GDBS/SVBS/CSBS
Midterm Project Overview

Goal

- These courses are designed to familiarize students with the development process by implementing and completing a multiple month project as a team.

Who are we

Fernando De La Cruz
Program Director (GD):
fdelacruz@fullsail.edu

Rebecca Leis
Program Manager (CS):
rleis@fullsail.edu

Shawn Kendall
Department Chair:
skendall01@fullsail.edu

Rebecca Carroll
Department Chair:
rebeccac@fullsail.edu

Steve VanZandt
CD of AHI:
svanzandt@fullsail.edu

Rod Moye
CD of PP2:
rmoye@fullsail.edu

John O'Leske
CD of PP3:
joleske@fullsail.edu
JohnOLeskeFS#4268

Full Midterm Project Process

Full Midterm Project Process

Applied Human Computer Interaction	Project and Portfolio II	Project and Portfolio III

- Pre-Production
  - Design Document
  - Product Backlog

- AHI Topics
  - Nielsen's heuristics
  - Usability
  - UX
  - UI
  - Emerging tech

- Core Functionality
  - Critical game systems
  - Interface and UI creation

- First Use/Playable
  - Usable / Playable experience
  - General Purpose / Fun Factor Achieved

- Alpha
  - Full Functionality

- Beta
  - Content complete
  - Balancing

- Finalizing
  - QA process
  - Presentation

Project Expectations - Games

Game Dev and Simulations and Visualization student teams

Game Expectations   - Scope

- Medium/Indy scoped game

- Game similar to game from NES, SNES/Genesis era, or mobile and web platforms game tend to work best

- Focus on functionality over assets

Game Expectations   - Minimums

- All games must have at least 15 minutes of engaging and varied play
  - Most have far more than 15

- Must contain at least one single player mode
  - To ensure the game can be easily demoed

Game Expectations   - Buildable

Select a game with your capabilities in mind

- Heavily story driven game
  - Someone one the team should be a writer

- Aesthetic as a hook (unique art styles)
  - Someone on the team should be able you source those assets

- 3D Animation heavy game
  - Someone on the team should be able to animate

- Game with 50 unique levels
  - Someone on the team should have the skills of a level designer

Game Expectations   - Platform Support

Must support a secondary platform
- ○ WebGL version exported to a webpage
- ○   or published to a web portal

or
- ○ Playable on an android device (tablet or phone)

- Keep platforms in mind when making design and production choices

- Keep project sizes low
  - ○ Overall project size at the end should be under 1GB to make those platforms viable.
  - ○ At least under 512 MB (for platform support)

Game Expectations   - Expo

- Games will be presented at the FPS Expo

- First Thursday of the month after PP3
  - 11am-1pm

Project Expectations - Apps

Computer Science student teams

App Expectations

- Medium scoped application

- You can use any API's you want as long as you are coding in Java, Kotlin, C#, or some kind of mid-level programming language

- Some examples of apps that have been created in the past are
  - A basketball pickup game finder app
  - A movie review and suggestions app
  - An app to find non chain locally owned restaurants
  - A D&D companion app to track character sheets and player progress during campaigns
  - A closet inventory management app that tracks what is dirty/clean, preset outfits, and storage locations

App Expectations   - Buildable

Design and pitch an app with your capabilities in mind

- Required access to a specialized data set like a database of all movies or SKUs of a grocery store
  - Do we have access to that data to make this application possible?

- Established community required for success
  - Do we have a way to bring in the first wave of users to create that community?

App Expectations   - Platform Support

All apps must support a mobile platform

- Usable on tablet or phone (apk or iphone app)

- Keep platforms in mind when making design and production choices

- Keep project sizes low
  - Overall project size at the end should be under 1GB to make those platforms viable.
  - Under 250MB preferred

Developer Expectations

Expectations: Problem Solving

Put less importance on knowing things ahead of time.

- The job IS problem solving
  - On the job, you learn things just-in-time.
  - You must be able to figure out solutions on your own.
  - Unity Documentation is surprisingly good

- It's not memorization and regurgitation. It is finding a way that works from multiple possible solutions

Expectations: Communication

Communication will be a challenge

- Mandatory schedule
  - CDs will see you 1 to 3 times a week
  - That isn't enough to keep communication open

- Stay connected with us
  - If something breaks, tell us we can help fix it
  - If there are issues with finding assets, tell us we can probably help
  - If something awesome changes in the project, tell us we would love to hear
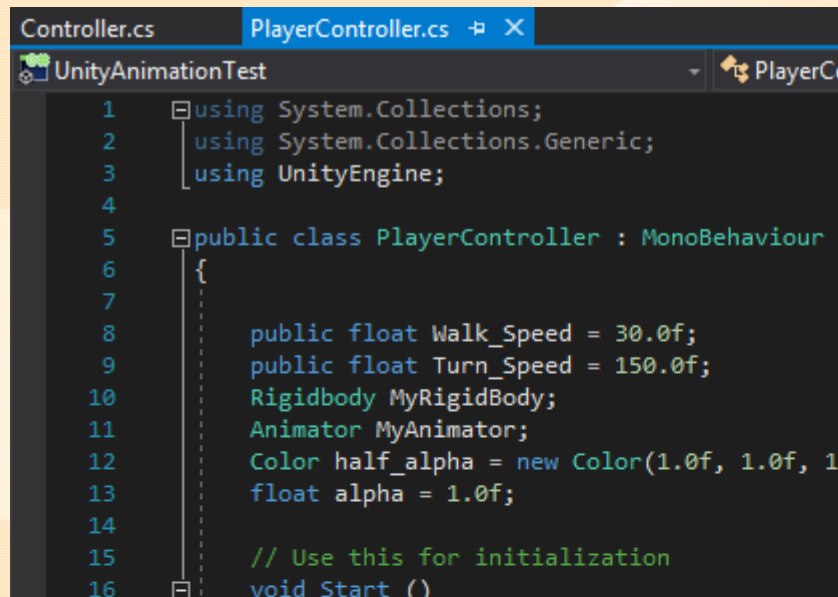
Expectations: Teamwork

Work with each other

- Have a set schedule
  - Check in daily
  - Set a schedule for everyone to work TOGETHER
  - You will always get more done as a group

Expectations: Be a developer

We will be doing a lot of different jobs

- Programmer
    - Creating functionality
   - Researching the engine or APIs
    - Fixing bugs

```csharp
Controller.cs        PlayerController.cs
UnityAnimationTest                                    PlayerC
   1    using System.Collections;
   2     using System.Collections.Generic;
   3     using UnityEngine;
   4
   5    public class PlayerController : MonoBehaviour
   6    {
   7
   8        public float Walk_Speed = 30.0f;
   9        public float Turn_Speed = 150.0f;
  10        Rigidbody MyRigidBody;
  11        Animator MyAnimator;
  12        Color half_alpha = new Color(1.0f, 1.0f, 1
  13        float alpha = 1.0f;
  14
  15        // Use this for initialization
  16        void Start ()
```

Expectations: Be a developer

We will be doing a lot of different jobs

- Designer
  - Defining mechanics and functionality
  - Scripting interactions
  - Level designs (games)
  - Balancing values (games)
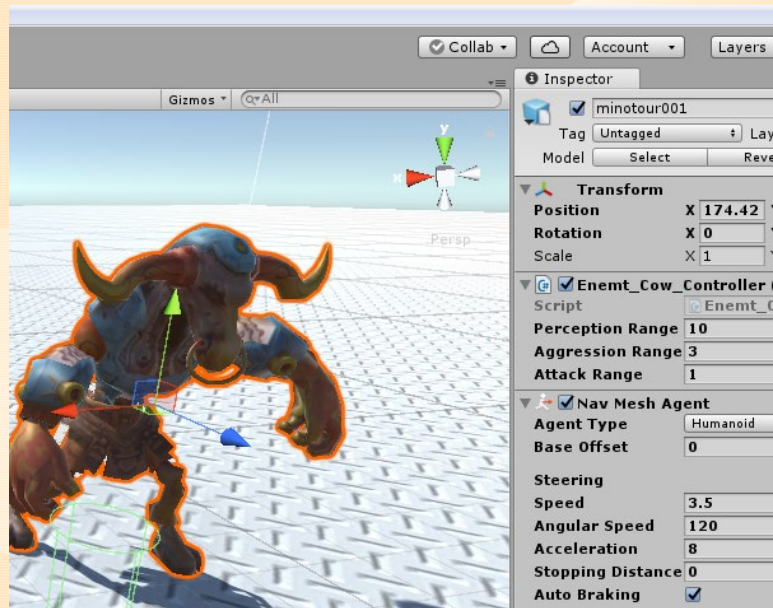
Expectations: Be a developer

We will be doing a lot of different jobs

- Producer
  - Defining expectations
  - Writing and following a Scheduling

🖥 **Timed Spikes**

in list World Obstacles

☐ Recurring   Add #tags ▾   S/E & More ▾

Labels

Difficulty: Easy ＋

Description

Edit

Intent:

- Spikes that pop into and out of the ground based on a timer that damage characters that collide with it.

☑ **Test Cases / Acceptance Criteria**   Delete...

0%

☐ Do spikes damage the player when the collide with the top of them?

☐ Do spike not damage the player if the touch only the side of them?

☐ Do the spikes go into and out of the ground based on a timer?

☐ Do spikes not damage while in the ground?

