Git : 버전 관리 시스템 (버전 : 의미있는 변화)

00. 주요 내용

# 가장 중요한 부분들에 대한 지식을 갖고 그 이후로는 모르는 부분을 찾아내며 공부

# add 를 통해 파일을 stage 로 보내고 commit 을 통해 그 파일을 repository 에 저장

- stage : commit 을 대기하고 있는 파일들이 있는 장소

- repository : commit 이 완료된 파일들이 저장되는 장소

# 차이점 확인은 실수의 유무를 확인할 수 있는 중요한 수단

# 파일의 이름이 달라도 내용이 같으면 같은 오브젝트를 갖음 - 중복 방지 목적

# git 의 원리

- blob : 내용을 담고 있는 객체

- tree : 해당 파일의 이름과 blob 을 담고 있는 객체

- commit : 이전 tree (parent) 와 현재 tree 와 주석을 담고 있는 객체

- index : 작업한 모든 파일의 tree 를 저장

- repository : commit 내용을 저장

- working area --(add)--> index (staging area) --(commit)--> repository

# 충돌이 났을 경우에는 git 에서 수정 권한을 우리에게 위임해서 직접 수정해야 함

# 3 way merge : Base 와 나 그리고 다른 사람의 수정사항을 비교해 merge 하는 방식

# 원격 저장소 : 내 컴퓨터의 저장소와 연결된 다른 저장소

01. 기본 명령어

# clear : 화면 초기화

# pwd : 현재 위치하고 있는 디렉토리 표시

# cd A\_Directory : A\_Directory 로 이동

# mkdir Directory : Directory 생성

# ls -al : 현재 디렉토리 안에 있는 파일들 표시

# vim A.txt : A.txt 파일 열기 (없으면 생성)

- i : 수정 시작

- esc : 수정 완료

- :w : 저장 / :q : 종료

# cat A.txt : A.txt 파일 읽기

# cp A.txt B.txt : A.txt 와 같은 B.txt 를 생성

02. Git 관련 명령어 (--help 를 통해 명령어 사용방법 확인 가능)

# git config --global user.name name : name 으로 작업자 이름 설정 (최초 1회만)

# git config --global user.email mail : mail 로 작업자 메일 설정 (최초 1회만)

# git init : 현재 디렉토리에서 작업 시작

# git status : 현재 파일들의 작업 상태 확인

- 현재 index 와 이전 repository 의 commit 의 내용을 비교

# git add A.txt : A.txt 파일을 git에서 관리 시작 (시작할 때와 수정했을 때 모두 실행)

- 선택적으로 파일을 commit 할 수 있음

- commit 을 할 수 있는 준비상태로 만듦

# git commit : 버전을 관리하기 위한 vim 파일 실행

- commit 을 할 준비가 된 파일들만 해당 버전에 포함시킴

# git commit --amend : 이전 commit 의 주석 내용 수정

# git log : commit 이 제대로 되었는지 확인

- -p : 이전 버전과의 차이점을 모두 표시

# git diff : 이전 내용과의 차이점을 표시

- git diff A\_I..B\_ID : A 버전과 B 버전 간의 차이점을 표시

# git reset --hard A\_ID : A\_ID 버전의 이후 버전들을 삭제

# git reset --hard ORIG\_HEAD : 삭제한 버전을 복구

# git revert : 추후에 설명

# git branch A : A 라는 목적을 위한 작업 공간 생성

- branch 를 생성하면 생성한 시점의 모든 작업 내용을 받아 이어감

- -d A : A branch 삭제

# git checkout A : A branch 로 이동

# git log --branches --decorate : 모든 branch 의 log 를 보여줌

# git log A..B : A 와 B log 의 차이를 보여주는 기능 (B 에는 있고 A 에는 없는)

# git merge A : 현재 branch 로 A branch 의 내용들을 가져옴

# git stash : 하던 작업을 멈추고 다른 branch 에서 작업할 수 있게 해줌

- apply : 해당 작업을 다시 보여줌 / drop : 삭제 / pop : 둘을 모두 실행

# git reflog : 작업 수행 기록들을 보여줌

# (부모 디렉토리에서) git init --bare remote : 원격 저장소 생성

# (작업 공간에서) git remote add 원격 저장소의 경로 : 원격 저장소와 연결

# git push -u 원격 저장소의 주소 A : A branch 에서 원격 저장소로 자동으로 push 를 하게 setting

# git remote add A 원격 저장소의 주소 : A 라는 이름으로 원격 저장소와 연결

# git clone 원격 저장소의 주소 A : A 라는 이름으로 원격 저장소 복제

# git push : 로컬 저장소에서 원격 저장소로 데이터를 보냄

# git pull : 원격 저장소에서 로컬 저장소로 데이터를 가져온 후 병합

# git fetch : 원격 저장소에 있는 내용을 가져오기만 함

- 변경사항 확인이 가능하며 확인 후 merge 를 해야 함

# ssh-keygen : ssh 를 사용하기 위한 비밀번호 생성

# git tag Version A\_ID : A\_ID 를 가진 commit 의 버전(releases)을 Version 으로 표시

- A\_ID 를 생략하면 가장 최신 commit 의 tag 를 설정

- commit ID 대신 tag 를 이용하면 편함

# git tag -a Version -m “ABC” : Version + ABC 하는 주석까지 작성

# git tag -d Version : Version tag 삭제

# git push --tags : tag 도 원격 저장소로 push

03. GitHub

# Fork : 해당 오픈소스 등을 복제해서 내가 마음대로 사용할 수 있게 함

04. Plus

# Terrain\*.asset binary

# LightingData\*.asset binary

# Lightingmap\*.asset binary

