Introduction

Welcome to Module 2! This week, we will learn about committing to version control using Git. Git is a distributed version control system. It tracks changes in your source code. In other words it keeps the history of your source code. Git is free and open-source software. It was created to aid with the development of the Linux kernel. Git has two advantages: distributed development and speed. The first advantage, distributed development, is where different teams work on different repositories. Repositories can synchronize with each other at certain times. Speed is provided by retrievals and synchronization of large code bases in very little time.

Topics and Learning Objectives

Topics

- Committing to version control with Git
- Basic Git commands
  - Committing
  - Branching
  - Reading diffs
- Git configuration
- Pushing/Pulling committed code to GitHub
- GitHub pull requests/pull request review

Learning Objectives

By the end of the module, you should be able to:

1. Develop a working knowledge of Git and GitHub.
2. Commit a web application to version control (local and remote).
3. Write good Git commit messages.

Readings & Resources

Reading

Required

- Github Git cheat sheet.
- 5 Useful tips for a better commit message.
- Pope, T. (2008). A note about Git commit messages. tbaggery.
- Thoughtbot, Inc. (n.d.). Mastering Git.

What Is Git?

Source: Wikimedia Commons

Git is a version control system created by Linus Torvalds in 2005. It is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Some facts about Git:

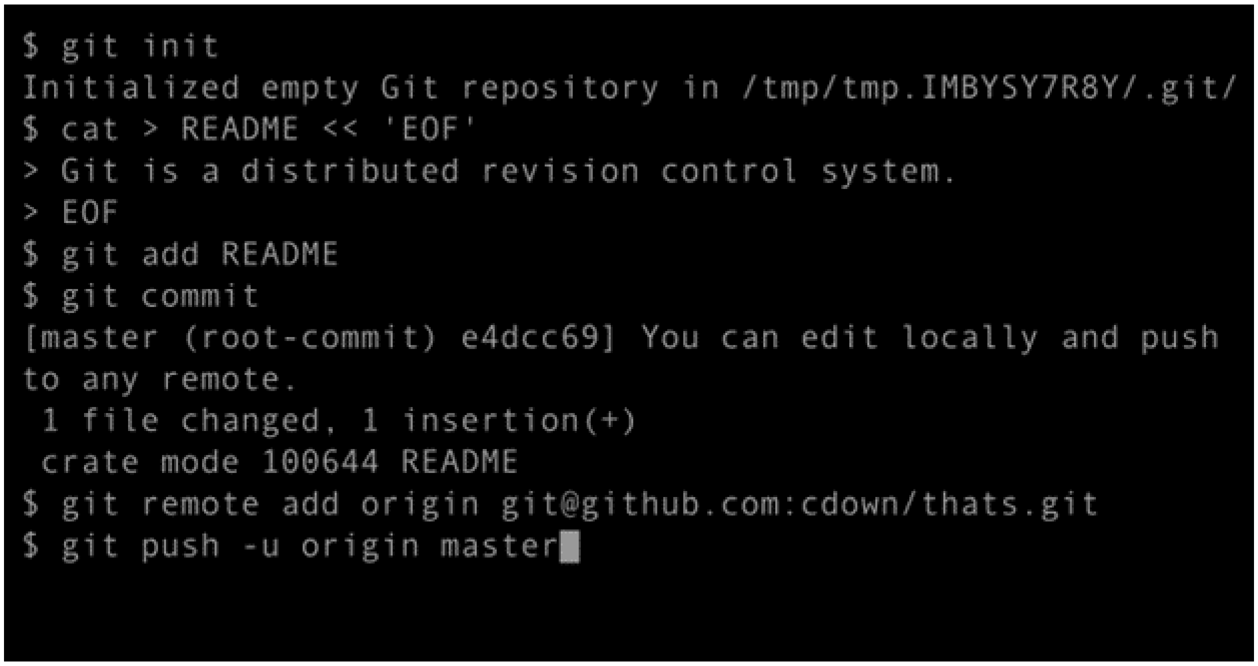
- Git may be locally installed from the website git-scm.com.
- Over 70% of developers use Git.
- Developers can work together using the Internet.
- Developers have access to full history of project (using project history).

GitHub

GitHub is a hosted distributed version control service that uses Git. It also provides its own features that are not supplied by Git. GitHub provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project.

GitHub is currently owned by Microsoft, and has free and paid plans.