Expectations: Be a developer

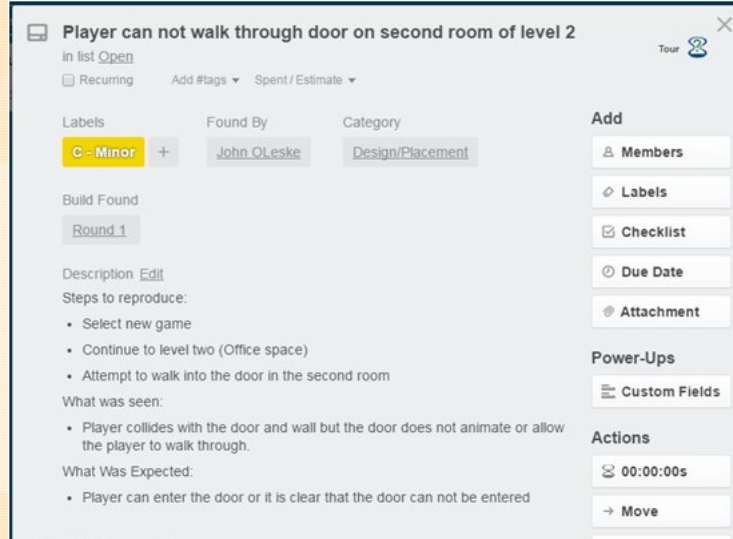We will be doing a lot of different jobs

- Artist
  - Adding assets to product
  - Setting up animations
  - Creating simple assets (programmer art, placeholders)
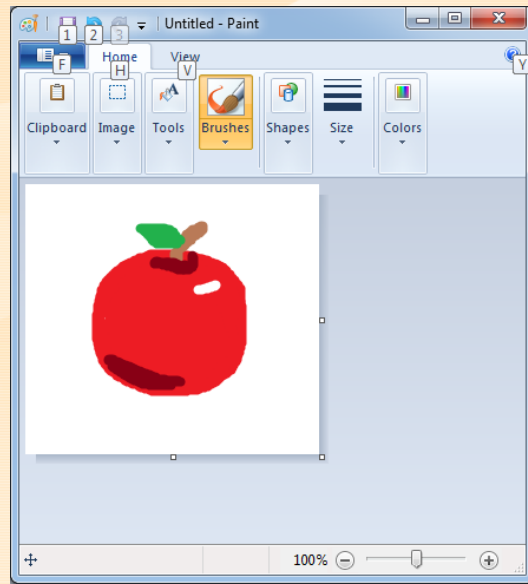
Expectations: Be a developer

We will be doing a lot of different jobs

- Quality Assurance Tester
  - Testing and reporting bug

Project Policies

Academic honesty

"Projects/Assignments: Students are expected to be honest and produce their own projects/assignments according to the specifications of their Course Director. **They must work solely on their projects/assignments unless otherwise noted by this Course Director.** Work submitted by our students is assumed to be a student's own thoughts, idea, and words. Discovery of the contrary will result in immediate consequences. **For group projects, all students whose names are submitted with the project are responsible for the content and will be subject to disciplinary action should plagiarism be discovered.**"

- Student Manual, page 17

Academic honesty: Midterm Specifics

- Code/Functionality

- All functionality in the final product must be created by a student team member

- Any functionality not included in the unity installation must be authored by a student team member
  - Scripts/Code
  - Prefabs/Objects/Scenes/Interfaces created in the development environment

If it can be made in the development environment, you are expected to make it.

(For game projects you may not use the unity asset store to add functionality to the project.)

Academic honesty: Midterm Specifics

- Assets

- Assets authored by non-student team members may be used as long as there are legal rights to use the assets
  - Textures/sprites/stock photos
  - Audio/sfx/music
  - Models/meshes
  - Animations

- Any assets used that was not created by a student team member must be have their source credited in the game's credits

Academic honesty: Levels

- Level 1 (0 score on the assignment and month's professionalism, conduct probation, and suspension):
  - Directly copying work from another source and submitting 1 it as one's own.
  - Submitting work completed by another individual or student as one's own.
  - …

- Level 2 (0 score on the assignment and month's professionalism, and conduct probation):
  - Completing any work for another student that fulfills an academic requirement.
  - Knowingly furnishing false information to an instructor or any other representative of the University.
  - Repeated violations that fall under the Levels 3 or 4 headings.

- Level 3 (0 score on the assignment and month's professionalism):
  - Submitting work that was turned in from another course or a previous attempt at this course without prior approval from the course instructor.
  - Significant omission or misuse of citation and/or references in course work.
  - In group work, including one's name to "tag along" on work of other team members in which he/she did not significantly contribute.

- Level 4 (0 score on the month's professionalism):
  - In group work, allowing a team member to include his/her name to "tag along" on the work of other team members despite having not significantly contributed to that work.

Academic honesty

If you are unsure, ask

- There can be some grey areas with this as time goes on.

- If an item is in question, bring it up to staff

Tools

Discord

discordapp.com

- Keep communication open while working remotely
- Share files that are not part of the project
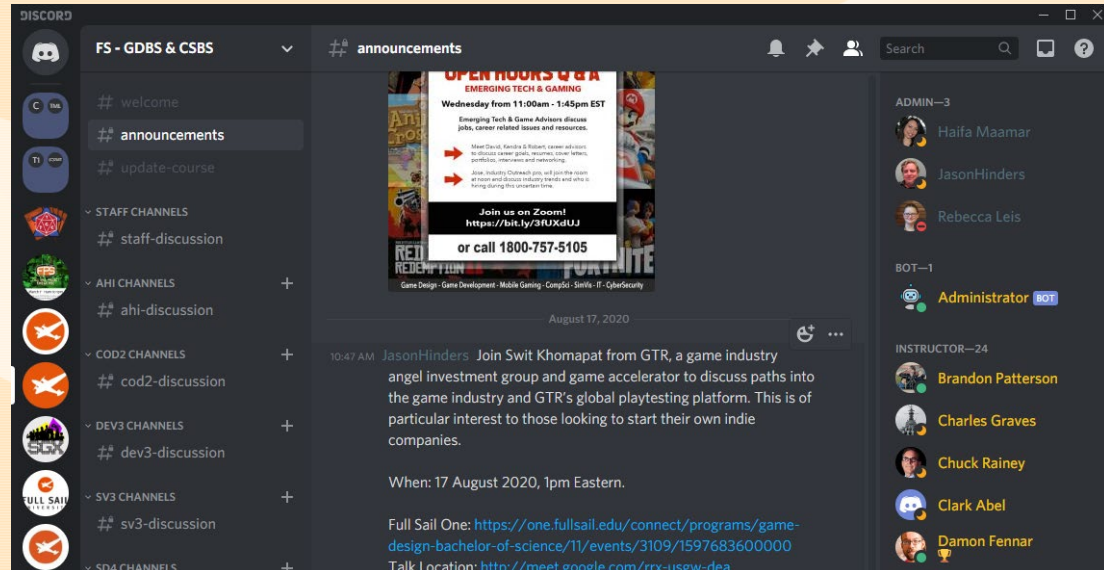
Trello

trello.com

- Part of our design space
- Our task management system

Plus for trello (Chrome plugin)

