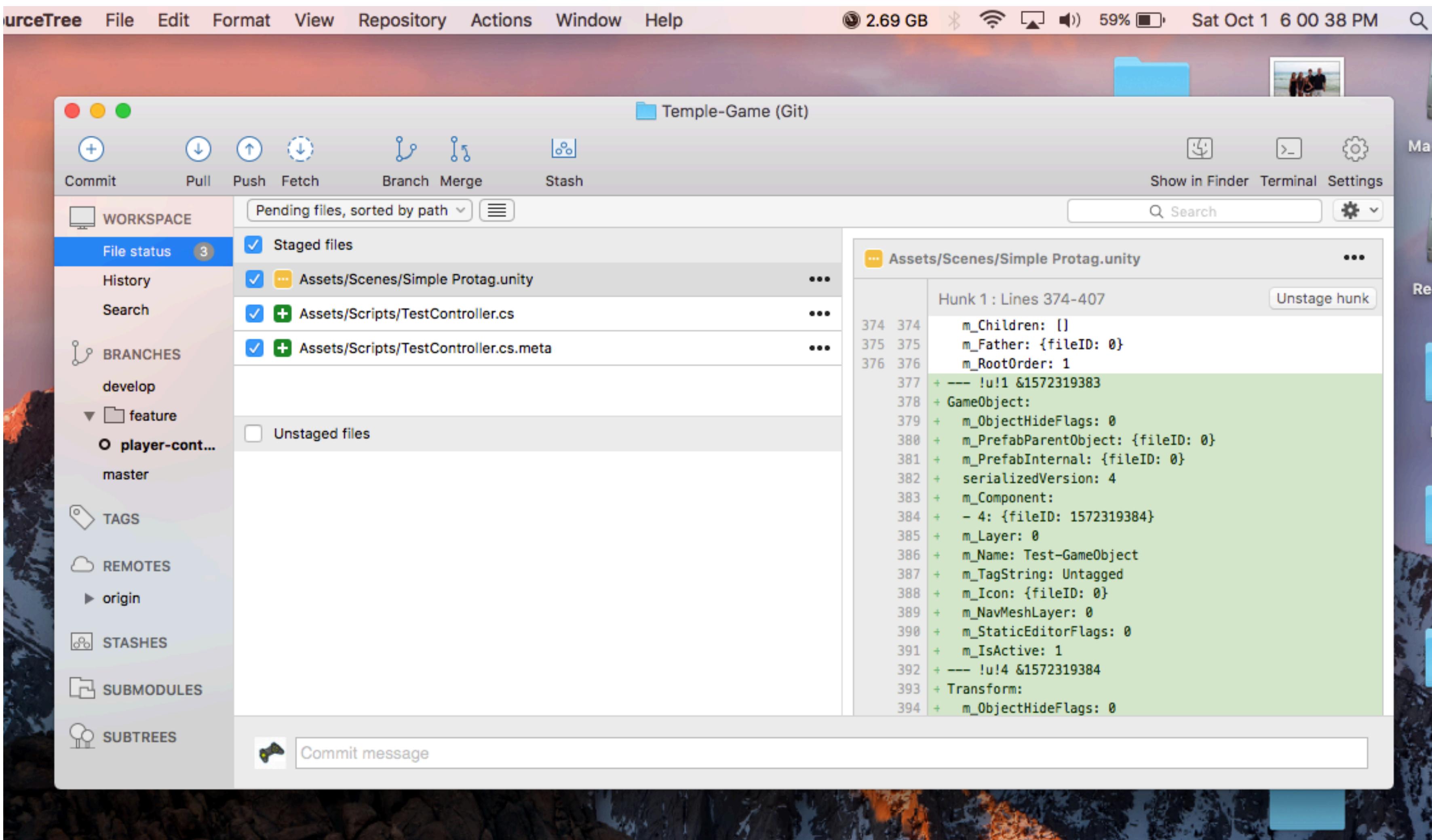
CREATE BRANCH IN SOURCE TREE

- Don't use spaces, all lowercase



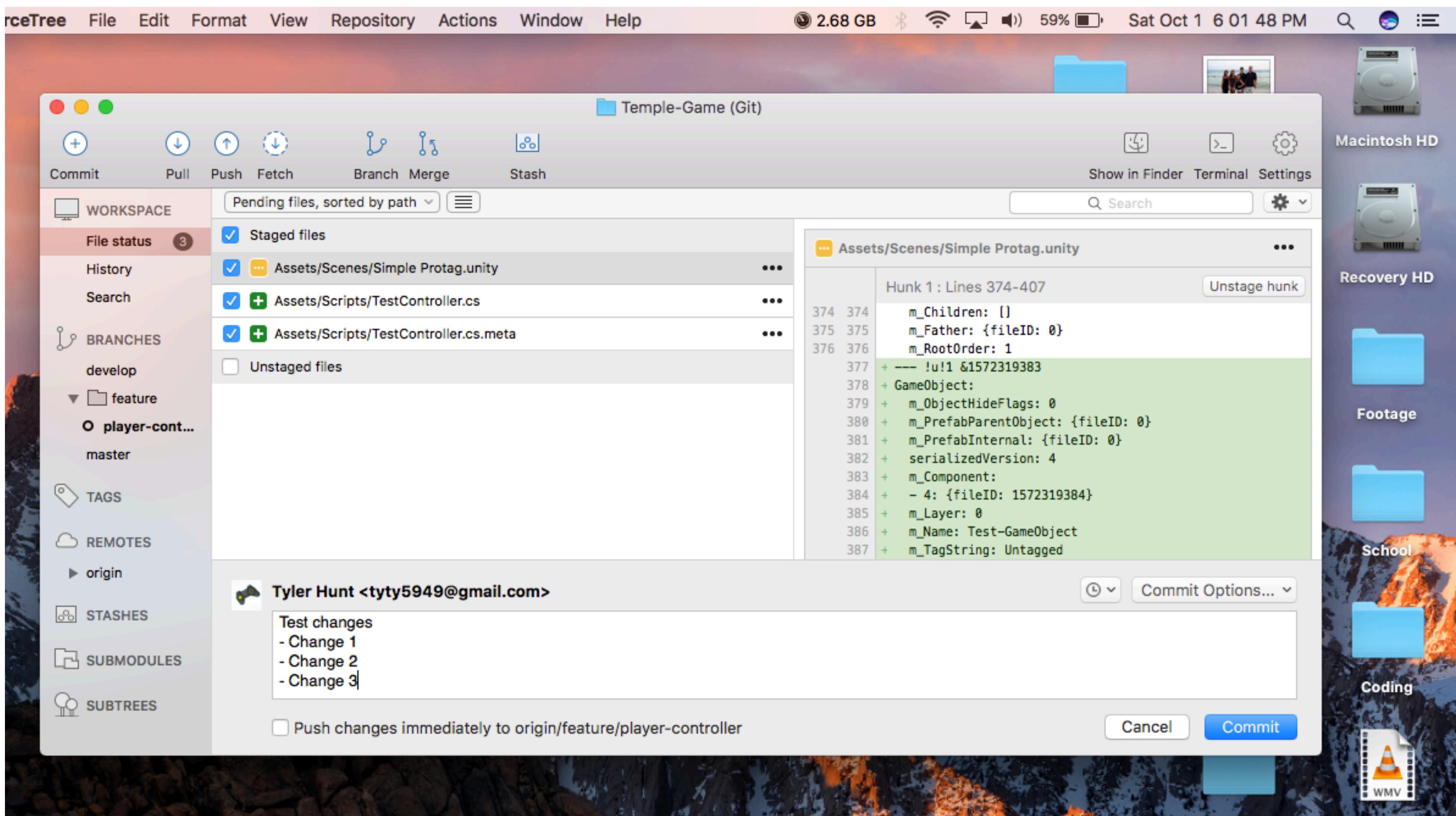
STAGE CHANGES IN SOURCE TREE

.....