Working with Git



```
$ git init
Initialized empty Git repository in /tmp/tmp.IMBYSY7R8Y/.git/
$ cat > README << 'EOF'
> Git is a distributed revision control system.
> EOF
$ git add README
$ git commit
[master (root-commit) e4dcc69] You can edit locally and push
to any remote.
 1 file changed, 1 insertion(+)
 crate mode 100644 README
$ git remote add origin git@github.com:cdown/thats.git
$ git push -u origin master
```

Figure 2.1. A representative git session, demonstrating the initialization of a repository, and the addition of file and a remote.

Source: Wikimedia Commons

Git is used to manage source code in a project. Before we write any source code we must set up the project directory structure, initialize git (see first line in Figure 2.1 above) and commit files (see line 7 in Figure 2.1 above). The steps below are the recommended steps to create a project directory and initialize it to be tracked by Git. To carry out the steps below you can simply start with the project directory and within it you can have a single text file or “readme.txt” file (see lines 3 to 6 in Figure 2.1 above).

To work with Git, we would take the following steps:

1. Initialize Git on a folder/directory. This action creates a repository.
2. Git creates a hidden folder that keeps track of changes within the project.
3. A file is considered modified when it is added, changed or deleted.
4. Modified files can be staged.
5. Staged files may be committed. This stores a permanent snapshot of the files. A project history is created by adding this snapshot to the git branch.
6. Full history of every commit is visible.
7. You can revert back to a previous commit.

Git keeps track of changes made by each commit instead of storing separate files from every commit.

Using Git

- Git may be used from the command line using the “git” command
  - git --version
- In order for a developer to check code into Git they must configure their user name and email. This identity is used to keep track of who makes commits.
- User name and email may be configured from command prompt
  - git config –global user.name “user1”
  - git config –global user.email user1@torontomu.ca

Creating a Git Repository

- A git repository resides within a directory/folder. This is called the project folder.
- A developer may create a project folder from the command prompt using the mkdir command.
  - mkdir project1
- Once a project folder exists the user can navigate into it and initialize Git.
  - cd project1
  - git init

Creating Source Code

- Project files may be created within the project directory/folder.
- The git status may be used in order to identify tracked files and untracked files.
  - A tracked file is a file that Git knows and it is part of the git repository
  - An untracked file is a new file that it is not part of the git repository. All new files are not part of the git repository.
- To commit source code changes is a three step process
  - Add the file or track the file – git add filename
  - Stage tracked files
  - Commit staged files

Git Commands – Local Repository

Click on each of the git commands below to reveal their definitions:

Click-n-reveal: **git init**

This creates a new empty git repository or converts an existing folder to a git repository.

Click-n-reveal: **git add**

This tracks a file.

Click-n-reveal: **git commit**

This saves changes from tracked files to a repository.

Click-n-reveal: **git push**

This updates the remote copy of the repository with the local changed one and commits it.

Click-n-reveal: **git status**

This displays information since the last commit.

- It lists files that have changes but are not committed (working directory vs staging directory) and
- Indicates how many commits exists that have not been pushed to the remote repository.

Click-n-reveal: **git log**

This shows the commit history.

Click-n-reveal: **git log –graph –online**

This shows branch info as a graph.

Branches

- In Git a branch is a new version of the main repository. In Figure 2.2 below, the main repository is represented by block 1 and the branch from block 1 is represented by block 2.
- Git has a master or main branch. In Figure 2.2, block 1 represents the master or main branch. Branches are always created from the master or main branch. Block 2 is a branch from the master branch. Block 3 will merge into the master branch.
- Developers can work on different branches without impacting the master or main branch.
- When developers have completed work on their own branch, then it can be merged into the master or main branch. In Figure 2.2, block 3 is merged into the master branch and block 7 is merged into the master branch.
- Branching in Git is lightweight and fast as compared to other version control systems.

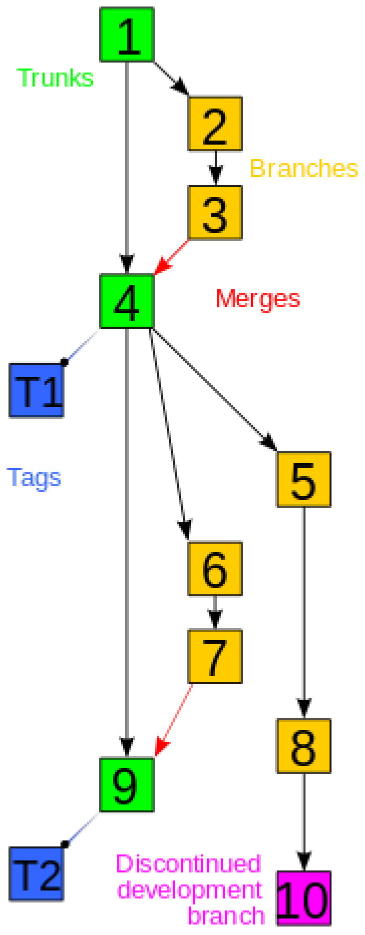


Figure 2.2. Visualization of the "history tree" of a revision controlled project, showing branching, merging, tagging, etc.

