Report

Fundamentals of Robot Programming

Homework assignment №2

Student: Oleg Rodionov

Sequencing

In the root there are README.txt + 5 files + 3 folder. One simple part contains addition 2 files or 1 folder by 1 commit.

Stage 1. Create local repository and commit README

1. Create working directory: mkdir githw2
2. Change directory cd githw2
3. git init - command to create local Git repository.
4. Create README.md in the WD with the inscription: “Robopy. Initial content”

- touch README.md; echo “Robopy. Initial content” > README.md

5. Add file to the staging area and implement the first commit

- git add README.md
- git commit -m “Initial commit”

Stage 2. Create the connection between local and remote repositories

1. Connect remote repo: git remote add origin git@github.com:rodosha98/GitHomework.git

I use SSH.

2. Push there the README file: git push -u origin master

Stage 3. Create branches

I will use 4 branches: one independent branch for each folder(docs, examples, robopy) and the last one for files in the root.

1. Create 1 branch: git branch docs
2. Create 2 branch: git branch examples
3. Create 3 branch: git branch robopy
4. Create 4 branch: git branch files
5. Show all branches: git branch

Now we can see, that there are 5 branches including master.

Stage 4. Work with the first branch

Consider the first branch. We need to add all files and folders in the local repo and also we need to change README after commits. These actions are the same for each branch, the difference is only in the contents of folders and files. Therefore, I will consider in detail only one branch.

1. Switch the branch: git checkout docs
2. Put necessary files in the working directory (Manually). “Docs" contains folder “images”, which composed of 6 files and 2 separate files.
3. Add new folder and file in it to the Staging area:

- git add docs/_config.yml docs/index.md

4. Add changes to the Readme. All changes save in the branch.

- Open README in the editor: gedit README.md and write changes manually right in the editor or
- Use terminal command echo Added configuration files from the DOCS folder >> README.md
- git add README.md

5. Use git status to see modified files
6. git commit -m “ Added configuration files from the DOCS folder.”
7. Send changes to the remote repo: git push -u origin docs

Add files from the image folder in the same way. In order to save time, I decided to add this folder as 1 commit, as it contains only pictures.

1. Add the whole folder: git add docs/images

Or git add docs/images/logo.png docs/images/make_gif.png docs/images/ Orion5.png. plot_pose.png docs/images/puma560.JPG docs/images/qt.png docs/ images/ docs/images/robopy.png docs/images/transforms.png

2. Add changes to the Readme. echo Images are available >> README.md;

git add README.md

3. Commit changes

git commit -m “ Added images from the DOCS folder.”

4. Push to the remote repo: git push -u origin docs

Stage 5. Work with other branches

Working with other branches is no different - only the content. Therefore, below I added only the code how to implement a local repository.

The second branch. examples branch

First 2 files. Position files

git checkout examples;

git add examples/pose_multiply.py examples/pose_plot.py;

echo In pose python files there are examples how to define the position of the tool >> README.md;

git add README.md;

git commit -m "2 position estimation python files are added. README file is correspondingly modified";

git push -u origin examples;

Last 2 files. Animation files

git add examples/puma560_animation.py examples/puma560_plot.py;

echo In animation python files there are example of Puma project with animation of moving and robot’s configuration plot >> README.md;

git add README.md;

git commit -m "2 animation python files are added. README file is correspondingly modified";

git push -u origin examples;

The third branch. robopy branch

git checkout robopy;

1.Init file

git add robopy/__init__.py;

git commit -m "Append the __init__.py file as regular package implementation in Python";

git push -u origin robopy;

2. Base folder

Commit 1: first 3 files(1-3):

git add robopy/base/_init_.py robopy/base/check_args.py robopy/base/common.py;

echo Base is implemented with 10 python files. This is the first the part, which contain init file for import, file with arguments and common file with the main functions for matrix type definition>> README.md;

git add README.md;

git commit -m "3 python files are appended. They contains function and arguments definition. README.md has been modified ";

git push -u origin robopy;