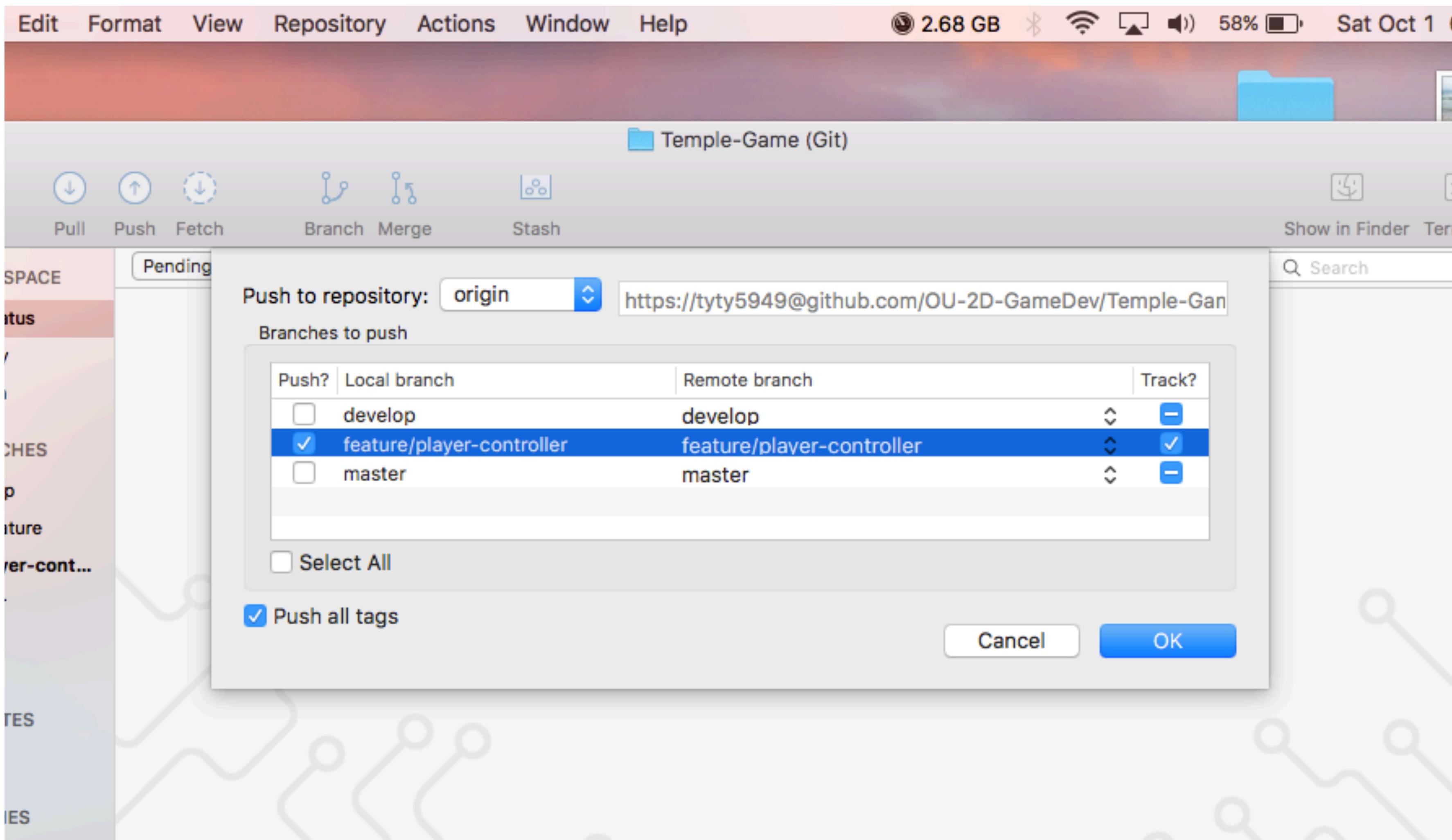
COMMIT IN SOURCE TREE

.....