plusfortrello.com

- Task ownership
- Task hour tracking

TortoiseGit

tortoisegit.org

- Our Git versioning client
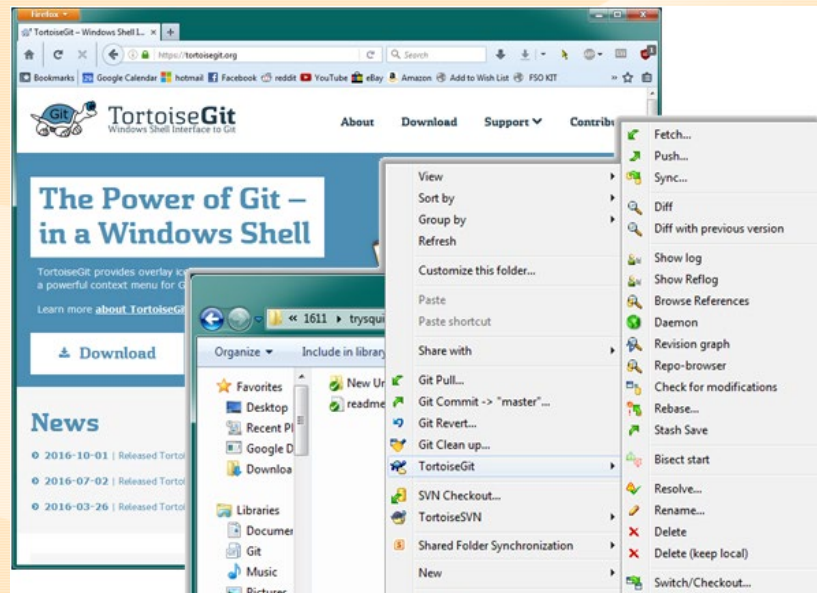
Git Hub

github.com

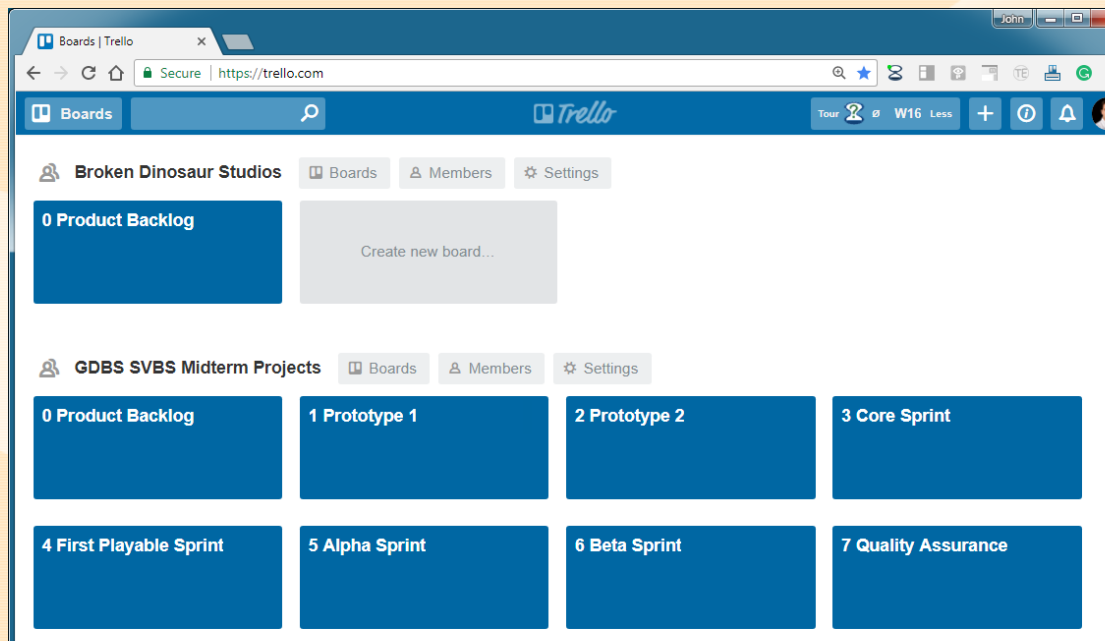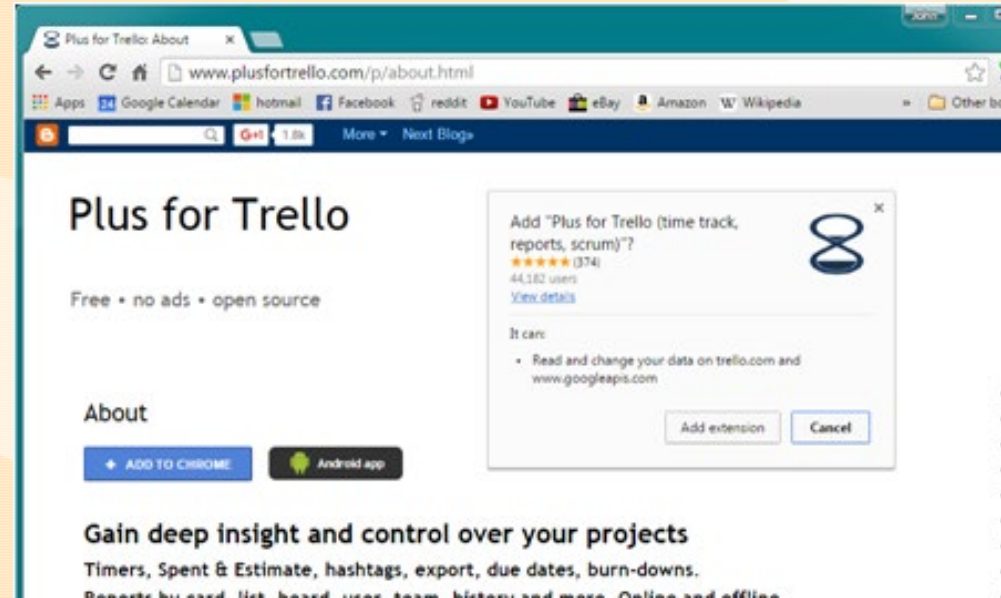- Git server will be provided through GitHub
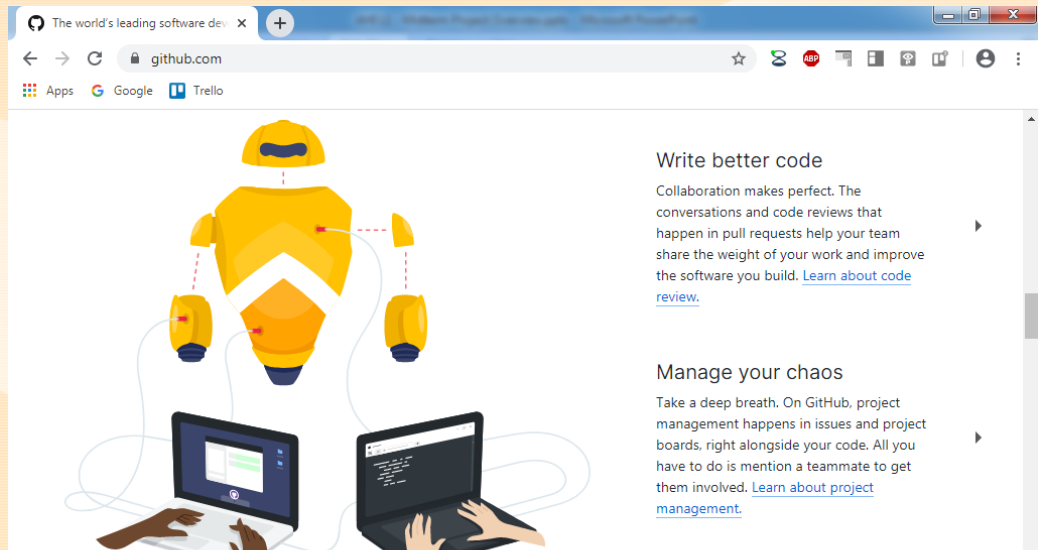
Git LFS

git-lfs.github.com

- Allows github to receive large sized files
- Larger than 50MB (github max file file)
- (Zipped milestones, Video files, Resource packages …)

Git Large File Storage

An open source Git extension for versioning large files

Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

Unity3D

(For Game Dev and Sim Vis teams)
- Unity3d.com

- Our dev environment
- Scripting in C#

- Ensure each team member is using the same version
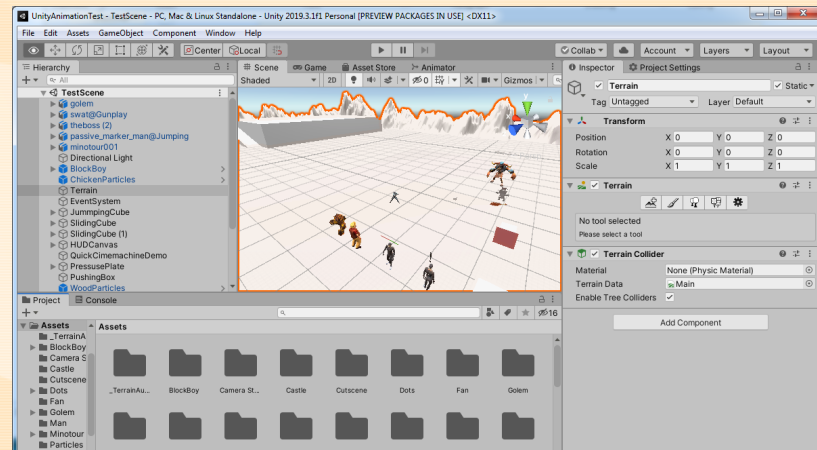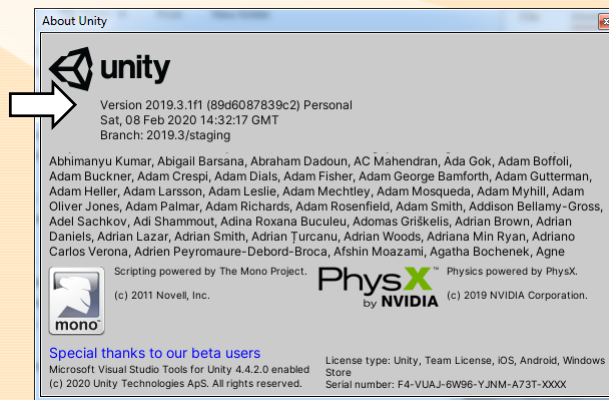
Application Development Environment

- (For CS teams)

- The choice of development IDEs is heavily dependent on the application being worked on.
  - Mobile APP, WebApp, PC Executable all significant different requirements and needs for IDEs.
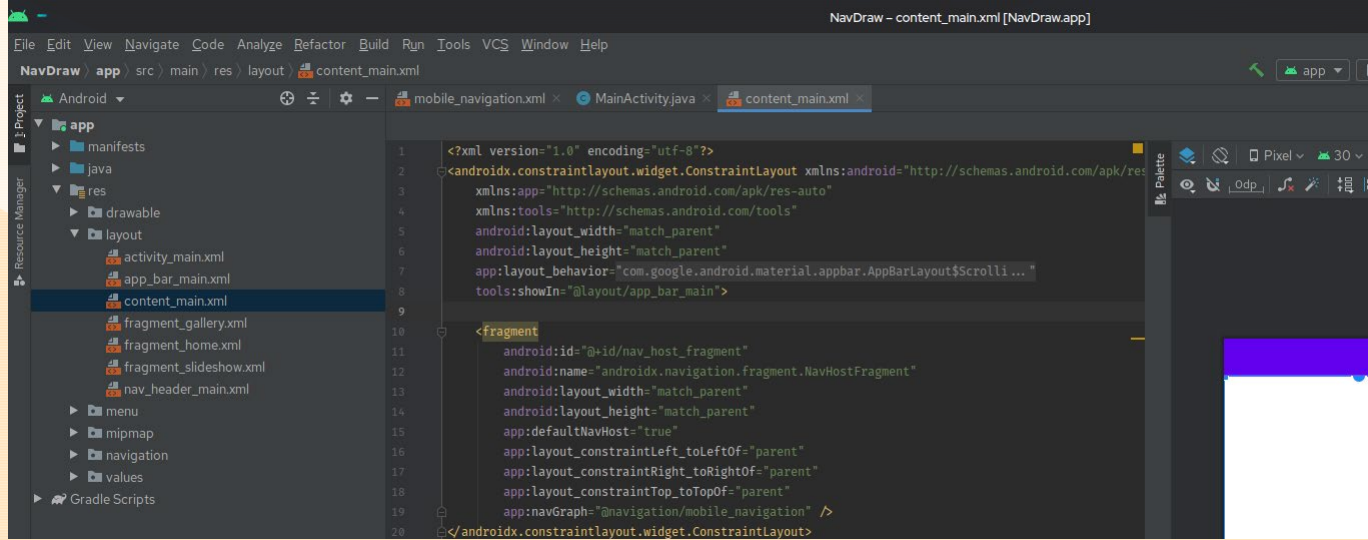
Install, Create Accounts, or Confirm

Sent to joleske@fullsail.edu

- Unity
  (unity3d.com)
  Everyone on same version

- TortoiseGit
  (tortoisegit.org)

- Git framework
  (git-scm.com)

- Git LFS
  (git-lfs.github.com)

- Discord
  (discordapp.com)

- Trello
  (trello.com)
  Accounts created
  (Username is after the @ on
  your profile page)

- Chrome
  (google.com/chrome)

- Plus for trello plugin
  (plusfortrello.com)
  Sync method set to
  "Recommended- Store inside Trello
  (S/E Trello card comments)"

- Github
  Accounts confirmed

<Activity> Form Teams

Form team
- 3-5 students per team
- Will be working with each other for months

Collect the info
- Team
  - Team name
  - Discord channel

- For each team member
  - Full name
  - Trello username
    - (Username is after the @ on your profile page)
  - Github username
  - Discord username
  - Email

<Activity> Discuss ideas

I will be setting up servers and documents for all of the teams

Think of a game or App

- Discuss as a team what you would like to develop

<Activity> Discuss ideas

While I create team documents and servers form a basic elevator pitch for your game or application

- Elevator pitch.
    - A one or two sentence description of the product.
    - This should encapsulate the hook or selling point of the product.
    - Why would someone want to buy and use this product over other products?
    - Why should a company be willing to invest in making this product?

