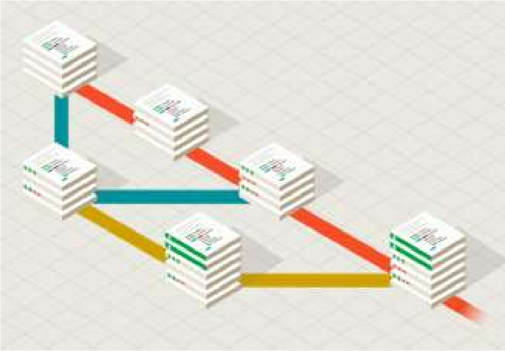
Git branching and merging

We will discuss how to

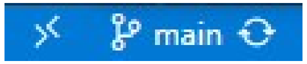
- create feature branches
- use the Git Graph extension in VS Code
- switch to another branch
- merge branches
- reset a branch to an older commit

Create a new feature branch

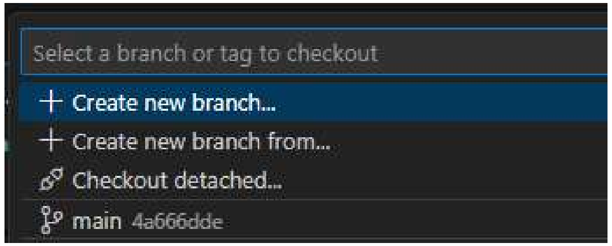
When several developers work on the same project, they can do this more independently using branches, remember the picture from Git web site:



You see the name of the current branch
in the status line of VS Code on bottom left:

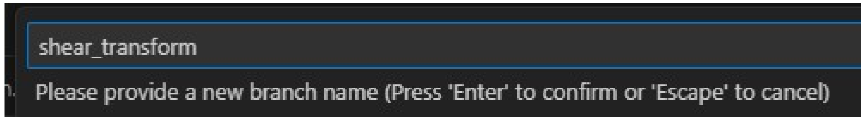


Click on it, then on top of VS Code
the following dialog is opened:



Select a branch or tag to checkout

Click on “Create new branch” and enter its name:

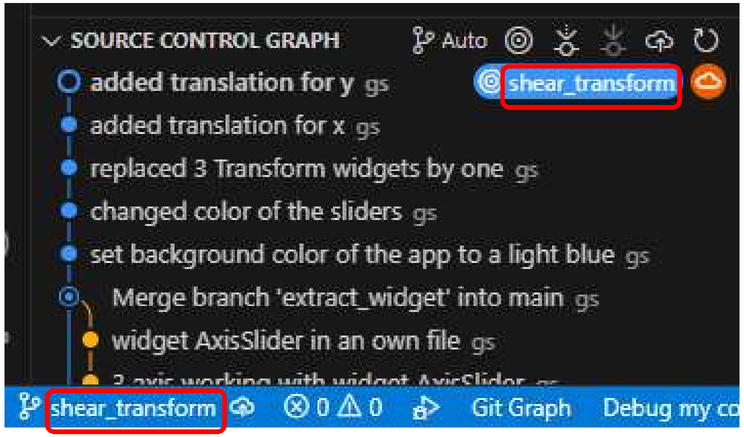


shear_transform

Please provide a new branch name (Press 'Enter' to confirm or 'Escape' to cancel)

Working on new feature branch

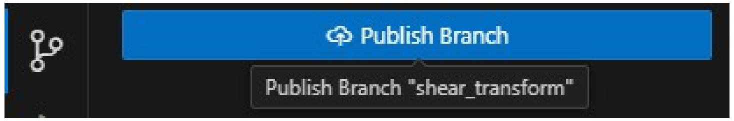
After creation of the new branch, you see it both in the status line and in Source Control Graph:



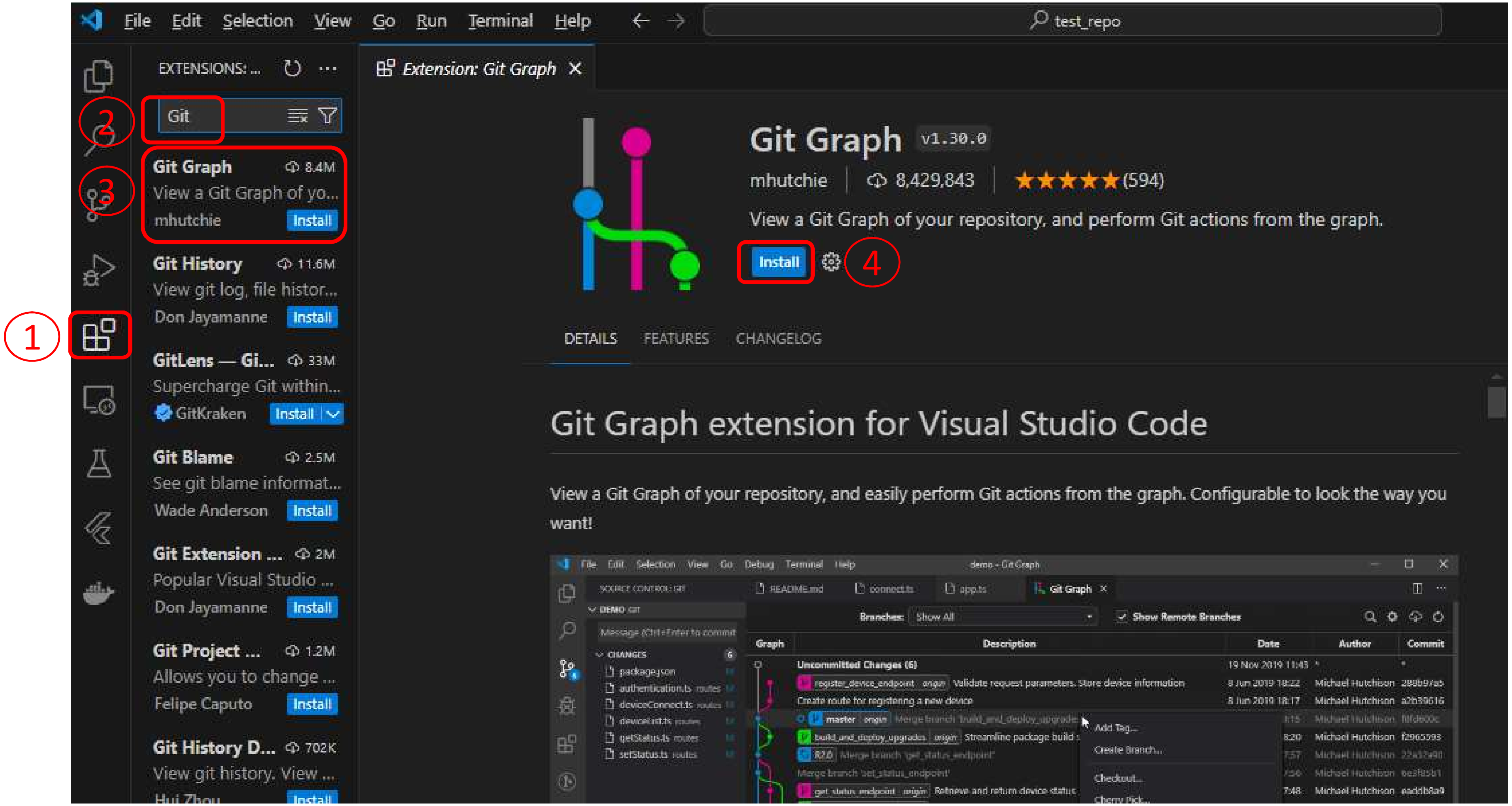
```
added translation for y gs
added translation for x gs
replaced 3 Transform widgets by one gs
changed color of the sliders gs
set background color of the app to a light blue gs
Merge branch 'extract_widget' into main gs
widget AxisSlider in an own file gs
3 axis working with widget AxisSlider gs
```

When you are now performing commits, they are done in the feature branch, while main branch keeps to be unchanged.