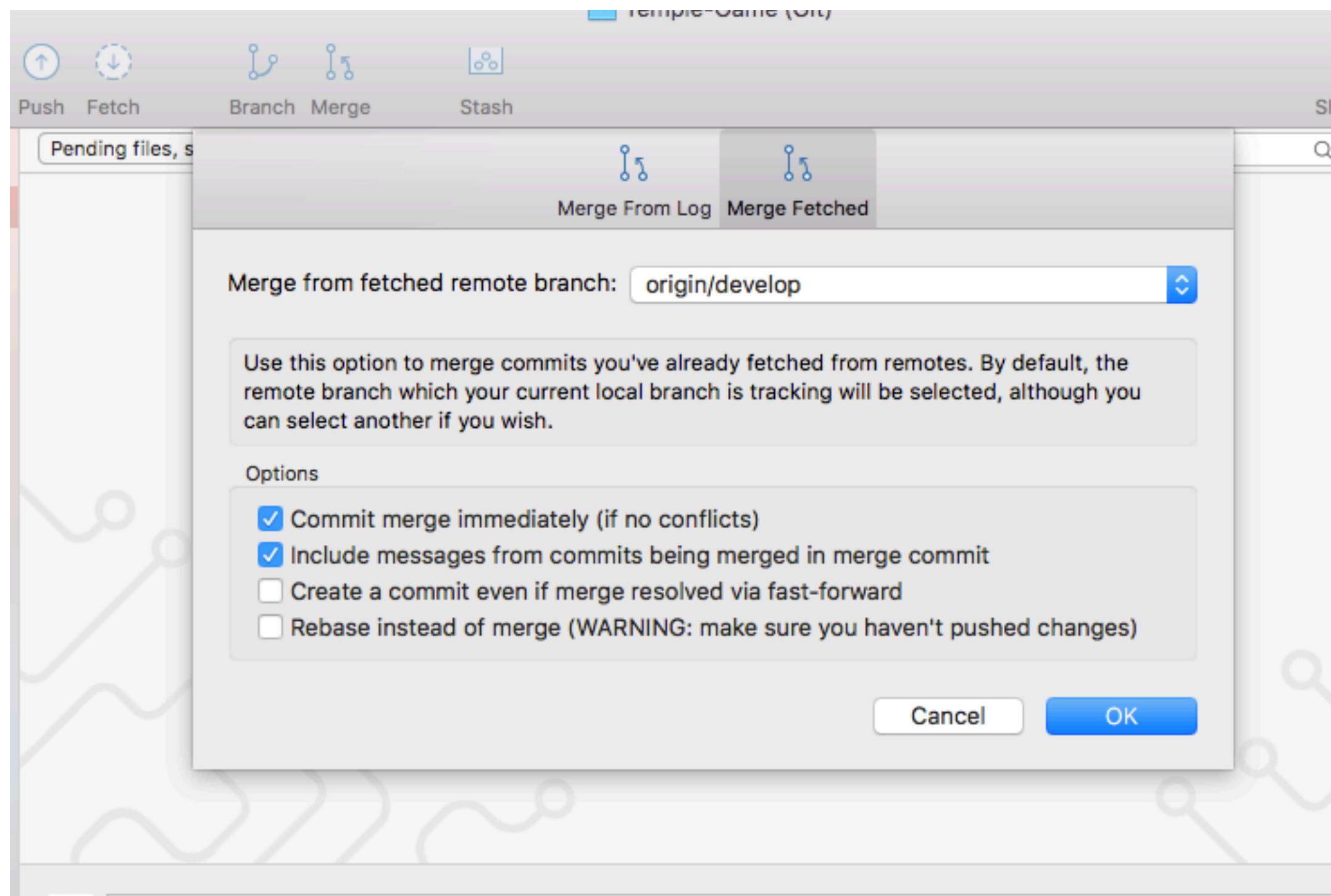
PUSH IS SOURCE TREE

.....



MERGE FROM DEVELOP

- If you need a feature that has already been merged into develop, or to test if your patch is compatible with the branch



- Conflicts will have to be resolved before the merge can be committed. The merge changes WILL be reflected in your working copy.