- When I am done, I will look for each team to present their basic pitch

- We will be looking for a more cleaned up version as part of the "Developing a Pitch" activity on FSO before next lecture

Clone The Repo

<Activity> Clone the project folder

Clone the repo

- After the clone the project folder should look like this
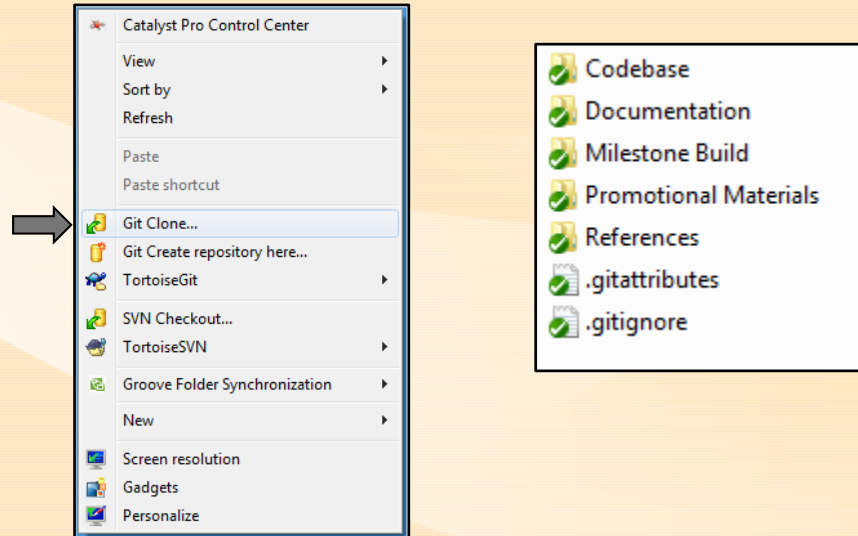
<Activity> Let's explore the files

- .Git
  - The hidden folder containing the local repo
- Codebase
  - Your unity project
- Documentation
  - Preproduction and production documents to be filled out
- Milestone Build
  - A single most recent good build of the game
- References
  - Lecture power points, rubrics, FAQs, Project archives, Asset Sources
- .gitignore
  - The file that defines what should not be pushed to the repo

.git
Codebase
Documentation
Milestone Build
References
.gitignore
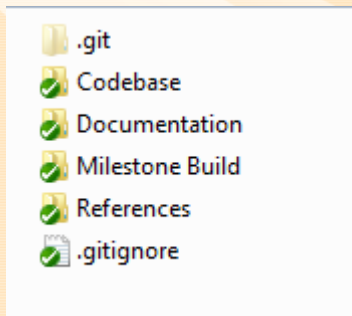
Pre-Production

The purpose of Preproduction

- Pre-production
  - Make decisions on the product
  - Prove the validity of those decisions with rapid iteration of prototypes
  - Throw out what doesn't work and keep what does
  - Document the full scope based on the above

- Production
  - Make the rest of game based on the above

The purpose of Preproduction

- As hard as preproduction is, production is much harder without it
  - And more expensive in the long run

- Fixing problems on paper takes less time and resources than implementing the system to see it doesn't work.

Top-Down Versus Bottom-up

- Top down
  - Starting at the big picture and breaking down into smaller and smaller components
  - What do we want? Now how can we make that happen?

- Bottom Up
  - Defining the components and integrating and combining to get to a bigger picture
  - What can we do? Now what can we make with that?

- Important to look at the game from both points of view

The Design Document

Design Document

- Top-down design
  - Defining the intent of the product as a whole and breaking that down into its various aspects
  - Maintain the overarching vision of the product as a whole

- Solidify a single vision
  - Collaborate as a team to agree upon the content
  - Solve problems before they occur in production
  - Understand the overall scope of the project

- Maintain the vision
  - Reference decisions that were already made
  - Avoid conflicting designs
  - Avoid reinventing what was already there
  - Bring new team members up to speed

Design Document

Agree on the core design

- Overall goals or the product
- The selling points of the product
- Comparable products
- Overarching systems, features, and mechanics

Design Document

- Template document has been provided for you
  - In your repository's documentation folder

  - This absolutely must be customized for the game you are making
    - All games and Apps are different
    - Must describe the aspects of the product you are making

Trello

Our task management platform

Trello: Our task management platform

- Bottom‑up design
  - Defining the individual aspects of the product
  - Break down into deliverable tasks all the work that needs to be done to create this application.

- Brainstorm features
  - Break down into deliverable tasks all the work that needs to be done to create this application.
  - This is done now so once development starts in earnest you pick a card put it into progress and create it without needing to design or research aspects on the fly.

Trello: Our task management platform

- Create new card on trello for each feature to be added to the game
  - User login systems
  - Menu pages
  - Map interfaces
  - Backend systems
  - Data management
  - Player action
  - Characters
  - Enemy types
  - Items
  - Weapons
  - Power ups
  - Game modes
  - …

Trello: Our task management platform

Our team's Trello boards

0a Core Product Backlog

0b Extended Product Backlog
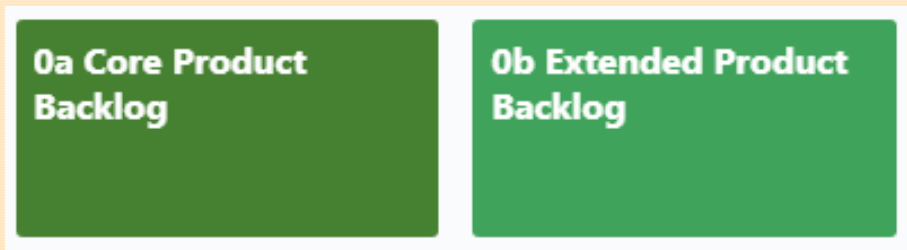
- "0a Core Product Backlog"
  - All items and features required to create the product's vertical slice
  - M.V.P. Minimum viable product

- "0b Extended Product Backlog"
  - All the wish list, would be cool to have, stretch goals… of the product

Trello: Our task management platform

- The trello backlog is made with the understanding that it generally contains more work than you think you could actually finish by the end.

- If things go better than expected the stretch ideas are already waiting to go into production

- if we don't get to everything that is also fine

Trello

Generally, when looking at an application created by a small team of developers over a couple months having many cards is expected.

Shoot for 100-150 items/features across the 2 boards
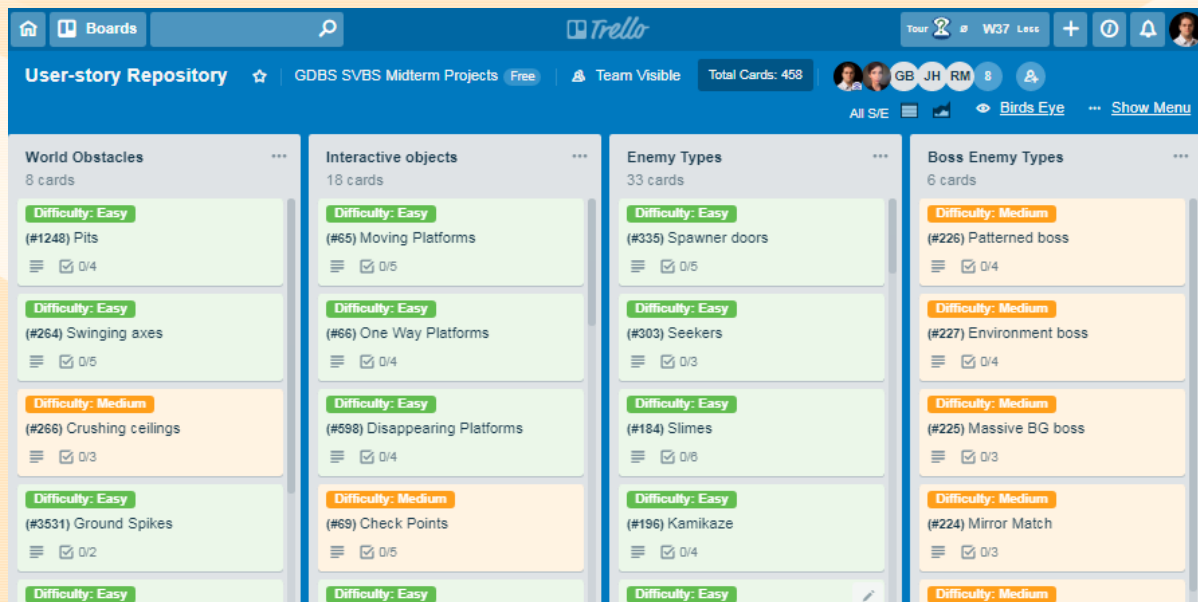
Risk Assessments

Risk Assessments

- All known risks identified on the project
  - How dangerous the risk is to the overall project plan
  - What actions must be taken to mitigate the risk
  - Deadline for those actions
  - Contingency plan

- Template document has been provided for you.
  - In your repository's documentation folder
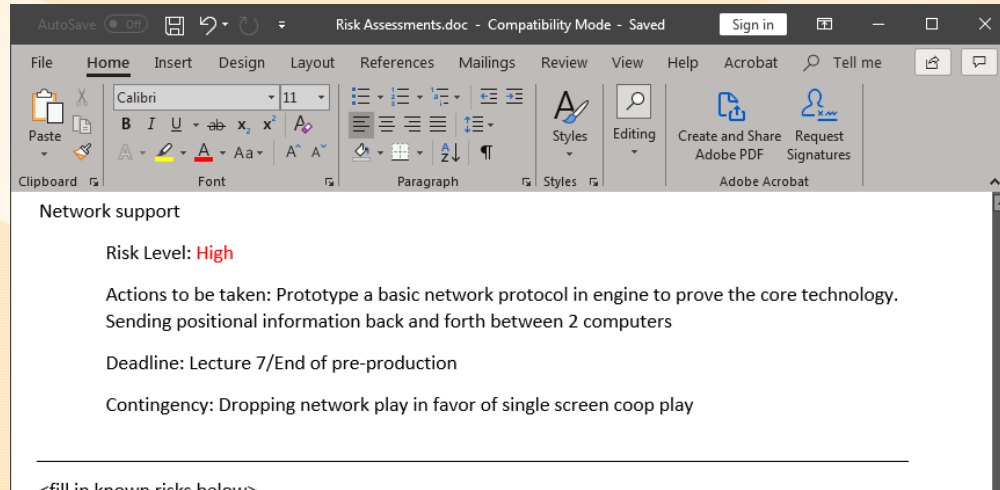
Coding Standards and Best Practices