Source: Wikimedia Commons

In Figure 2.2 above, the master or main branch is represented by boxes in the trunk (1, 4, 9). Branches stem from the master or main branch (including boxes 2, 3, 5, 6, 7, 8), and some can even be discontinued development branches (e.g., box 10).

Merging Branches

- All “good code” resides in the master branch.
- Ideally you can work from the master branch; however, most of the time you work on your own branch.
- When the code in your branch has been perfected it can be merged back into the master branch.
- When merging changes, merging conflicts may occur. Merging conflicts may be determined using the “git status” command.
- Merging conflicts have to be manually fixed and then a merger may be reattempted. If there are no conflicts the merger will succeed.

Remote Repository

- A Git remote repository is a repository that does not reside on your development machine.
- Any Git repository that resides on your development machine is a local repository.
- Any Git repository on GitHub is a remote repository.
- Remote repositories are accessed using URL (uniform resource locator).
- A remote repository is copied to your development machine by cloning a git repository.

Clone vs Checkout

- In order to obtain source code from Git or GitHub you have to clone a project.
- Once you have a project then you can move into the appropriate directory by checkout a branch.
- Two command that help navigating through branch structure are:
  - **git status** – Status command allows us to see current branch, tracked changes and untracked changes.
  - **git branch** – Branch command allows us to see all branches, including current branch.

Git Commands – Remote Repository

Click on each of the git commands below to reveal more about them:

Click-n-reveal: **Git clone**

Cloning is the process of creating a working copy of a remote repository on a local directory or working directory. For example,

```
git clone username@git_server_hostname:/path_of_repository
```

Click-n-reveal: **Git pull**

This is if you have already cloned the repository and need to update local repository in order to match remote repository. For example,

```
git pull origin main
```

Click-n-reveal: **Git fetch**

This is the process of updating only git information from the remote repository.