FIXING CONFLICTS

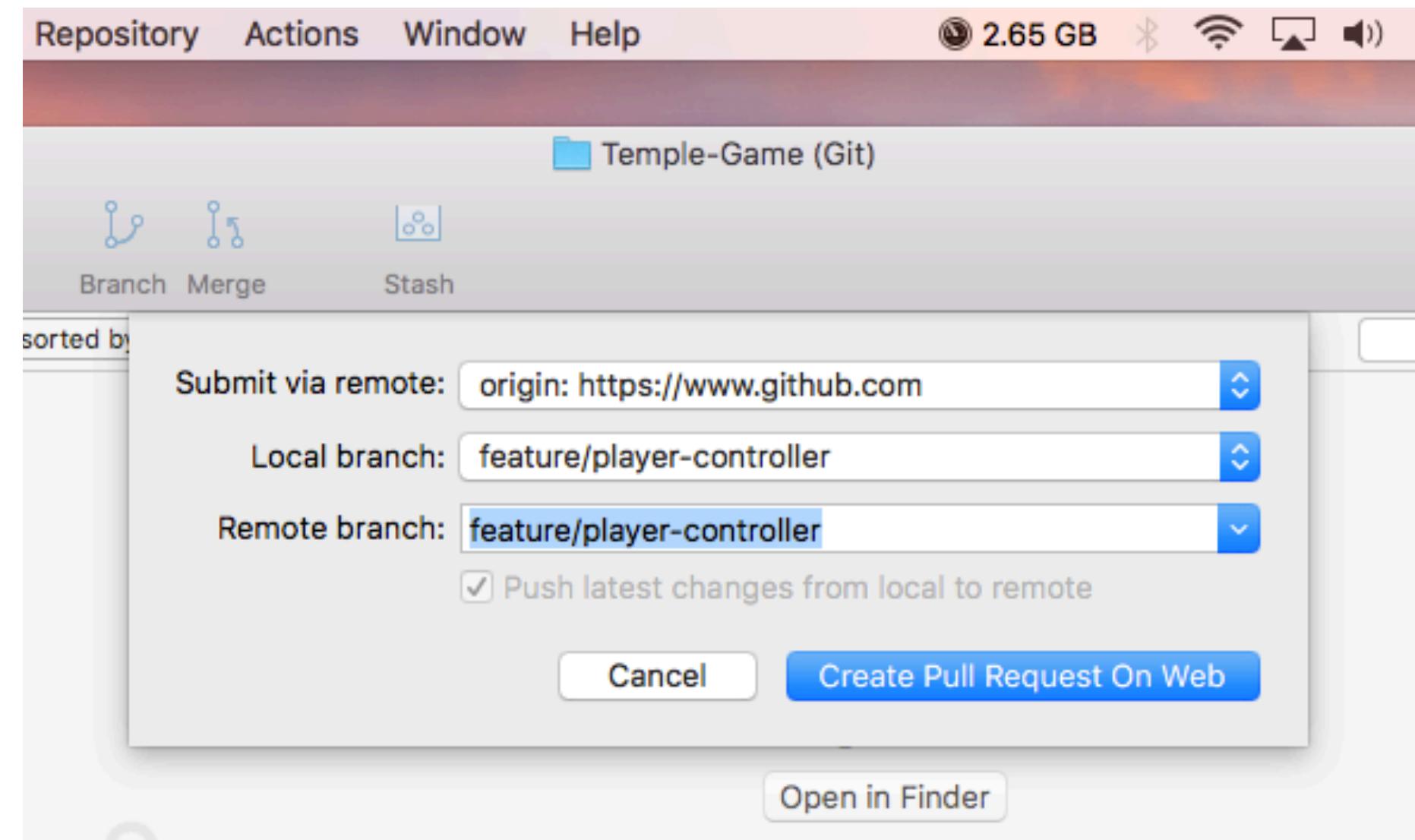
- SourceTree will indicate what files have conflicts
- Manually edit files and choose the line of code to resolve the conflict
- Stage the edited files, then commit and push

The screenshot shows a Java code editor window with the file "SimpleApplication.java" open. The code defines a class "SimpleApplication" with three methods: main, uiFunction, and addSecurity. The code includes comments and some conflict markers. Lines 10 and 14 are marked with a red 'X' and show markers for 'HEAD' and 'refs/heads/new_idea'. Lines 18 and 26 are also marked with a red 'X' and show similar markers. The code is as follows:

```
3 public class SimpleApplication {  
4     /**  
5      * @param args  
6      */  
7     public static void main(String[] args) {  
8         // TODO Auto-generated method stub  
9         <<<<< HEAD  
10        // Fix a core bug  
11        =====  
12        // Alternate core fix  
13        >>>>> refs/heads/new_idea  
14    }  
15  
16    public static void uiFunction() {  
17        <<<<< HEAD  
18        // Fix a UI bug  
19        }  
20  
21    public static void addSecurity() {  
22        // Adds the needed security  
23        =====  
24        // Alternate UI fix  
25        >>>>> refs/heads/new_idea  
26    }  
27  
28  
29}
```

GITHUB PULL-REQUEST

.....