Coding Standards and Best Practices

The team agreed to a standard
- Commenting
- Naming conventions
- File Formatting
- Indentation and Spacing
- General Programming practices

- Template document has been provided for you.
  - In your repository's documentation folder
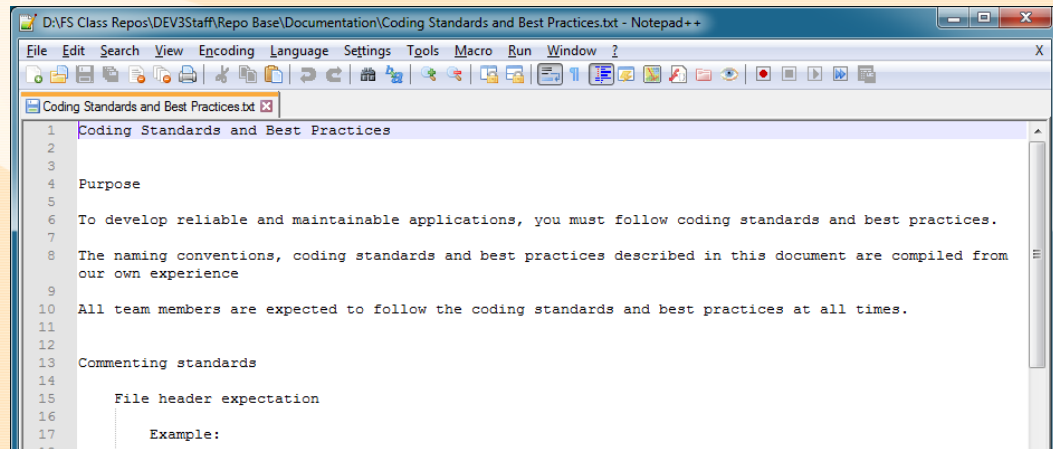
Living Documents

Living Documents

- Game/software design documents are often "living documents,"
  - They are constantly changing based on tweaks in the design/development of your product.

- Product backlogs are also constantly updated during the pre-production and production process.
  - A new feature is requested
  - large bug is found
  - product direction changes
  - New cards are added to the product backlog to track that work

- While working on tasks and design changes through the month the documents should be updated for any changes made

Version Control

Git Basics

- Understand the system behind the interface

  - Three main sections to pay attention to
    - Working files
    - Local Repository
    - Remote repository

    - (actually 4 with the staging phase but that mostly handles itself)

Working Files	Local Repo	Remote Repo

		`Level1.unity   (v1)`
		`Level2.unity   (v1)`
		`Level3.unity   (v1)`
		`MainMenu.unity (v1)`
		`…`
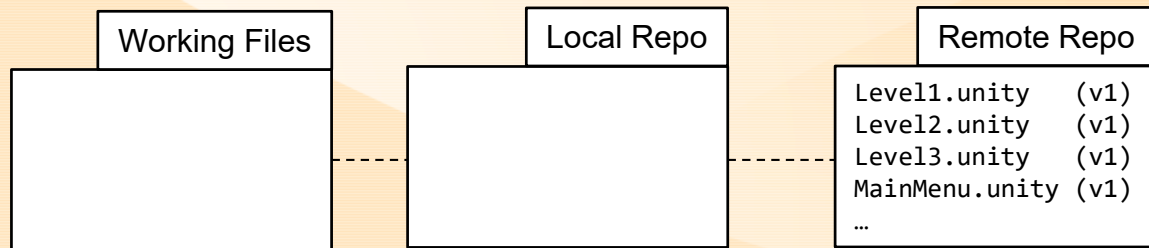
Clone

Contributors need to clone the repository to start working on the shared files.

 Git Clone...
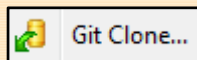
- Get a bring remote repository into the local repository

- Populate working files from local repository

Working Files	Local Repo	Remote Repo
Level1.unity (v1)	Level1.unity (v1)	Level1.unity (v1)
Level2.unity (v1)	Level2.unity (v1)	Level2.unity (v1)
Level3.unity (v1)	Level3.unity (v1)	Level3.unity (v1)
MainMenu.unity (v1)	MainMenu.unity (v1)	MainMenu.unity (v1)
…	…	…
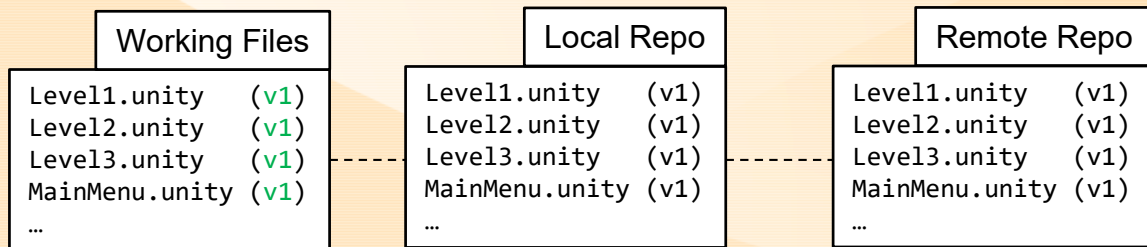
Commit

Once changes to the files has been made that work need to be committed

Git Commit -> "master"...
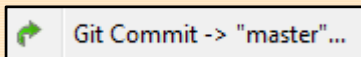
- Commit saves the changes to the local repository

- Once committed there is a timestamp of the files that can always be returned to

Working Files		Local Repo		Remote Repo	
Level1.unity	(v1)	Level1.unity	(v1)	Level1.unity	(v1)
Level2.unity	(v2)	Level2.unity	(v2)	Level2.unity	(v1)
Level3.unity	(v1)	Level3.unity	(v1)	Level3.unity	(v1)
MainMenu.unity	(v3)	MainMenu.unity	(v3)	MainMenu.unity	(v1)
…		…		…	
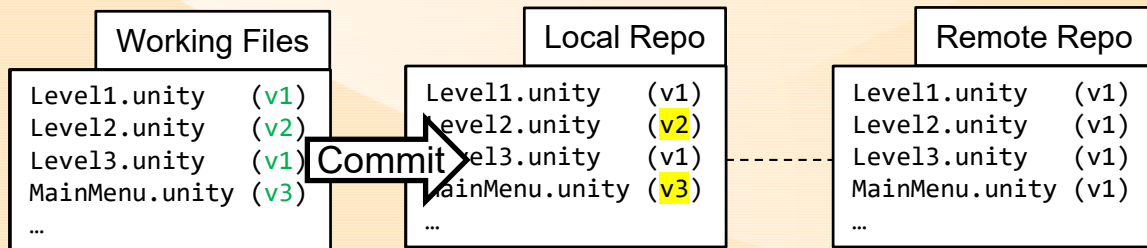
Commit

Push

Push integrate changes onto the remote repository
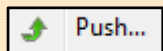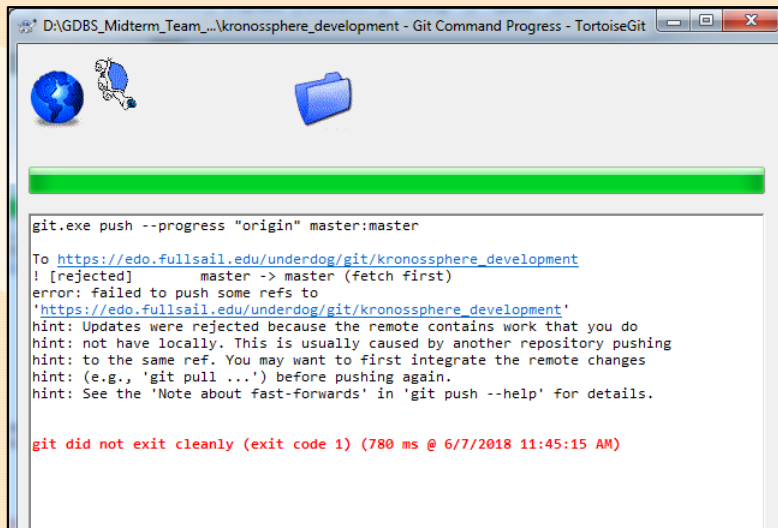
- Non committed work does not get pushed

- If a file hasn't been added it doesn't get committed

Working Files		Local Repo		Remote Repo	
Level1.unity	(v1)	Level1.unity	(v1)	Level1.unity	(v1)
Level2.unity	(v2)	Level2.unity	(v2)	Level2.unity	(v2)
Level3.unity	(v1)	Level3.unity	(v1)	Level3.unity	(v1)
MainMenu.unity	(v3)	MainMenu.unity	(v3)	MainMenu.unity	(v3)
…		…		…	
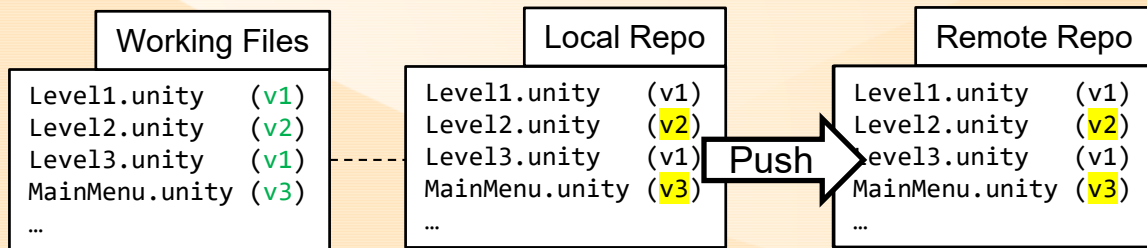
Push