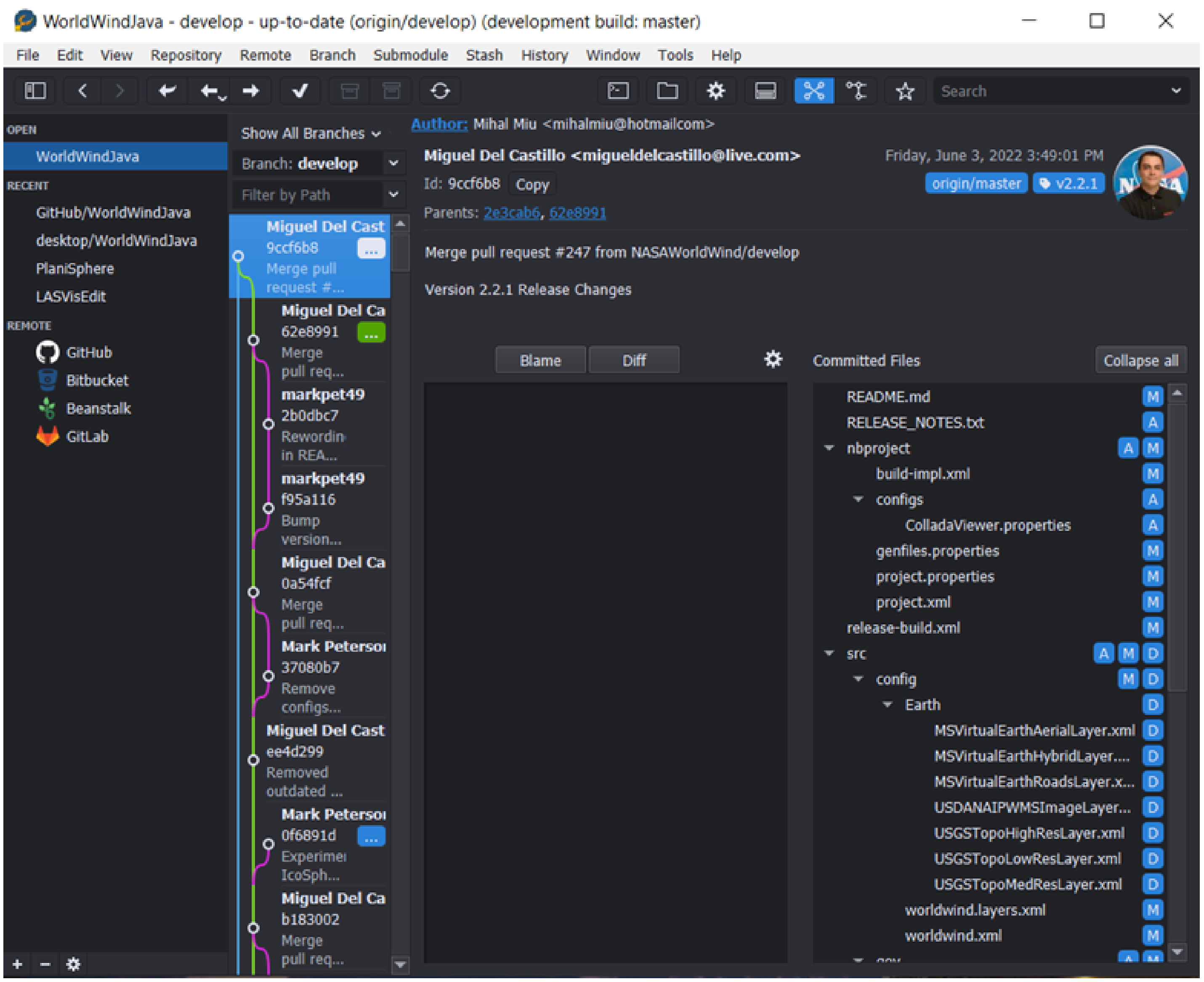
Git GUI

- In addition to the git command line tools from git-scm.com, we can use GUIs.
- Gitttyup is a graphical Git client designed to help you understand and manage your source code history. It supports Linux, macOS, and MS Windows.
- GitHub Desktop is a graphical Git client that focuses on projects hosted on GitHub. It supports macOS and MS Windows.
- You may also find other Git GUI clients on the git-scm.com site.

GittyUp

- Gittyup showing NASA WorldWindJava repository (see Figure 2.3)

- In Figure 2.3 below, we can see Gittyup displaying local branch history and commits. The latest commit was made into the “develop” branch by Miguel Del Castillo on June 3, 2022. Viewing the local branch history (including merges into branches) is extremely easy in Gittyup, since it creates a graph where commits are organized chronologically. Achieving this with the Git CLI (Command Line Interface) is not a trivial task and will require multiple commands. Complex tasks are easier to achieve with Gittyup when compared to Git CLI
- Gittyup is open source software



Miguel Del Cast 9ccf6b8 Merge pull request #...
Miguel Del Ca 62e8991 Merge pull req...
markpet49 2b0dbc7 Rewordin in REA...
markpet49 f95a116 Bump version...
Miguel Del Ca 0a54fcf Merge pull req...
Mark Petersoı 37080b7 Remove configs...
Miguel Del Cast ee4d299 Removed outdated ...
Mark Petersoı 0f6891d Experimei IcoSph...
Miguel Del Ca b183002 Merge pull req...

Author: Mihal Miu <mihalmiu@hotmailcom>
Miguel Del Castillo <migueldelcastillo@live.com> Friday, June 3, 2022 3:49:01 PM
Id: 9ccf6b8
origin/master v2.2.1
Parents: 2e3cab6, 62e8991

Merge pull request #247 from NASAWorldWind/develop

Version 2.2.1 Release Changes

README.md M
RELEASE_NOTES.txt A
nbproject A M
build-impl.xml M
configs A
ColladaViewer.properties A
genfiles.properties M
project.properties M
project.xml M
release-build.xml M
src A M D
config M D
Earth D
MSVirtualEarthAerialLayer.xml D
MSVirtualEarthHybridLayer.... D
MSVirtualEarthRoadsLayer.x... D
USDANAIPWMSImageLayer... D
USGSTopoHighResLayer.xml D
USGSTopoLowResLayer.xml D
USGSTopoMedResLayer.xml D
worldwind.layers.xml M
worldwind.xml M