A screenshot of the GitHub web interface for the repository "OU-2D-GameDev / Temple-Game". The top navigation bar includes "Code", "Issues 0", "Pull requests 0", "Projects 1", "Wiki", "Pulse", "Graphs", and "Settings". The main content area is titled "Open a pull request" with the sub-instruction "Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#)". Below this, there are dropdown menus for "base: develop" and "compare: feature/player-controller", with a green checkmark indicating "Able to merge. These branches can be automatically merged". A "Test changes" section contains a "Write" tab and a preview of the pull request content, which lists "- Change 1", "- Change 2", and "- Change 3". The right sidebar shows "Labels: None yet", "Milestone: No milestone", and "Assignee: No one—". At the bottom are "Create pull request" and "Styling with Markdown is supported" buttons.

PULL-REQUESTS

- Used to review and recommend changes before they go live on the develop branch
- At least one person must approve before the branch can be merged. IN THEORY, everyone reviews the code and either makes comments or approves.
- Notification via Slack that a pull-request has been made. As soon as possible, either comment, or approve the request.

COMMENTING

.....

The screenshot shows a GitHub pull request interface. The top navigation bar includes tabs for 'Branches - OU-2D-GameD...', 'GitHub Help', '# general | OU 2D GameDe...', 'Test changes by tyty5949 ...', and a '+' button. The main title bar shows the URL <https://github.com/OU-2D-GameDev/Temple-Game/pull/2/files>. Below the title bar, there are links for 'Most Visited' and 'Getting Started'.

The main content area displays a diff view of a C# script. The top status bar indicates 'Changes from all commits' (3 files), '+55 -0', and a green progress bar. The bottom status bar shows 'Changes from all commits' (3 files) and a blue progress bar.

A modal window is open over the code editor, titled 'Write' (Preview tab selected). It contains the text: 'This is a test comment. Fix this code.' and a note: 'Attach files by dragging & dropping or selecting them.' A Markdown styling note says 'Styling with Markdown is supported'. At the bottom of the modal are 'Cancel', 'Add single comment', and a large green 'Start a review' button.