Push: Changes on remote error

"error: failed to push some refs to…
 hint: updates were rejected because the
 remote contains work that you do not have
 locally"

(Read all of the error message, not just the
red text)

Push

You can't push if there are changes on the remote server that you do not have on your local repo

Working Files	
Level1.unity	(v1)
Level2.unity	(v2)
Level3.unity	(v1)
MainMenu.unity	(v3)
…	

Local Repo	
Level1.unity	(v1)
Level2.unity	(v2)
Level3.unity	(v1)
MainMenu.unity	(v3)
…	

Remote Repo	
Level1.unity	(v1)
Level2.unity	(v1)
Level3.unity	(v2)
MainMenu.unity	(v1)
…	
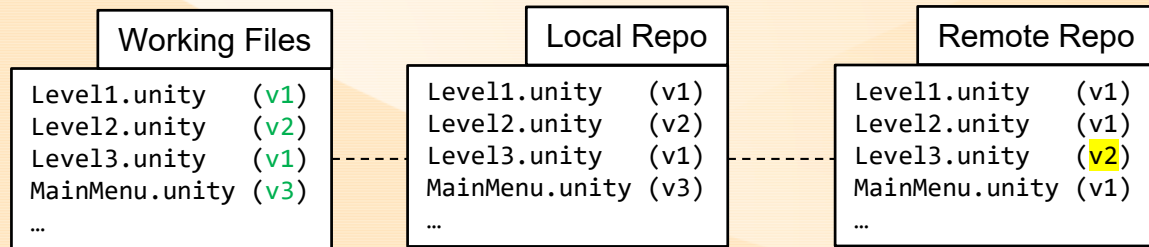
Pull

Pull changes from the remote and integrate them onto your build

| Git Pull... |
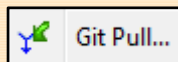
- Changes get integrated into the local repo

- If integration is good the working files change to match

Working Files	Local Repo	Remote Repo
Level1.unity (v1)	Level1.unity (v1)	Level1.unity (v1)
Level2.unity (v2)	Level2.unity (v2)	Level2.unity (v1)
Level3.unity (v2)	Level3.unity (v2)	Level3.unity (v2)
MainMenu.unity (v3)	MainMenu.unity (v3)	MainMenu.unity (v1)
…	…	…

Pull ⟵   Pull ⟵
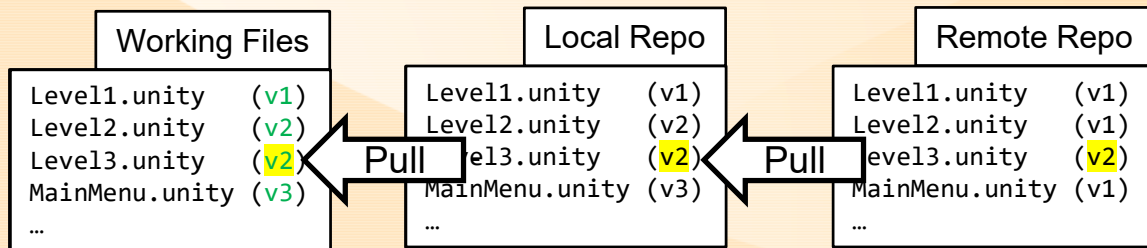
Pull

Do not pull with unity open

Git Pull...
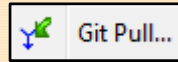
● Unity is always updating and recompiling based on changes in the files

● If you pull with unity open unity and there is an error unity will attempt to recompile with the error and break

Working Files		Local Repo		Remote Repo	
Level1.unity	(v1)	Level1.unity	(v1)	Level1.unity	(v1)
Level2.unity	(v2)	Level2.unity	(v2)	Level2.unity	(v1)
Level3					
MainMe					
...					

Unity 2018.3.4f1 Personal - Untitled - WorkingTitle - PC, Mac & Linux Standalone <DX11>

File Edit Assets GameObject Component Window Help

Center Local

Collab Account Layers

Hierarchy   Scene   Game   Asset Store   Animator   Inspector   Navigation
Create Q All   Shaded   2D   Gizmos Q All

Pull: Uncommitted work error

"error: Your local changes to the following files would be overwritten by merge… please commit your changes or stash them before you merge."
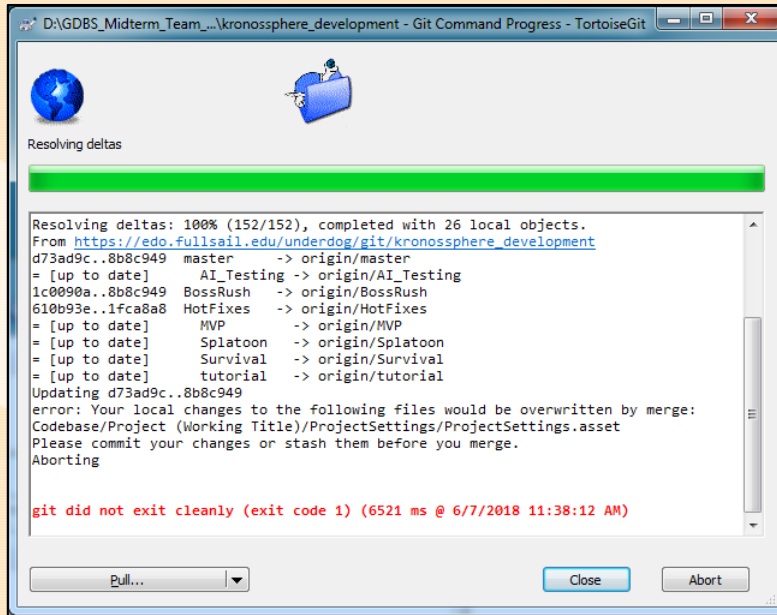
Uncommitted work error

Resolution

- You have changes to your working files that have not been committed
- Once you commit your changes you can pull and get the remote changes

| Git Commit -> "master"... |
| Git Pull... |

Working Files		Local Repo		Remote Repo	
Level1.unity	(v2)	Level1.unity	(v2)	Level1.unity	(v1)
Level2.unity	(v2)	Level2.unity	(v2)	Level2.unity	(v1)
Level3.unity	(v1)	Level3.unity	(v2)	Level3.unity	(v2)
MainMenu.unity	(v3)	MainMenu.unity	(v3)	MainMenu.unity	(v1)
…		…		…	
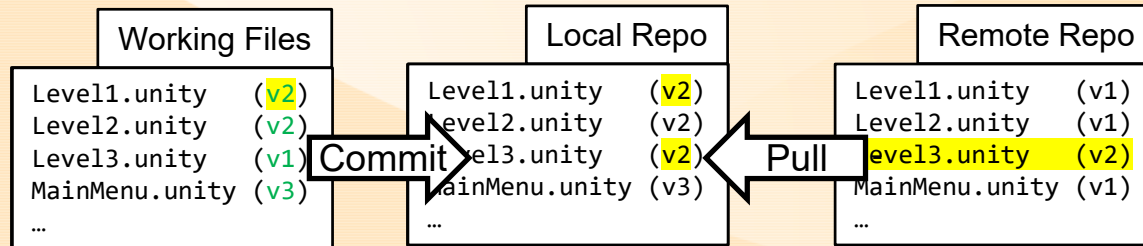
Commit →

← Pull

Conflict

"Automatic merge failed: fix conflicts and then commit the results"
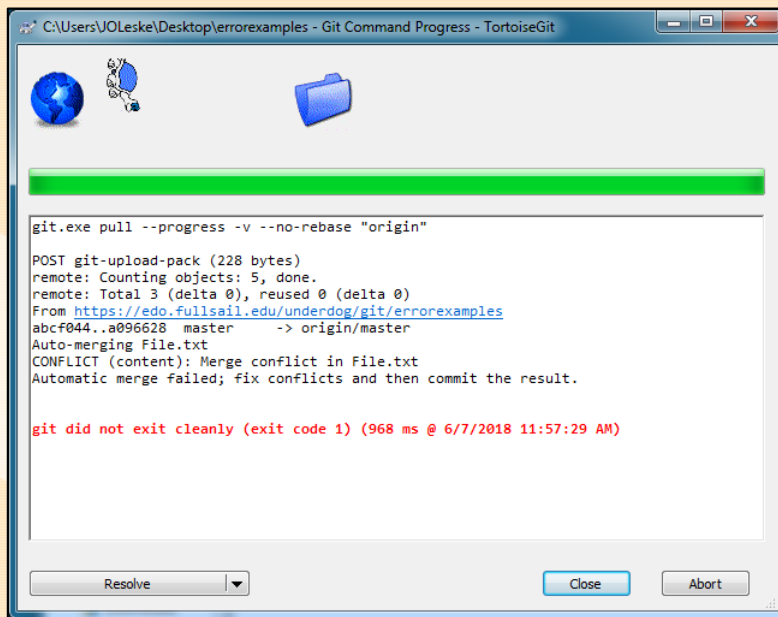
Conflict

Conflicts are created when 2 pull and then modify the same file

These two Devs already have a conflict though they don't necessarily know it

Working Files A	
Level1.unity	(v1)
Level2.unity	(v2a)
Level3.unity	(v2)
MainMenu.unity	(v1)
…	

Local Repo A	
Level1.unity	(v1)
Level2.unity	(v1)
Level3.unity	(v2)
MainMenu.unity	(v1)
…	

← Pull

Working Files B	
Level1.unity	(v1)
Level2.unity	(v2b)
Level3.unity	(v2)
MainMenu.unity	(v1)
…	

Local Repo B	
Level1.unity	(v1)
Level2.unity	(v1)
Level3.unity	(v2)
MainMenu.unity	(v1)
…	

← Pull

Remote Repo	
Level1.unity	(v1)
Level2.unity	(v1)
Level3.unity	(v2)
MainMenu.unity	(v1)
…	

Conflict

The first dev will be able to commit and push
with no error

Working Files A

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Commit

Local Repo A

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Push

Working Files B

```
Level1.unity    (v1)
Level2.unity    (v2b)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Local Repo B

```
Level1.unity    (v1)
Level2.unity    (v1)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Remote Repo

