

REPORT NO. 02.1: VERSION CONTROL WITH GIT

COURSE CODE: CSE 404
COURSE TITLE: SOFTWARE ENGINEERING AND ISD
LABORATORY

Submitted by

SHANJIDA ALAM (ID: 353)

Submitted to

Dr. Md. MUSFIQUE ANWAR, Professor

Dr. Md. HUMAYUN KABIR, Professor



Computer Science and Engineering
Jahangirnagar University

Dhaka, Bangladesh

September 05, 2024

Contents

1	OBJECTIVE	1
1.1	Objectives	1
2	GIT COMMAND	2
2.1	Git Commands	2
3	GIT COMMAND LIST	7

1. OBJECTIVE

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. And GitHub is a web-based platform that provides version controls and collaboration features for software development projects, utilizing Git as its underlying version control system. Git and GitHub are related to each other.

1.1 Objectives

The objectives are:

- **Collaboration:**Git enables several developers to collaborate on a single project at the same time without changing each other's modifications. Branching and merging make this easier.
- **Backup:**Git maintains a distributed backup of the complete project, including its history.
- **Documentation:**Git promotes the usage of meaningful commit messages, which help in team communication by outlining the intent and nature of changes.
- **Learning and Sharing:**GitHub is a great platform for learning by doing practical coding and participating in ongoing projects. It also allows developers to showcase their work to potential employers or collaborators.

2. GIT COMMAND

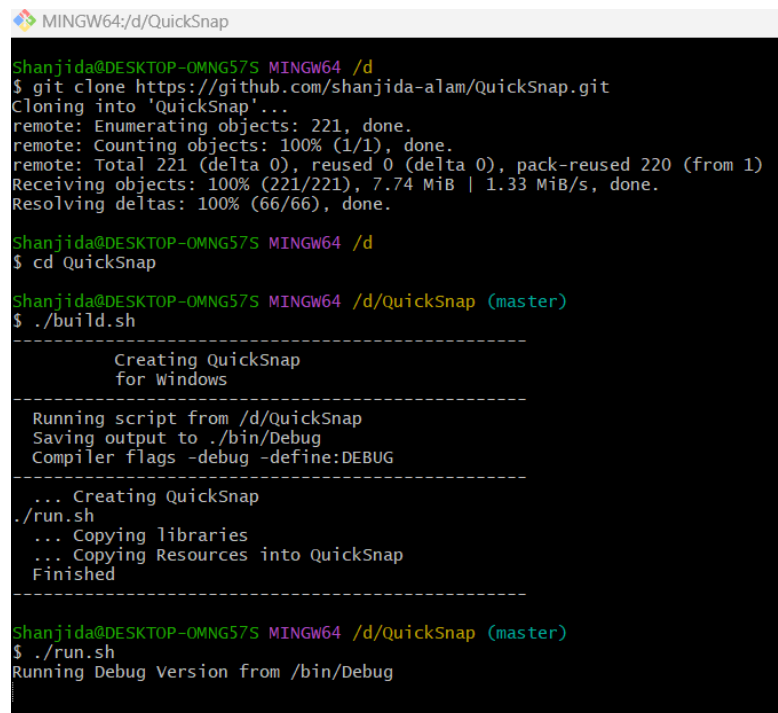
I lead the Git version control lab as a team leader (TL). In essence, I explain here how I use Git for version control.

2.1 Git Commands

First, I would navigate to <https://github.com/macite/QuickSnap> and fork the repository in my GitHub account.

The commands are:

- **Git Clone Command:** `git clone https://github.com/shanjida-alam/QuickSnap.git`



```
MINGW64:/d/QuickSnap

Shanjida@DESKTOP-OMNG57S MINGW64 /d
$ git clone https://github.com/shanjida-alam/QuickSnap.git
Cloning into 'QuickSnap'...
remote: Enumerating objects: 221, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 221 (delta 0), reused 0 (delta 0), pack-reused 220 (from 1)
Receiving objects: 100% (221/221), 7.74 MiB | 1.33 MiB/s, done.
Resolving deltas: 100% (66/66), done.

Shanjida@DESKTOP-OMNG57S MINGW64 /d
$ cd QuickSnap

Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (master)
$ ./build.sh

-----
          Creating QuickSnap
          for Windows
-----

Running script from /d/QuickSnap
Saving output to ./bin/Debug
Compiler flags -debug -define:DEBUG
-----

... Creating QuickSnap
./run.sh
... Copying libraries
... Copying Resources into QuickSnap
Finished
-----

Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (master)
$ ./run.sh
Running Debug Version from ./bin/Debug
```