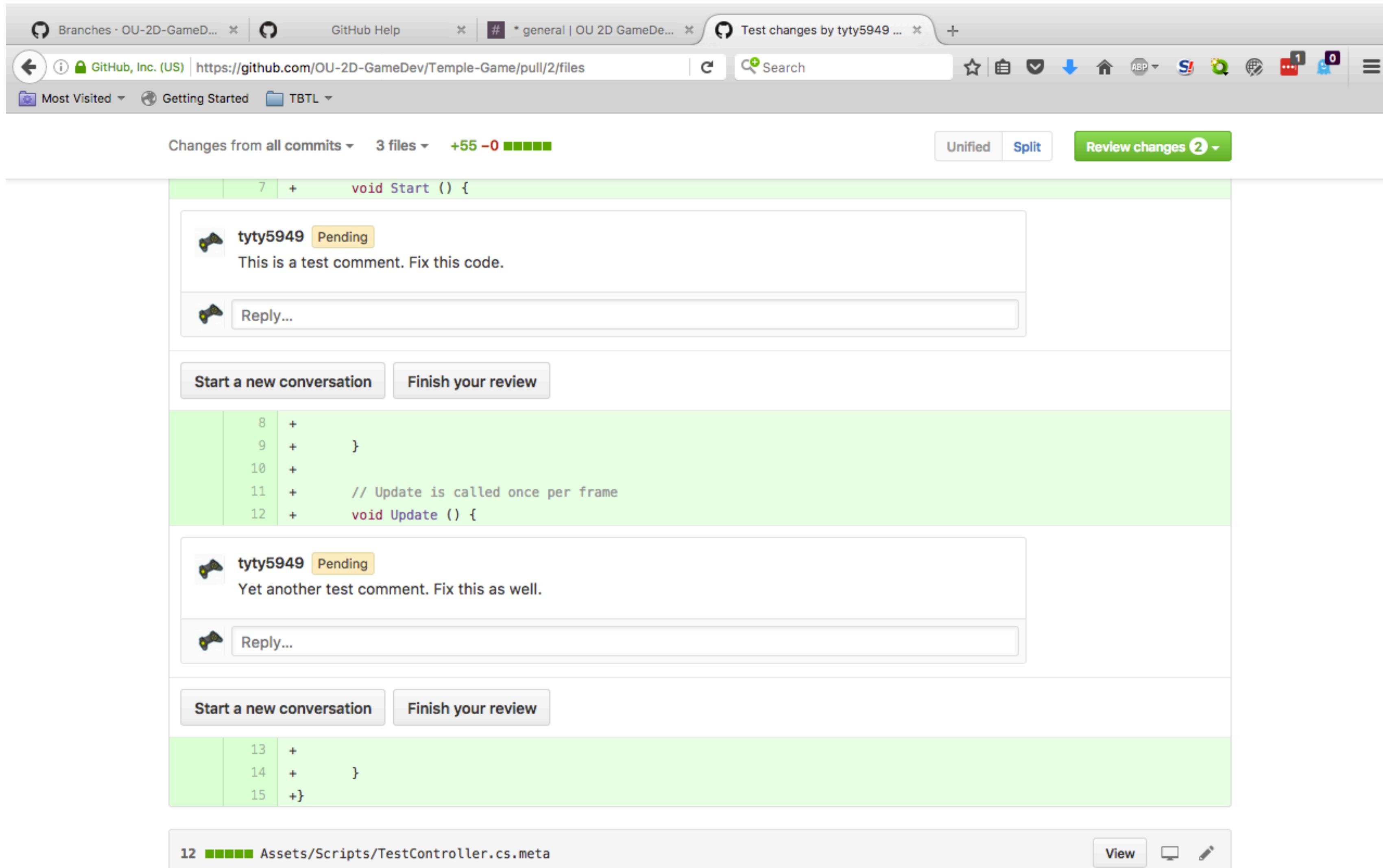
The code editor shows the following C# code:

```
@@ -0,0 +1,15 @@
1  using UnityEngine;
2  using System.Collections;
3 +
4  public class TestController : MonoBehaviour {
5 +
6      // Use this for initialization
7      void Start () {