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```
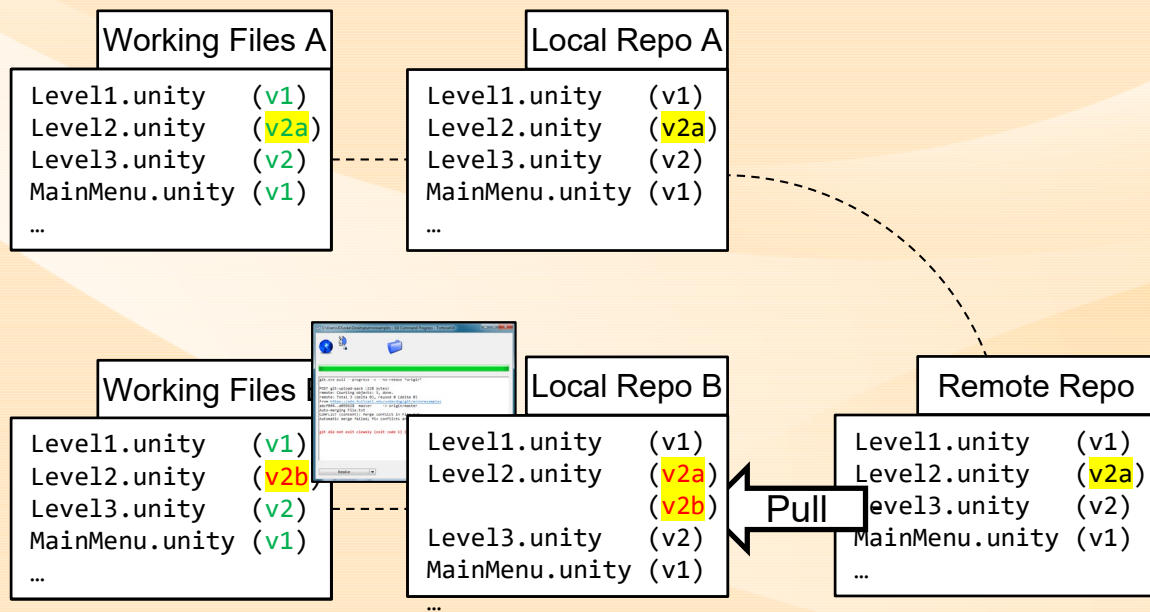
Conflict

The second dev will be able to commit but will be blocked aby the `remote contains work that you do not have` error

Working Files A

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Local Repo A

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Working Files B

```
Level1.unity    (v1)
Level2.unity    (v2b)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Commit ➡

Local Repo B

```
Level1.unity    (v…)
Level2.unity    (v…)
Level3.unity    (v…)
MainMenu.unity  (v1)
…
```

Remote Repo

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
```
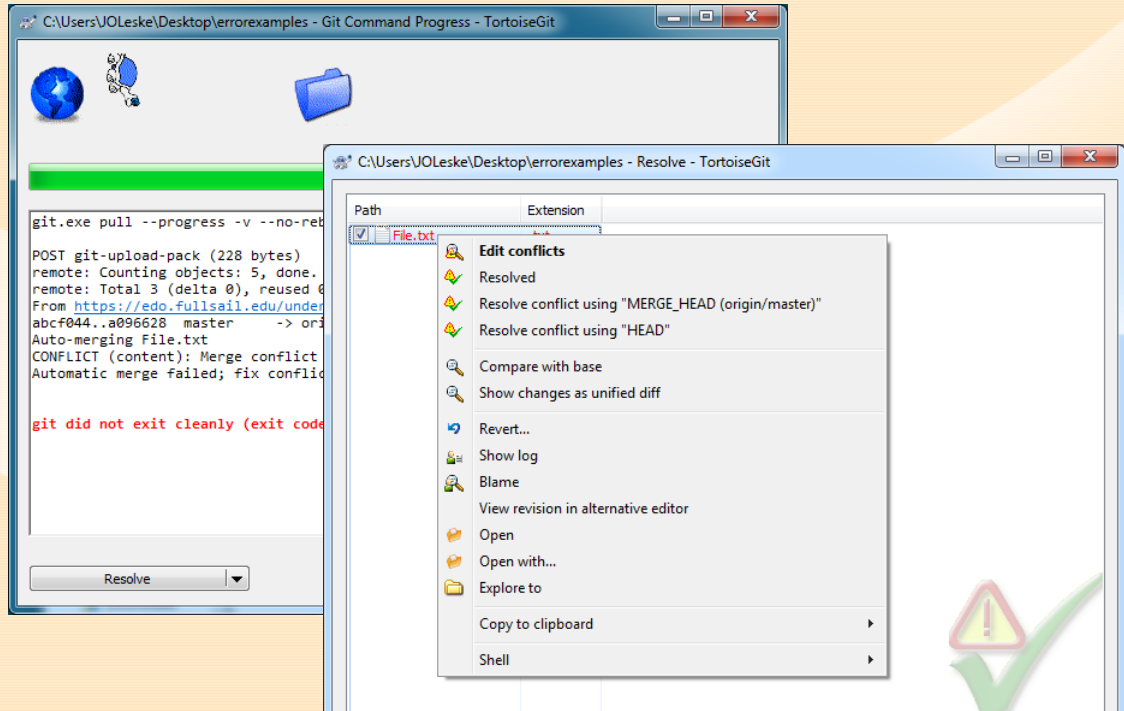
Conflict

When the second dev pulls to fix the first error they will then get the "`conflict`" error because git doesn't know what to do with the changes from DevA and DevB since the both changes the same file

Working Files A

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Local Repo A

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Working Files B

```
Level1.unity    (v1)
Level2.unity    (v2b)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Local Repo B

```
Level1.unity    (v1)
Level2.unity    (v2a)
                (v2b)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Remote Repo

```
Level1.unity    (v1)
Level2.unity    (v2a)
Level3.unity    (v2)
MainMenu.unity  (v1)
…
```

Pull

Conflict

There are 4 main options to resolve a conflicted file

Conflict: Resolution ""
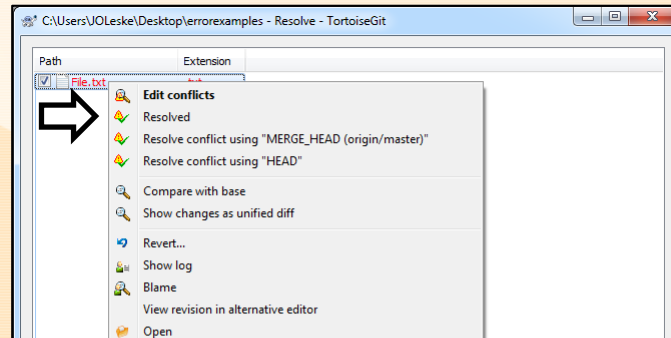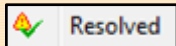
Edit Conflicts
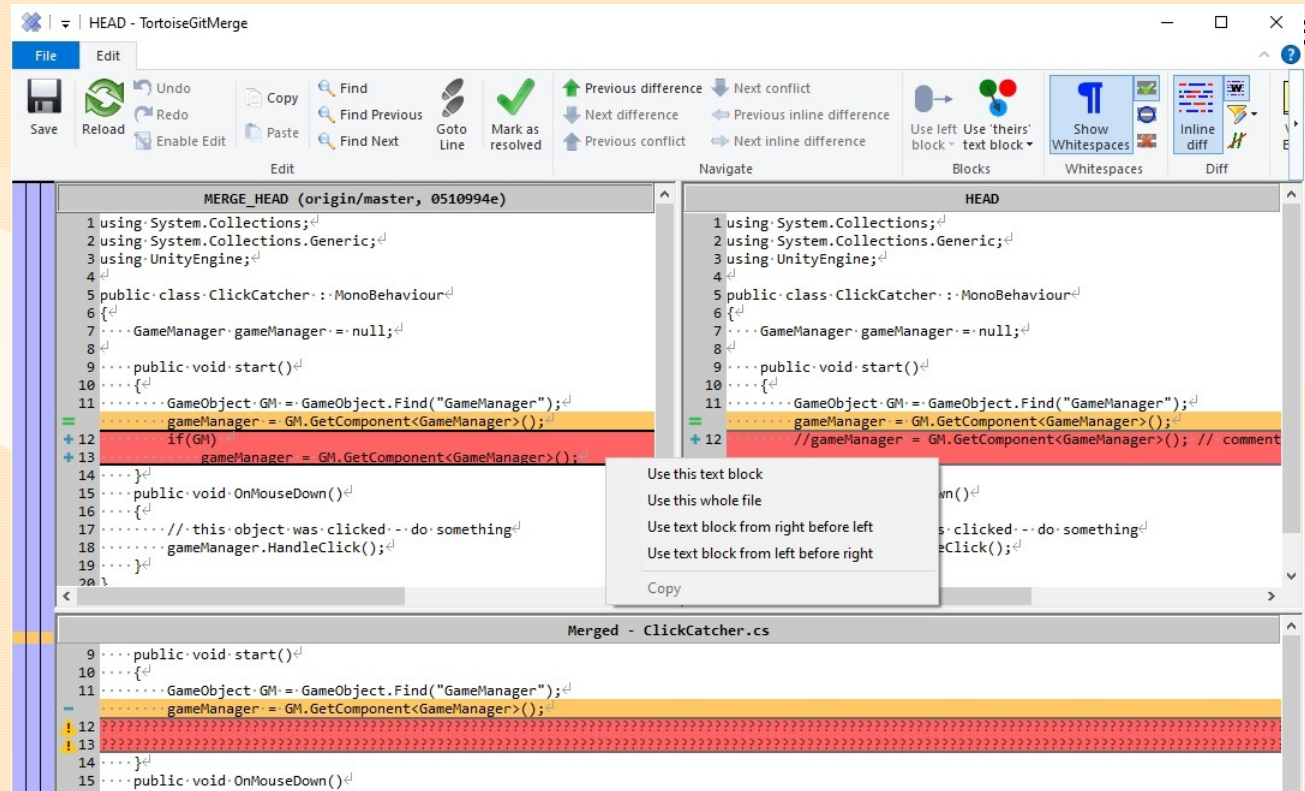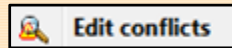
Conflict: Resolution