When others should see your feature branch, you have to publish it:



Install extension “Git Graph” in VS Code

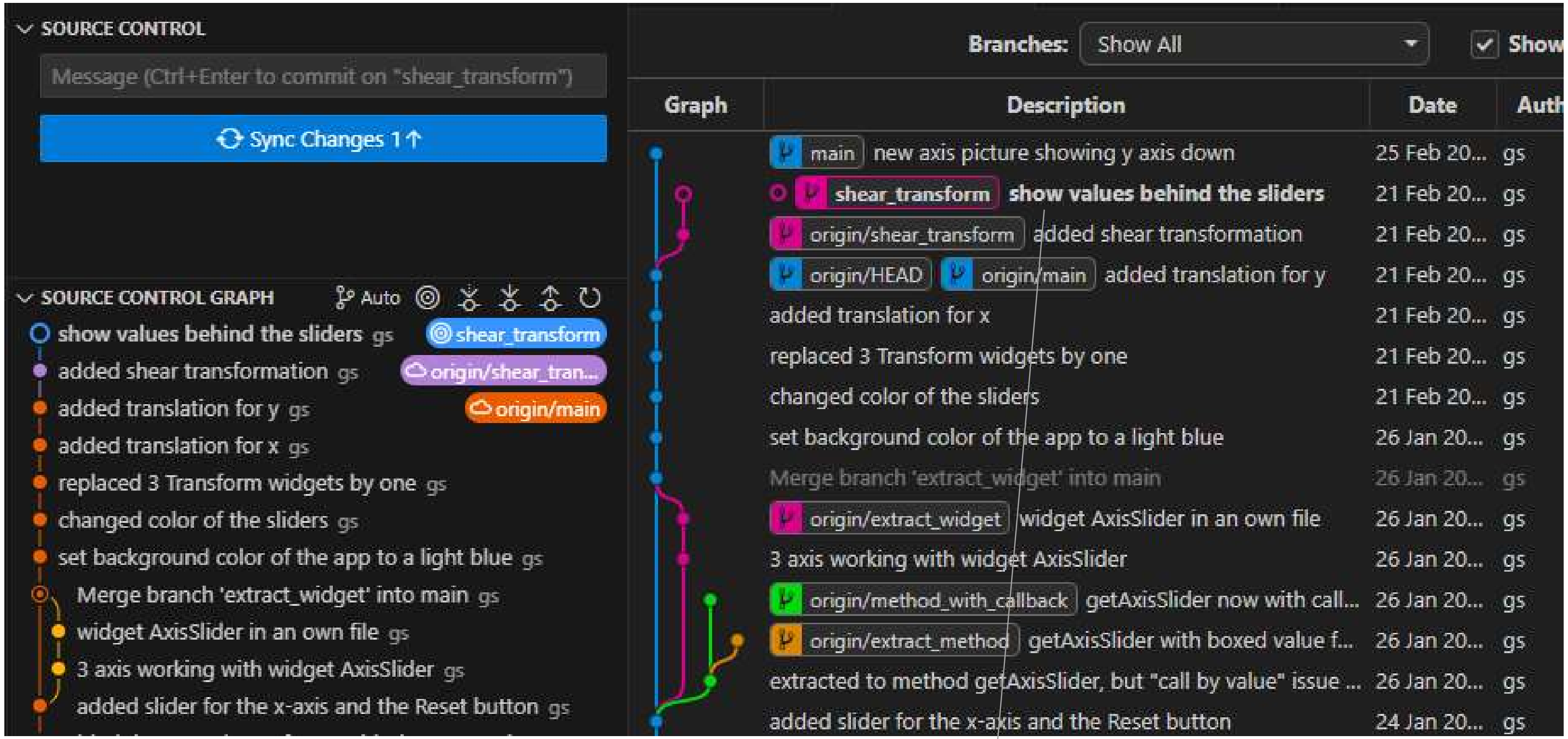


Git Graph: View a Git Graph of your repository, and perform Git actions from the graph.

Git Graph extension for Visual Studio Code

View a Git Graph of your repository, and easily perform Git actions from the graph. Configurable to look the way you want!