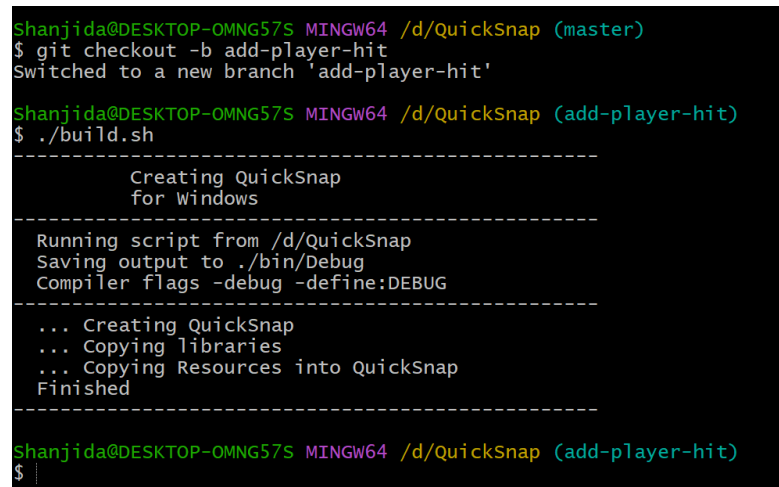
Figure 2.1: Screenshots of git clone

Here, I basically open the git bash terminal in my D drive folder and type the git clone command to clone the repository in my machine. And check that the project is cloned my local computer.

Purpose of git clone command: The purpose of the git clone command is to copy an existing Git repository to a new location on my local computer.

Meaning of the Clone parameter in the git clone command: The meaning of the clone parameter in the git clone is that it creates a new directory and copies all the contents of the repository, including all versions of every file and folder. By default, git clone creates a reference to the remote repository called "origin".

- **Creating and working with branches in git:** git checkout -b add-player-hit



```
Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (master)
$ git checkout -b add-player-hit
Switched to a new branch 'add-player-hit'

Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)
$ ./build.sh

-----
          Creating QuickSnap
          for Windows
-----
Running script from /d/QuickSnap
Saving output to ./bin/Debug
Compiler flags -debug -define:DEBUG
-----

... Creating QuickSnap
... Copying libraries
... Copying Resources into QuickSnap
Finished
-----

Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)
$
```

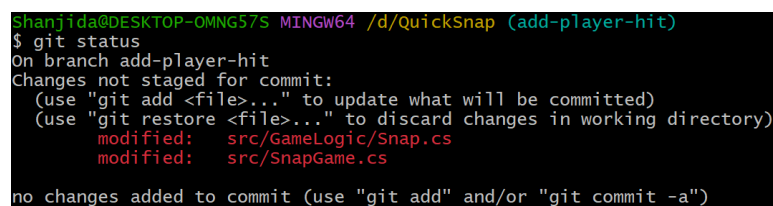
Figure 2.2: Screenshots of creating new branch and switching to a new branch 'add-player-hit'

In this step, I would create a new branch with the help of the 'git checkout -b add-player-hit' command and switch to a new branch, 'add-player-hit'. Then I would open the **SnapGame.cs** file and go to the **HandleUserInput** method to change some portion of the code. And also open the **Snap.cs** file and again go to the **PlayerHit** method and update it so that the player's score is decreased if they don't snap two cards with the same rank.

Purpose of create branch command in git: The purpose of the **git checkout -b branch-name** command is creating a new branch and switching to the new branch. It can use to create a new branch from an existing one.

Meaning of the '-b' parameter in the git checkout -b branch-name command: The **-b** parameter in the **git checkout** command is used to create a new branch and switch to it in a single step. The **checkout** parameter means to switch to a new branch, and the **-b** parameter means to create a new branch.

- **Git status command:** `git status`



```
Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)
$ git status
On branch add-player-hit
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/GameLogic/Snap.cs
        modified:   src/SnapGame.cs

no changes added to commit (use "git add" and/or "git commit -a")
```

Figure 2.3: Screenshots of git status command

In this step, in my terminal, I type the command **git status** to check the current state of my working directory and staging area in Git. It lets me see which changes have been staged, which haven't, and which files aren't being tracked by Git. Here two files are marked with a red colour, it means that I changed those files, and this is indicating my current state of the working directory.

Purpose of the git status command: The purpose of the **git status** command is to list out the files that are in my working directory but not yet tracked by Git. It shows which files have been modified, which are staged for the next commit, and which are not yet staged.