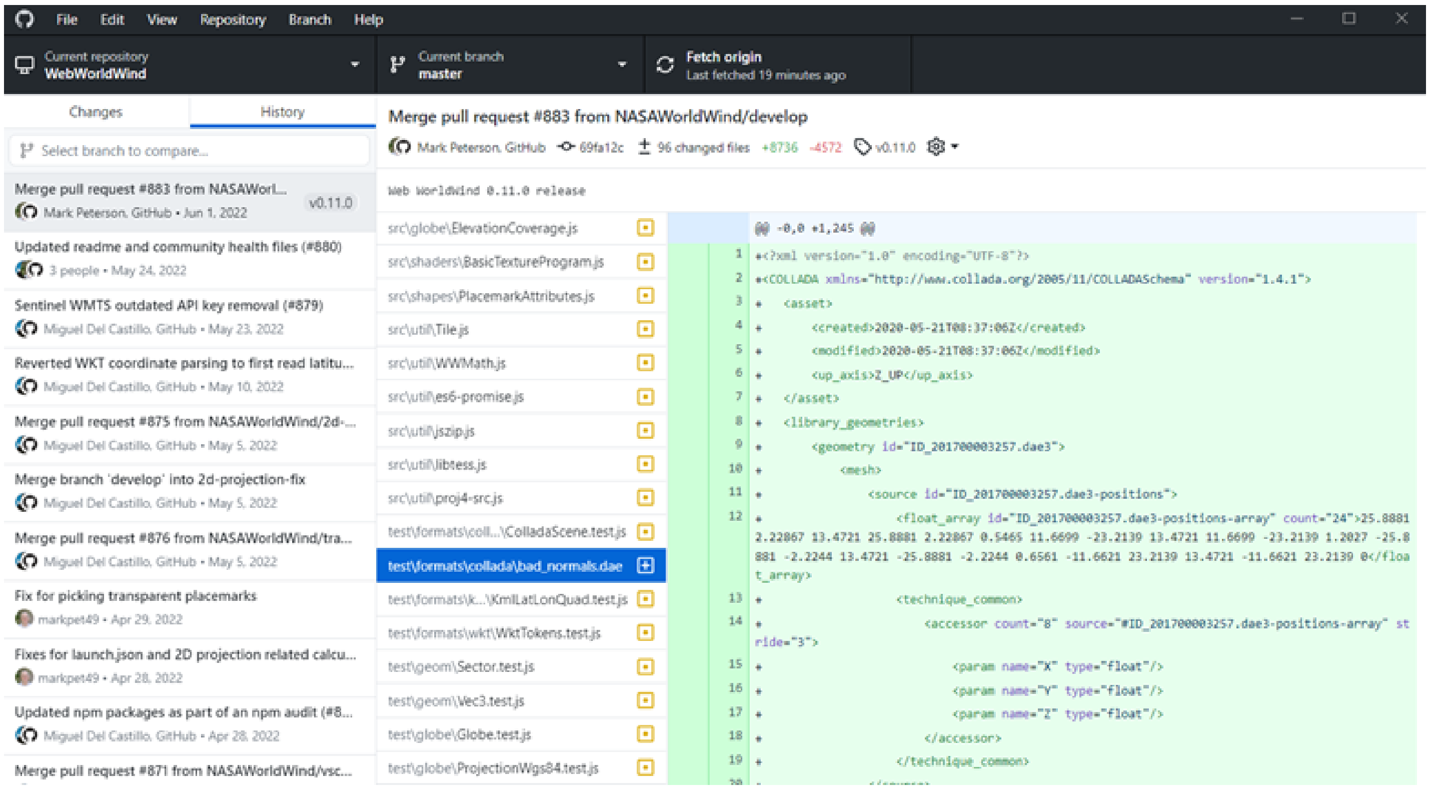
Figure 2.3. The NASA WorldWindJava repository on Gittyup.