“Git Graph” simplifies your work with branches



SOURCE CONTROL

Message (Ctrl+Enter to commit on "shear_transform")

Sync Changes 1↑

SOURCE CONTROL GRAPH Auto

- show values behind the sliders gs — shear_transform
- added shear transformation gs — origin/shear_tran...
- added translation for y gs — origin/main
- added translation for x gs
- replaced 3 Transform widgets by one gs
- changed color of the sliders gs
- set background color of the app to a light blue gs
- Merge branch 'extract_widget' into main gs
- widget AxisSlider in an own file gs
- 3 axis working with widget AxisSlider gs
- added slider for the x-axis and the Reset button gs

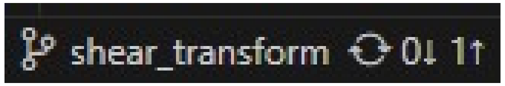
Branches: Show All ☑ Show

Graph	Description	Date	Auth
	main new axis picture showing y axis down	25 Feb 20...	gs
	○ shear_transform **show values behind the sliders**	21 Feb 20...	gs
	origin/shear_transform added shear transformation	21 Feb 20...	gs
	origin/HEAD origin/main added translation for y	21 Feb 20...	gs
	added translation for x	21 Feb 20...	gs
	replaced 3 Transform widgets by one	21 Feb 20...	gs
	changed color of the sliders	21 Feb 20...	gs
	set background color of the app to a light blue	26 Jan 20...	gs
	Merge branch 'extract_widget' into main	26 Jan 20...	gs
	origin/extract_widget widget AxisSlider in an own file	26 Jan 20...	gs
	3 axis working with widget AxisSlider	26 Jan 20...	gs
	origin/method_with_callback getAxisSlider now with call...	26 Jan 20...	gs
	origin/extract_method getAxisSlider with boxed value f...	26 Jan 20...	gs
	extracted to method getAxisSlider, but "call by value" issue ...	26 Jan 20...	gs
	added slider for the x-axis and the Reset button	24 Jan 20...	gs

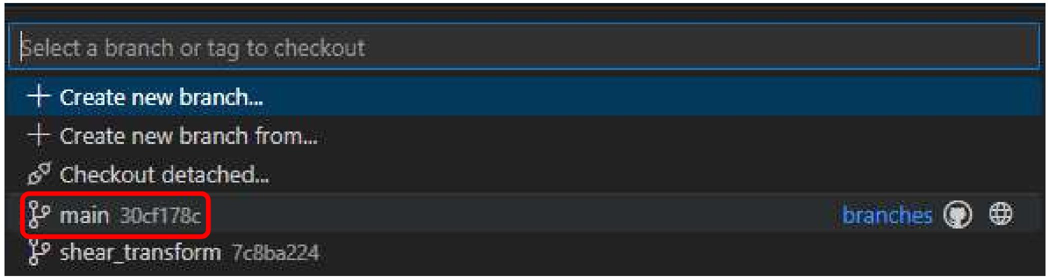
The current branch is shown in bold and with a small circle mark

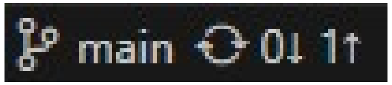
How to change the current branch

As already mentioned, you see the name of the current branch in the status line of VS Code on bottom left:

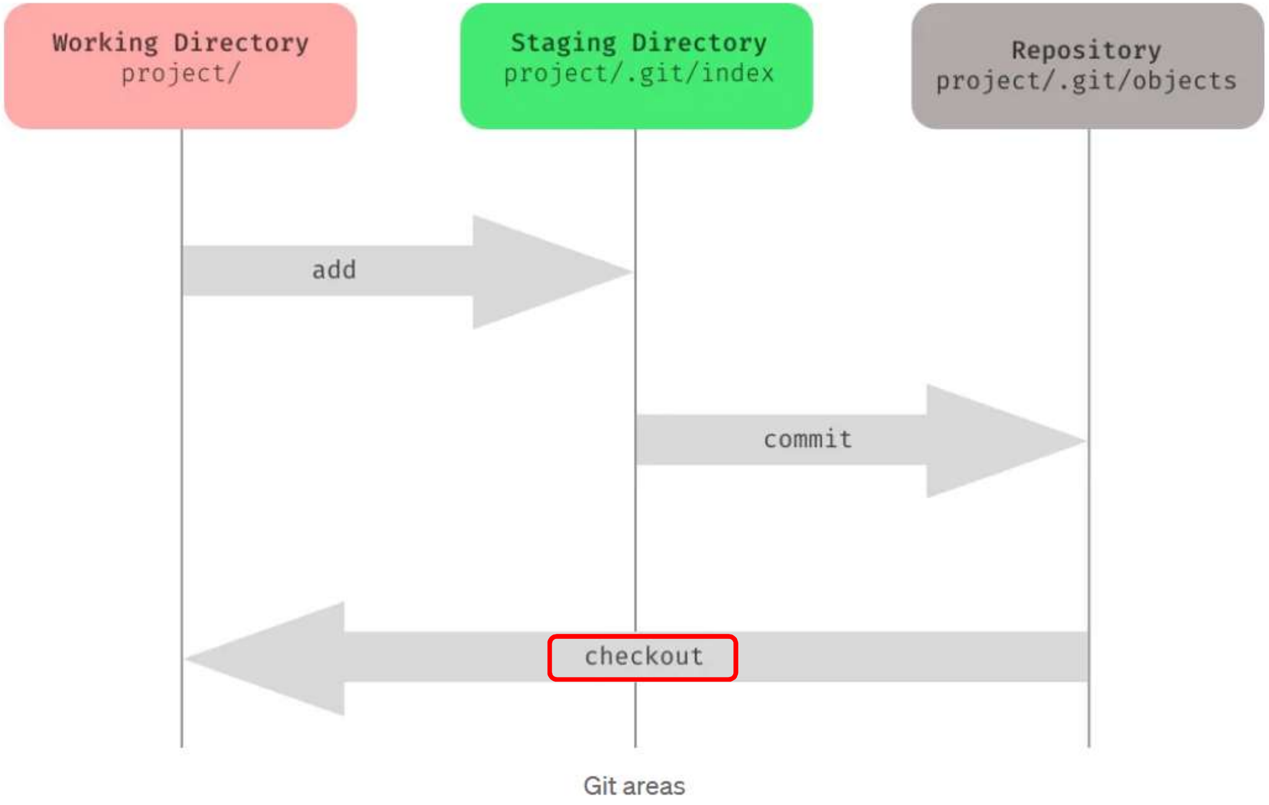


To “switch” to another branch (in git it’s called “to **checkout** another branch”) click on the branch name in the status line and in the upcoming dialog select the branch which should be checked out:





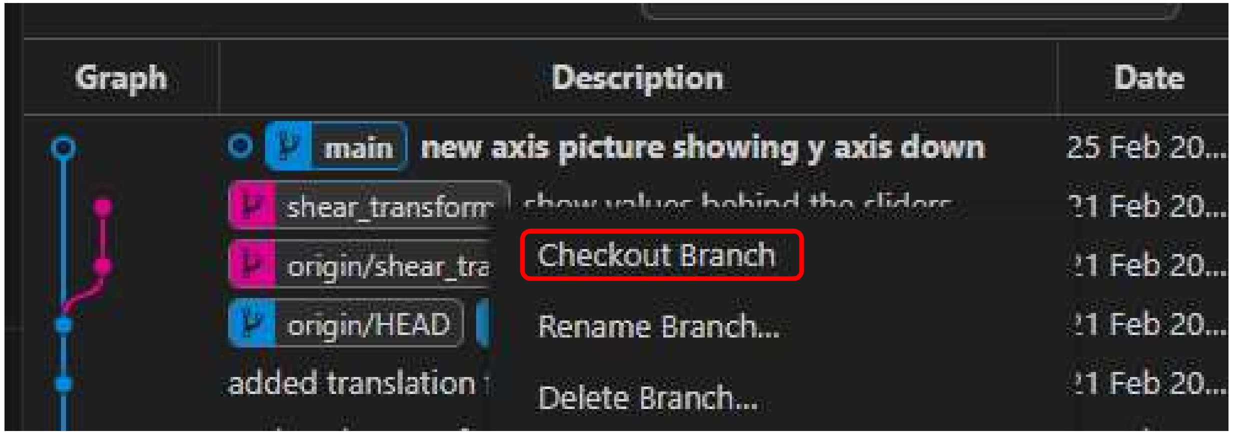
Reminder from “13a Git installation and first usage”



