Git part of Presentation:

* What is git?
  + Popular Revision Control Software
  + Created by Linus Torvalds (creator of Linux) in 2005
* Why git?
  + Distributed version control
  + Flexible branching
  + Most popular
  + Widest support/third-party tools
* What is github?
  + Github != Git
  + Github is a Microsoft product that provides online hosting services for git repositories
  + Competitors to github include:
    - Bitbucket
    - Gitlab
    - self-hosted
* Git terminology
  + **Tracy Section start**
  + Clone
    - Copy a repository to your local machine
  + Add
    - Add new/modified files to the “staging area”
  + Commit
    - Commit to adding those staged changes to your locally-hosted clone of the git repository
  + Push
    - Upload your changes to the cloud-hosted repository (on github)
  + Pull (combination of the fetch and merge commands)
    - Download new commits from the cloud-hosted repo and update your locally-hosted clone to be “up-to-date”
    - You don’t want to git clone every time there’s an update
    - You should usually always git pull before you start working on the git repository, otherwise you’re more likely to deal with a divergent branch
  + Pull request (talk about how to set up a pull request system)?
    - You are requesting that the cloud-hosted repo pulls in your commits
    - A more polite git push
  + **Tracy Section end? Carson Begin**
  + Revert
    - Team lead has right to revert changes
  + Merge
  + Merge conflicts
  + Remote
    - The address of the cloud-hosted repository
      * “[git@github.com](mailto:git@github.com):user\_name/repo\_name”
      * “https://github.com/user\_name/repo\_name””
  + Branches
    - Note to presenter: discourage branching outside of Main in general (however TL6 will need to branch outside of main towards the end of the project)
  + Divergent branches
* Importance of a .gitignore at the root of the repo
  + Examples of necessary files
    - “X.cs in the Assets folder”
    - “X.cs.meta in the Assets folder”
  + Examples of unnecessary files
    - Executables (git will let you commit a large executable, but you will likely get an error when you try to push an executable to github because it has a limit on the size of each file)
    - Build/\*
  + .gitignore templates are your friend
    - May need to amend a template to exclude executables or other unnecessary files
* Four different ways to use git
  + Command line (PAT, SSH, HTTPS, etc.)
  + Web client
  + Desktop app
  + Text-editor/IDE integration (visual studio code)
* Tutorial on using git with visual studio code
  + How to create a repository? (most people won’t need to create a repository for this project as this is a responsibility of TL1 do that)
  + More importantly, how to clone a repository
* Live demo
  + One of use will create a git repo hosted on github and push code to it with Kevin in Coeur d'Alene
    - **Show a successful push**
    - **Show an unsuccessful push that results in divergent branches**
    - Show an unsuccessful push that results in merge conflicts
    - **Show git revert**
  + Possibly demonstrate the CLI version of git if there’s spare time
* Best practices (to fill up the time)
  + You should ideally never git push --force
  + Each commit should have changes that are related
    - A function that prints a name is not related to a function that reads a file into memory (probably) and shouldn’t be included in the same commit
  + Make small incremental commits
    - Reduces the opportunity for merge conflicts to arise
    - If something breaks, it’s much easier to identify where things went awry
  + Write descriptive commit messages
    - Poor example: “Fixed bug”
    - Good example: “Fixed issue where camera didn’t reset on player respawn”
  + Have a consistent commit message style
    - Example:
      * Capitalize the first letter of the message
      * Start with a verb

SA and Storyboarding Part of Presentation:

* Systems Analysis presentations are happening Feb. 14
* All groups will need to have the following completed:
  + RFP
  + Champion Document
    - Use case scenario and diagram
    - Data flow diagram
    - Rough unit test plan (we will not be taught this by the time the document is due)
    - Gantt chart
    - Pert chart
  + Presentation
    - Storyboard
    - Global use case
    - Context diagram
    - Diagram 0
    - Individual use case diagrams

Notes from BC Meeting:

* Team lead can revert anyone’s changes for the sake of fixing the game before presenting
* Suggest that the repos should be private for this project
  + If any of the group members would like to put this project in their portfolio, suggest they speak with their respective TL1 in the future about forking/making the core repo public
  + Suggest them to add a license to any public git repositories they create
* Documents will be turned in through the Docs folder in the git repo (instead of on canvas)
* Part 1: Unity (Kevin leading with Logan following)
  + Kevin presenting
  + Go through creating repo through vscode from scratch
  + Invite Logan (Logan’s email: finley4.logan@gmail.com)
  + Add .gitignore for Unity
  + Create a Unity project
  + Initialize the repo in VSCODE
  + Push unity project to git
  + Go into the examples:

1. Two successful pushes
   1. Each upload a file of our name to the assets folder
2. Git divergent branch
   1. Logan: push a change to the repo
   2. Kevin: without pulling, try to push a different change
   3. Kevin: get divergent branch error
   4. Kevin: pull first and then push
3. Git revert
   1. Logan: push a bug
   2. Kevin: git revert and swear
   3. success!

* Part 2: C++ Command line (Logan leading uwith Kevin following)
  + Logan presenting
  + Mention authentication methods (SSH, etc.)
  + Create a repo on github
  + Invite kevin as collaborator (Kevin’s email: kevinrwing@gmail.com
  + Logan initialize an empty repo
  + Logan manually add remote and set remote-url
  + Logan create main.cpp and cout << “Hello from Logan” << std::endl;
  + Logan: push
  + Kevin: pull
  + Kevin write another cout statement
  + Kevin: push
  + Logan: pull
  + Logan: compile and execute the code