Source: Mihal Miu

GitHub Desktop

- GitHub Desktop showing NASA WorldWindJava repository (see Figure 2.4 below)

- Note the UI is not as visually appealing as Gittyup
- GitHub is freely downloadable from GitHub



Merge pull request #883 from NASAWorl... v0.11.0
Mark Peterson, GitHub • Jun 1, 2022

Updated readme and community health files (#880)
3 people • May 24, 2022

Sentinel WMTS outdated API key removal (#879)
Miguel Del Castillo, GitHub • May 23, 2022

Reverted WKT coordinate parsing to first read latitu...
Miguel Del Castillo, GitHub • May 10, 2022

Merge pull request #875 from NASAWorldWind/2d-...
Miguel Del Castillo, GitHub • May 5, 2022

Merge branch 'develop' into 2d-projection-fix
Miguel Del Castillo, GitHub • May 5, 2022

Merge pull request #876 from NASAWorldWind/tra...
Miguel Del Castillo, GitHub • May 5, 2022

Fix for picking transparent placemarks
markpet49 • Apr 29, 2022

Fixes for launch.json and 2D projection related calcu...
markpet49 • Apr 28, 2022

Updated npm packages as part of an npm audit (#8...
Miguel Del Castillo, GitHub • Apr 28, 2022

Merge pull request #871 from NASAWorldWind/vsc...

Merge pull request #883 from NASAWorldWind/develop
Mark Peterson, GitHub 69fa12c 96 changed files +8736 -4572 v0.11.0

Web WorldWind 0.11.0 release

src\globe\ElevationCoverage.js
src\shaders\BasicTextureProgram.js
src\shapes\PlacemarkAttributes.js
src\util\Tile.js
src\util\WWMath.js
src\util\es6-promise.js
src\util\jszip.js
src\util\libtess.js
src\util\proj4-src.js
test\formats\coll...\ColladaScene.test.js
test\formats\collada\bad_normals.dae
test\formats\k...\KmlLatLonQuad.test.js
test\formats\wkt\WktTokens.test.js
test\geom\Sector.test.js
test\geom\Vec3.test.js
test\globe\Globe.test.js
test\globe\ProjectionWgs84.test.js

```
@@ -0,0 +1,245 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<COLLADA xmlns="http://www.collada.org/2005/11/COLLADASchema" version="1.4.1">
+   <asset>
+       <created>2020-05-21T08:37:06Z</created>
+       <modified>2020-05-21T08:37:06Z</modified>
+       <up_axis>Z_UP</up_axis>
+   </asset>
+   <library_geometries>
+       <geometry id="ID_201700003257.dae3">
+           <mesh>
+               <source id="ID_201700003257.dae3-positions">
+                   <float_array id="ID_201700003257.dae3-positions-array" count="24">25.8881
2.22867 13.4721 25.8881 2.22867 0.5465 11.6699 -23.2139 13.4721 11.6699 -23.2139 1.2027 -25.8
881 -2.2244 13.4721 -25.8881 -2.2244 0.6561 -11.6621 23.2139 13.4721 -11.6621 23.2139 0</floa
t_array>
+                   <technique_common>
+                       <accessor count="8" source="#ID_201700003257.dae3-positions-array" st
ride="3">
+                           <param name="X" type="float"/>
+                           <param name="Y" type="float"/>
+                           <param name="Z" type="float"/>
+                       </accessor>
+                   </technique_common>
```

Figure 2.4. The NASA WorldWindJava repository on GitHub Desktop.

Source: Mihal Miu

Video

Please watch the following video, Git Introduction. This video demonstrates how Git is used from the command prompt. Commands such as “git init”, “git add”, and “git commit” are used to create your project repository. Source code history may be viewed with “git status” or “git log” commands.

Middleware Web Template

TMU Video

Summary

Source code history is important since multiple people may contribute to an application. Source code contributions made by developers may or may not overlap. In order to maintain continuity, source code history may be browsed by current developers without the presence of former developers. Git is the most widely used version control system by developers.

Assessments

Each week, you'll find here reminders of assignments or labs that you should be working on. For Week 2:

- Lab 2: Opening your first pull request and code review (5% of the final grade) is due in 7 days, i.e., the end of day, Friday, of Week 2.

Reminder

You have a weekly Zoom session. The date/timing for your Zoom session can be found in the Course Schedule in the Course Outline. The link to the Zoom session will be provided separately to you by your instructor. The Zoom session will consist of 20–30 minutes of lecture, followed by questions and answers (Q&A), followed by a lab period. You may ask questions about the previous or current week's content or labs.

Please bring any questions to the weekly Zoom session. However, if you have any questions during the week outside of the Zoom session, you may post them in the Discussion Board. Your instructor may respond in the Discussion board or during the weekly Zoom session.

Click-n-reveal: **Where is the Discussion board?**

Click on the “Communication” area at the top main course menu, and select Discussions from the dropdown menu.

References

- Loeliger, J. (2022). Version Control with Git (3rd ed.). O’Reilly Media, Inc.