8          +
9      }
10 +
11     // Update is called once per frame
12     void Update () {
13 +
14     }
15 }
```

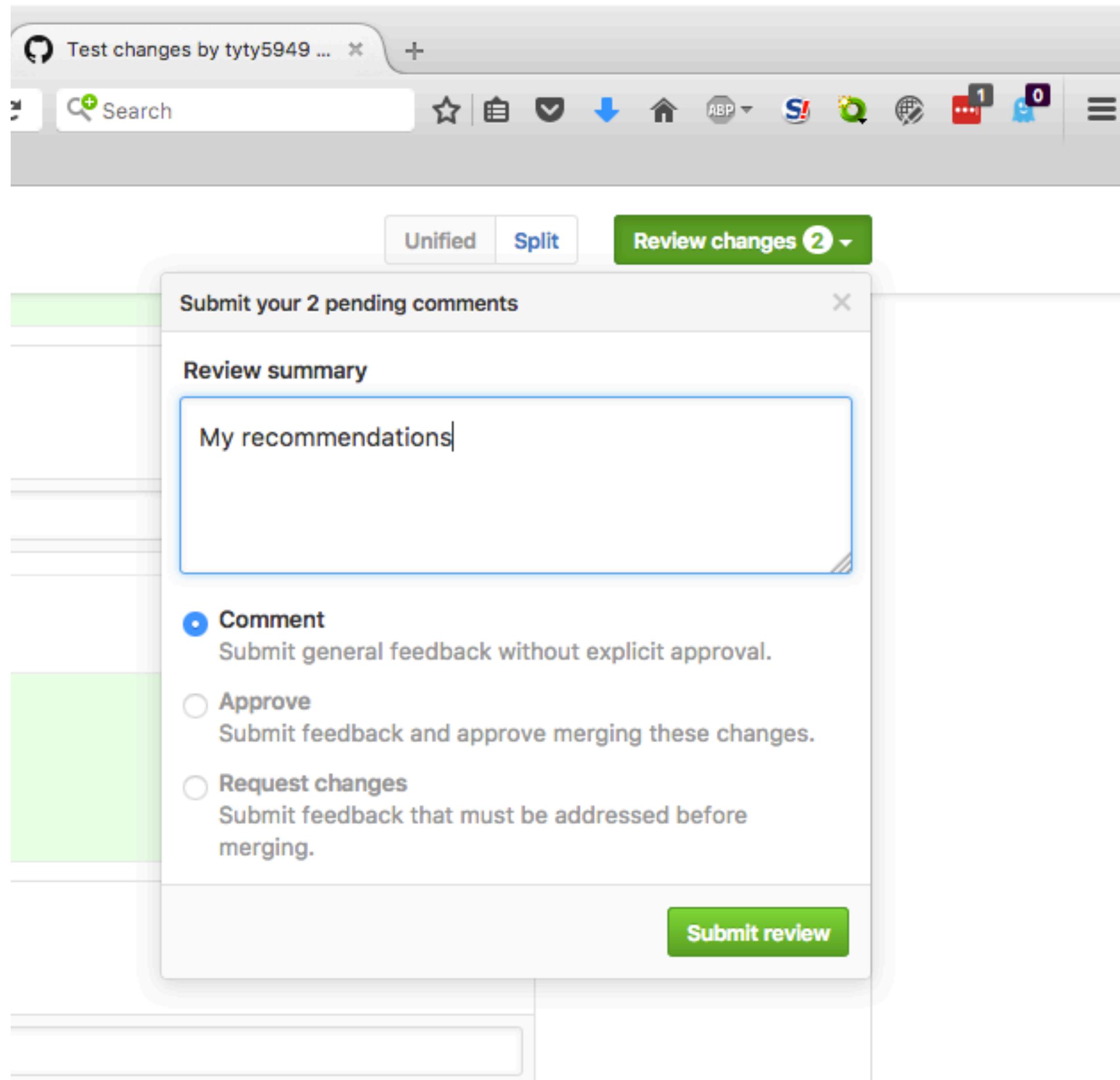
The bottom left corner of the code editor shows the file path 'Assets/Scripts/TestController.cs'.