Copied from https://konrad126.medium.com/understanding-git-index-4821a0765cf

Checkout another branch using Git Graph

Press right mouse button on the name of the branch you want to checkout:

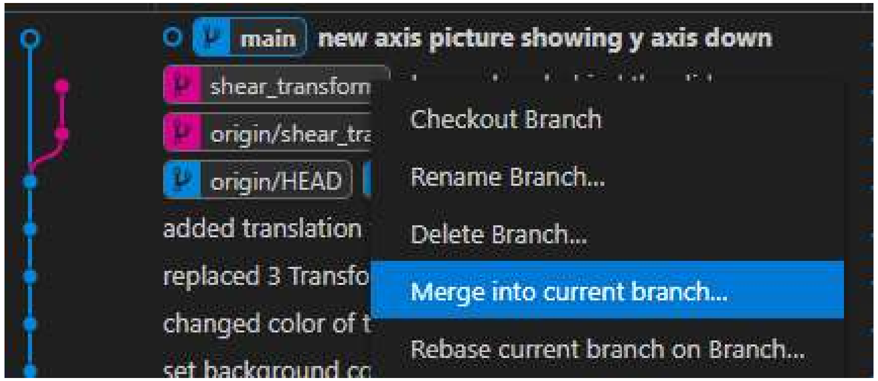


Graph	Description	Date
	main new axis picture showing y axis down	25 Feb 20...
	shear_transform show values behind the sliders	21 Feb 20...
	origin/shear_tra	21 Feb 20...
	origin/HEAD	21 Feb 20...
	added translation	21 Feb 20...

Take care:
it is important that you click right mouse button on the name of the branch and not on the description text in the same line !

Merge feature branch back to main branch

To bring the changes done in a feature branch back to main branch, you must merge.
To do so, checkout main branch and then click right mouse button on feature branch:



main new axis picture showing y axis down

shear_transform

origin/shear_tra

origin/HEAD

added translation

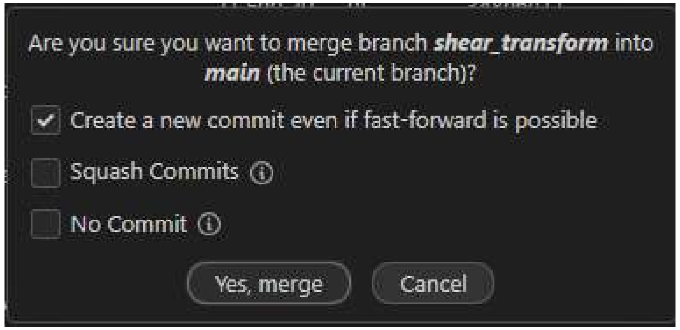
replaced 3 Transfo

changed color of t

set background co

Checkout Branch
Rename Branch...
Delete Branch...
Merge into current branch...
Rebase current branch on Branch...

→

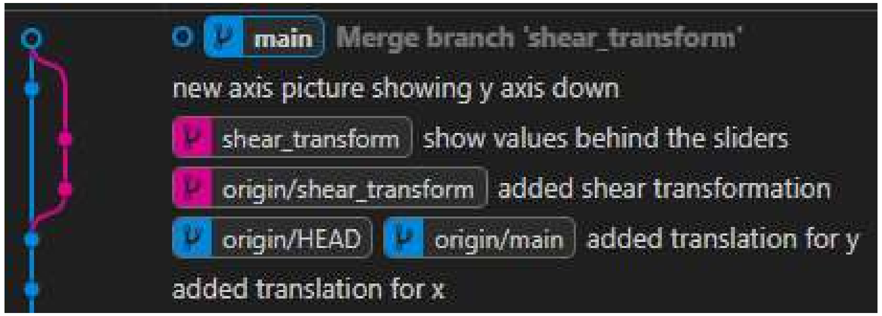


Are you sure you want to merge branch ***shear_transform*** into ***main*** (the current branch)?