Resolved

- If the conflict is resolved manually without using the git interface they can simply be marked as resoled
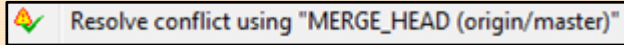
Conflict: Resolution

Resolved

Resolved

- DO NOT do this unless you have actually fixed the conflict elsewhere
- Conflicted files have extra data added to them by git to track the 2 versions of the file
  - Any time you see"<<<<<<< HEAD" or ">>>>>>> <some GUID>" the file has not been correctly resolved and is most likely broken.

```
 9      public void start()
10      {
11          GameObject GM = GameObject.Find("GameManager");
1   <<<<<<< HEAD
13          //gameManager = GM.GetComponent<GameManager>(); // commented out due to null reference errors
14   =======
15          if(GM)
16              gameManager = GM.GetComponent<GameManager>();
17   >>>>>>> 0510994e26fdd8c922503fe5f4d90d4f7df27ec6
18      }
```

Conflict: Resolution

Resolve using "MERGE HEAD"

Resolve conflict using "MERGE_HEAD (origin/master)"

- Take all of my changes and throw them away to keep all the changes from the remote repo
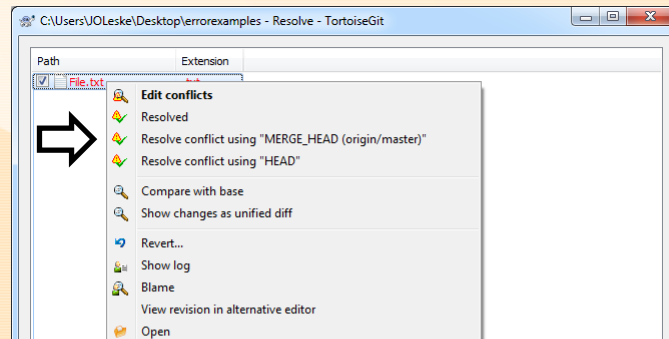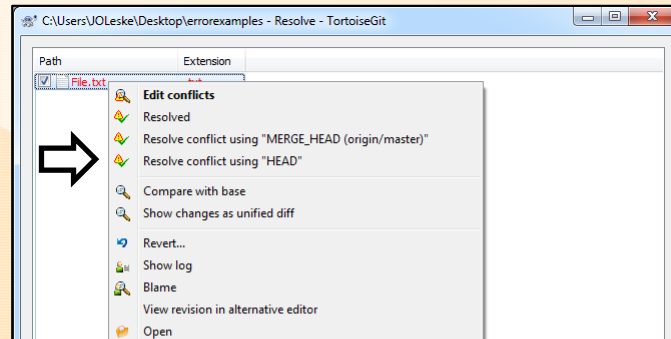
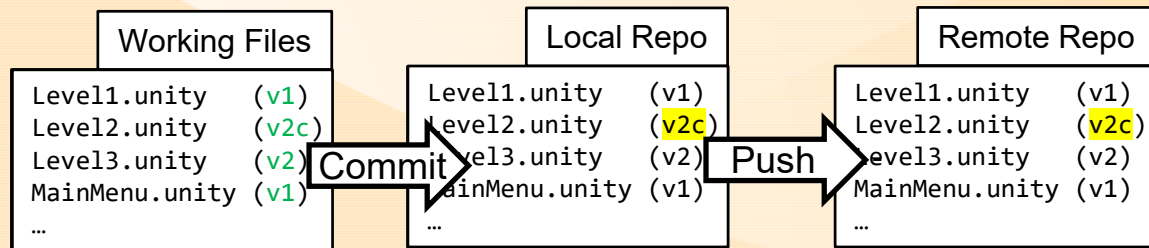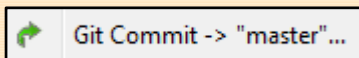# Conflict: Resolution

Resolve using "HEAD"

- Take all of the changes from the remote repo and throw them away to keep all of mine

Commit

After conflicts have been resolved you have to commit those changes

Git Commit -> "master"...

Working Files		Local Repo		Remote Repo	
Level1.unity	(v1)	Level1.unity	(v1)	Level1.unity	(v1)
Level2.unity	(v2c)	Level2.unity	(v2c)	Level2.unity	(v2c)
Level3.unity	(v2)	Level3.unity	(v2)	Level3.unity	(v2)
MainMenu.unity	(v1)	MainMenu.unity	(v1)	MainMenu.unity	(v1)
…		…		…	

Commit ⟹ Push ⟹

Git
Tips and best practices

Tips: Avoid having to merge

- Don't work on the same files at the same time
  - Binary files (non text) cannot be merged with Git
  - Don't work in the same scene at the same time
  - Don't work on the same prefabs at the same time

- Communicate and check in and check out assets that everyone contributes to
  - Scene files and Prefab files being the most common sources of conflicts

Tips: Use Sandbox Scenes

- Organize the shared scenes
  - Scene for each level

- Use sandboxes
  - Each person on the team needs their own work scene.
  - Complete as much work as you can and test in the sandbox

- Integrate into shared scenes
  - Only integrate after it has been tested to work on the sandbox scenes
  - Need to be sure only one person works on a scene at a time

Tips: Integrate Often

- Integrate to the remote server each time a task is complete

- The more time between pulls the more merge problems you can encounter

Tips: Do not bloat the server space

- Time is a resource
  - Don't push giant packages of resources if you are only using 1 thing from it
  - Huge repos take longer to push and pull

- Can cause down time when things go wrong
  - When the server is out of space no one can push or pull until the server space is freed and reset manually
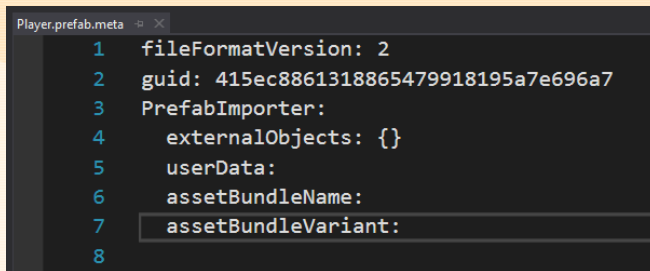
Tips: Only use 1 versioning system

- Don't use git in a folder that is synced with another application
  - One drive
  - Google drive
  - Dropbox
  - …

- Using 2 versioning systems at the same time can cause unexpected and unfixable issues

Tips: Learn from errors

- If things go wrong learn why and fix the problem in your process

- TortoiseGit does give useful error messages but you have to read the whole thing
  - Not just the red text
  - There is also the error help doc and contacting me on discord in case you get stuck

- Git does work
  - Most used version control system at present
  - Don't blame the hammer when you hit your thumb

Unity: Meta files

- How they work
  - Unity generates meta files for all files in the assets directory
  - Meta files contain GUIDs
  - Every reference to that asset/object in unity is done using that GUID as a reference
  - Deleting or regenerated a meta files makes all objects lose their references

```
Player.prefab.meta
1   fileFormatVersion: 2
2   guid: 415ec8861318865479918195a7e696a7
3   PrefabImporter:
4     externalObjects: {}
5     userData:
6     assetBundleName:
7     assetBundleVariant:
8
```

- Don't delete them
- Don't regenerate them
- Once one is versioned keep that version

Additional help if needed

- Common Git Errors and Working with Unity and Git links under the References of the repository

- Contact Staff
  - More than happy to help you resolve a git issues

- Lot and lots of additional online tutorials for git. Most working with command line interface.

# Activities

<Activity> Git crash course

- Create project file of some kind
  - Sandbox scenes in the scenes folder for the unity projects
  - Each team member create their own unique file

- Coding Standards and Best Practices
  - Each team member add your name to the file
  - (text document)

- Design document
  - Each team member add your name to the file
  - (Binary file)

- Sync versions to get everyone's changes
  - You will get conflicts and errors
  - That's the point of the activity

Assignment

Assignment

By Lecture 2

- Pre pro (drafts ready for review)
  - Trello boards filled out
    - Brainstorming features
  - Design Document worked on
  - Risk assessments worked on
  - Coding Standards and Best Practices agreed upon
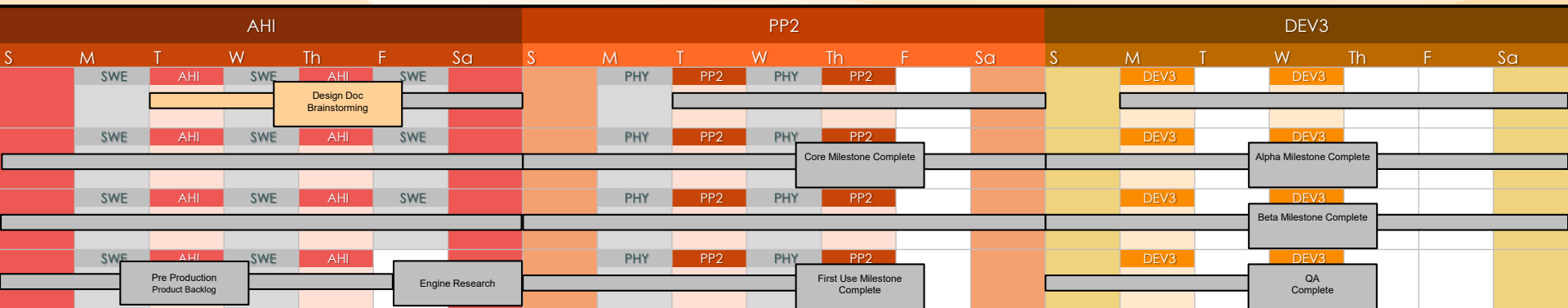
Additional info

Additional Info

Design document
- [The Anatomy of a Design Document: gamasutra.com](#)
- [GDC: 30 Things I Hate About Your Game Pitch](#)

Git
- [Programming Foundations: Version Control with Git: linkedin.com/learning](#)

Code standards
- [Standards and conventions used by Epic Games in the Unreal Engine 4 codebase](#)

Risk Assessments
- [Taming the Chaos: Lessons in Project Management](#)

Contact Info

- John OLeske
  - Discord: JohnOLeskeFS#4268
  - JOLeske@fullsail.edu
  - Trello: johnoleske
  - Work Phone: 407.551.2024 x 8926

- Office hours
  - Mon 1pm-5pm Thursday 1pm-5pm
  - By request available