Commit 2: next 3 files(4-7):

git add robopy/base/graphics.py robopy/base/model.py robopy/base/pose.py;

echo Those files comprised of functions to determine the position of robot, function to build model and to draw graphics >> README.md;

git add README.md;

git commit -m "3 python files to build the model, define the position and draw graphics are added. README.md has been modified";

git push -u origin robopy;

Commit 3: next 3 files(7-9):

git add robopy/base/quaternion.py robopy/base/serial_link.py robopy/base/super_pose.py;

echo Serial_link is needed to create links and after that group them to the robot. Quarternion is a method t calculate Forward and inverse kinematics of the robot. Super pose file contains modyfied functions to define the position>> README.md;

git add README.md;

git commit -m " Next 3 python files are needed to build a robot, estimate kinematics and define it's position. README.md has been modified";

git push -u origin robopy;

Commit 3: last files(7-9):

git add robopy/base/transforms.py robopy/base/util.py ;

echo Transform file contains functions to implement operations between matrices. Util file to delete your model and make changes after that>> README.md;

git add README.md;

git commit -m "Last 2 python files. README.md has been modified";

git push -u origin robopy;

git log

3. Media folder

Commit 1: images folder

git add robopy/media/imgs;

echo In images folder there are images of the floor and robotype logot>> README.md;

git add README.md;

git commit -m "Add 2 images of floor and logo. README.md has been modified";

git push -u origin robopy;

Commit 2: orion5 project

git add robopy/media/orion5;

echo Orion5 project contains 7 stl files which describe it's 3D model of each link>> README.md;

git add README.md;

git commit -m "Orion5 project 3D model is added. README.md has been modified";

git push -u origin robopy;

Commit 3: puma 650

git add robopy/media/puma560

echo Puma 560 project contains 7 stl files which describe it's 3D model of each link. Puma is 7-linked robot including the base >> README.md;

git add README.md;

git commit -m "Puma560 project 3D model is added README.md has been modified";

git push -u origin robopy;

4. Tests folder

Commit 1 - init file

git add robopy/tests/__init__.py;

git commit -m "Append the __init__.py file as regular package implementation in Python ";

git push -u origin robopy;

Commit 2 - test files

git add robopy/tests/test_common.py robopy/tests/test_pose.py robopy/tests/ test_transforms.py;

echo This test files include test data to prove the correctness of a model>> README.md;

git add README.md;

git commit -m "3 python test files are appended. README.md has been modified";

git push -u origin robopy;

Stage 6.Testing ang merging

All branches are needed to be tested. After this, if all is correct, we can merge all branches with a master.

1. Go to the master branch: git checkout master;

2. Merge with each branch:

- git merge robopy
- git push origin master
- git merge docs

After merging conflicts are happened. I fixed it manually for that:

1. git status - see where are the conflicts occurred.
2. gedit README.md - open file in editor
3. Make changes manually (delete signs of conflict and add text)
4. git add README.md - add upgraded file in the staging area
5. Git commit -m "Resolving conflict by combining 2 parts"
6. git pull - Downloads bookmark history and incorporates changes and check new conflicts
7. After fixing all conflicts - git push origin master

- git merge files
- Conflicts solving
- git push origin master
- git merge example
- Conflicts solving
- git push origin master

3. Check the correctness of a master branch

Stage 7.Cleaning

Delete the README.md file:

1. git rm README.txt - remove file from the git repo

If you want to save file in working directory use:

git rm —cached README.md

2. git commit -m “remove README.md”

3. Push changes in the remote GitHub:

    git push origin master

Delete Docs folder:

1. git rm -r docs - remove folder recursively from the git repo
2. git commit -m “remove docs folder”
3. Push changes in the remote GitHub:

    git push origin master

Add report:

1. Add report to the working directory
2. git add REPORT.md
3. git add report.pdf
4. git commit “Adding report in 2 files types”
5. git push origin master

Useful commands

1. gedit file.txt - open file in the editor
2. git reset —hard HEAD^ - reset everything to the previous commit
3. git clean -f -d - Remove directories(delete untracked files)
4. git clean -f -X - Remove ignored files git clean -f -X