☑ Create a new commit even if fast-forward is possible

□ Squash Commits ⓘ

□ No Commit ⓘ



main Merge branch 'shear_transform'

new axis picture showing y axis down

shear_transform show values behind the sliders

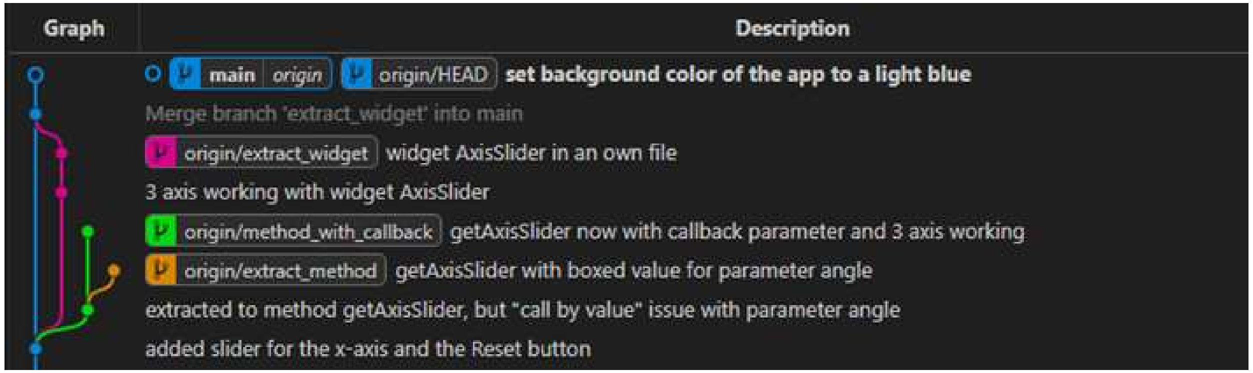
origin/shear_transform added shear transformation

origin/HEAD origin/main added translation for y

added translation for x

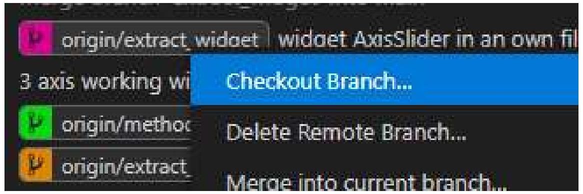
Local branches in a cloned repository

When you clone a repository from GitHub, only for the main branch a local branch is created:



Graph	Description
	main origin · origin/HEAD **set background color of the app to a light blue**
	Merge branch 'extract_widget' into main
	origin/extract_widget widget AxisSlider in an own file
	3 axis working with widget AxisSlider
	origin/method_with_callback getAxisSlider now with callback parameter and 3 axis working
	origin/extract_method getAxisSlider with boxed value for parameter angle
	extracted to method getAxisSlider, but "call by value" issue with parameter angle
	added slider for the x-axis and the Reset button

In case you want to continue working on a feature branch, check it out to create a new local branch for it:

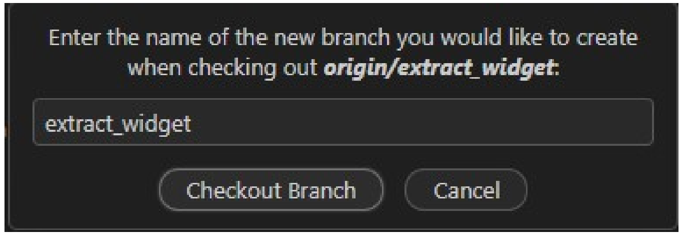


origin/extract_widget widget AxisSlider in an own fil

3 axis working wi

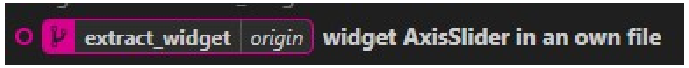
origin/methoc

origin/extract_



Enter the name of the new branch you would like to create when checking out ***origin/extract_widget***:

extract_widget



extract_widget origin **widget AxisSlider in an own file**

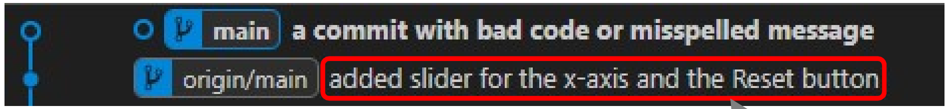
Remove commits from the repository

Once a commit is performed, it cannot be changed (it is **immutable**).

In case you have misspelled the commit message or detect an error in the committed code:

how to get rid of such a “bad” commit ?

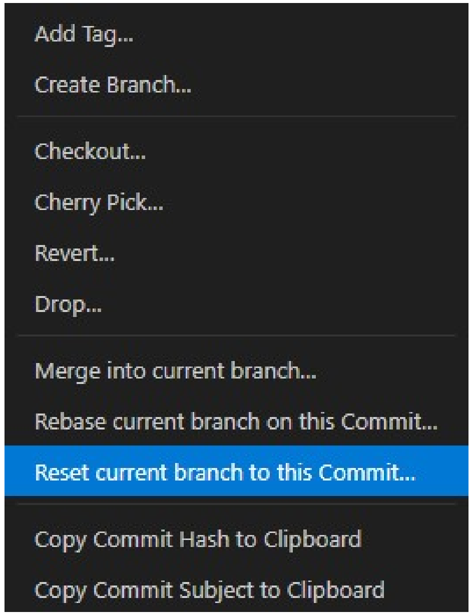
We assume the repository is not yet synced with GitHub:



main a commit with bad code or misspelled message

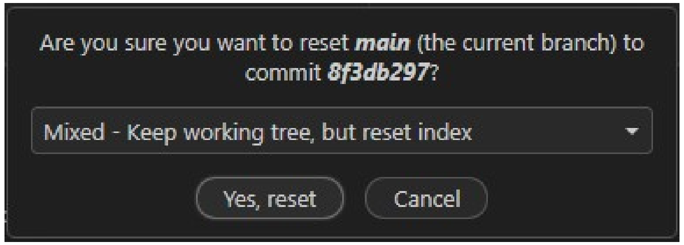
origin/main added slider for the x-axis and the Reset button

Press right mouse button on the description text in the second line and select:


Add Tag...
Create Branch...
Checkout...
Cherry Pick...
Revert...
Drop...
Merge into current branch...
Rebase current branch on this Commit...
Reset current branch to this Commit...
Copy Commit Hash to Clipboard
Copy Commit Subject to Clipboard

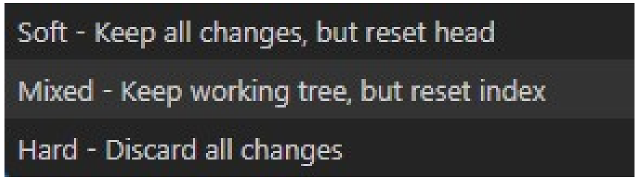
Soft, mixed and hard Resets

The following dialog will open:



Are you sure you want to reset ***main*** (the current branch) to commit ***8f3db297***?

Mixed - Keep working tree, but reset index



Soft - Keep all changes, but reset head
Mixed - Keep working tree, but reset index
Hard - Discard all changes

“**Soft**” will undo the last commit and keep its changes as “staged”, so you can directly enter a new commit message and commit again.
This is good when you had a misspelled commit message.

“**Mixed**” would also undo the last commit but keep its changes a “changed”.
Apply this when you want to do some minor code changes in your last commit.

As said in its description, “**Hard**” will discard all changes of the last commit.

In all 3 cases, the “bad” commit is deleted in your local repository.