- **Git diff command:** `git diff`

```

$ git diff
diff --git a/src/GameLogic/Snap.cs b/src/GameLogic/Snap.cs
index 42894a1..40052cf 100644
--- a/src/GameLogic/Snap.cs
+++ b/src/GameLogic/Snap.cs
@@ -140,6 +140,10 @@ namespace CardGames.GameLogic
     _score[player]++;
     //TODO: consider playing a sound here...
 }
+
+ else if ( player >= 0 && player < _score.Length)
+{
+_score[player]--;
+}

     // stop the game...
     _started = false;
diff --git a/src/SnapGame.cs b/src/SnapGame.cs
index ec78e90..5366925 100644
--- a/src/SnapGame.cs
+++ b/src/SnapGame.cs
@@ -26,6 +26,22 @@ namespace CardGames
     {
         myGame.FlipNextCard ();
     }
+
+     if (myGame.IsStarted)
+{
+if ( SwinGame.KeyTyped (KeyCode.vk_LSHIFT) &&
+SwinGame.KeyTyped (KeyCode.vk_RSHIFT))
+{
+//TODO: add sound effects
+}
+else if (SwinGame.KeyTyped (KeyCode.vk_LSHIFT))
+{

```

Figure 2.4: Screenshots of git diff command

In this step, in my terminal, I type the command **git diff** to compare and understand the changes in a project. Here some lines are marked green, which indicates that what modifications have been made since the last commit but haven't been added to the staging area.

Purpose of the git diff command: The purpose of the **git diff** command is used to show the differences between various states of my Git repository. It lists out the changes between my current working directory and my staging area.

- **Git commit command:** git commit -am "meaningful message"

```

Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)
$ git commit -am "add player hit with left and
right shift, with score decrement for miss hits"
[add-player-hit 3287f33] add player hit with left and right shift, with score decrement for miss hits
2 files changed, 20 insertions(+)
Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)

```

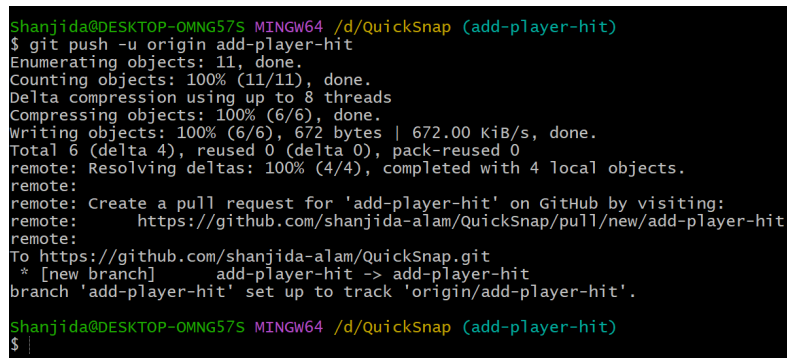
Figure 2.5: Screenshots of git commit command

Here, I type the git commit -am command to automatically stage all changes I have made in my code and attached a message to the commit explaining the changes.

Purpose of the git commit command: The purpose of the **git commit -am** "meaningful message" is to stage and commit tracked changes in the one step and attach a message to the commit.

Meaning of the parameter in the git commit command: **git commit** means commits changes to the local repository. **-a** means automatically stages any modified or deleted files (tracked files only). **-m "meaningful message"** means attaches a message to the commit explaining the changes.

- **Git push command:** `git push -u origin branch-name`



```
Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)
$ git push -u origin add-player-hit
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 672 bytes | 672.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
remote:
remote: Create a pull request for 'add-player-hit' on GitHub by visiting:
remote:   https://github.com/shanjida-alam/QuickSnap/pull/new/add-player-hit
remote:
To https://github.com/shanjida-alam/QuickSnap.git
 * [new branch]      add-player-hit -> add-player-hit
branch 'add-player-hit' set up to track 'origin/add-player-hit'.
Shanjida@DESKTOP-OMNG57S MINGW64 /d/QuickSnap (add-player-hit)
$ |
```

Figure 2.6: Screenshots of git push command

Here, I basically use the git push command to push my local commits to the remote repository and link my local branch to the remote branch.

Purpose of the git push command: The purpose of the **git push -u origin main** command is to push the local commits to the remote repository, typically for the first time, and set the upstream tracking for future pushes.

Meaning of the parameter of the git push command: **git push** this command indicates the committed changes from the local repository to a remote repository. **-u** this parameter sets an upstream tracking reference between the user's local branch and the remote branch. **origin** this is the default name for the remote repository we are pushing to.

3. GIT COMMAND LIST

Git Command	Meaning of this command
git clone https://github.com/shanjida-alam/QuickSnap.git	To create a local copy of a remote Git repository
git checkout -b add-player-hit	To create and switch to a new branch in Git
git status	Provides an overview of the current state of user working directory
git diff	Shows the differences between various states of user files in a Git repository.
git commit -am "meaningful message"	Staging modified files and committing them with a message
git push -u origin main	Used to push user local main branch to the remote repository and set up an upstream tracking relationship.

Table 3.1: Summary of Common Git Commands and Their Meanings