COMMENTING

.....



COMMENTING



APPROVING A PULL-REQUEST

.....

The screenshot shows a GitHub pull request interface for a repository named "OU 2D GameDev S...". The pull request is titled "Test changes #2" and is from user "tyty5949" into the "develop" branch from the "feature/player-controller" branch. The pull request has 1 commit and 3 files changed.

The main view displays the diff for the file "Assets/Scenes/Simple Protag.unity". The diff shows 28 additions and 0 deletions. The code changes are as follows:

```
@@ -374,6 +374,34 @@ Transform:  
 374 374     m_Children: []  
 375 375     m_Father: {fileID: 0}  
 376 376     m_RootOrder: 1  
 377 +--- !u!1 &1572319383  
 378 +GameObject:  
 379 +    m_ObjectHideFlags: 0  
 380 +    m_PrefabParentObject: {fileID: 0}  
 381 +    m_PrefabInternal: {fileID: 0}  
 382 +    serializedVersion: 4  
 383 +    m_Component:  
 384 +      - 4: {fileID: 1572319384}  
 385 +    m_Layer: 0  
 386 +    m_Name: Test-GameObject  
 387 +    m_TagString: Untagged  
 388 +    m_Icon: {fileID: 0}  
 389 +    m_NavMeshLayer: 0  
 390 +    m_StaticEditorFlags: 0  
 391 +    m_IsActive: 1  
 392 +--- !u!4 &1572319384  
 393 +Transform:
```

A modal window titled "Submit your review" is open, showing a "Review summary" section with a text input field labeled "Leave a comment". Below it are three radio button options: "Comment" (selected), "Approve", and "Request changes". A tooltip for the "Comment" option states: "Pull request authors can't approve their own pull request". The "Approve" option is described as "Submit feedback and approve merging these changes". The "Request changes" option is described as "Submit feedback that must be addressed before merging". A "Submit review" button is at the bottom of the modal.

UNITY WORKFLOW

- Keep workspace organized (folders, naming, readability)
- Write scripts and name things in a way which anyone will be able to understand what they do
- If necessary, create a new scene to test a feature. Try not to modify game levels just to test a feature.
- Scenes created for testing should have the same name as the branch of the feature
- If checking out a feature to test it, you will have to load the scene manually, it will NOT automatically load.