
GDBS/SVBS Midterm Project Overview

Goal

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

Who are we

Program Director:
Jason Hinders
jhinders@fullsail.com

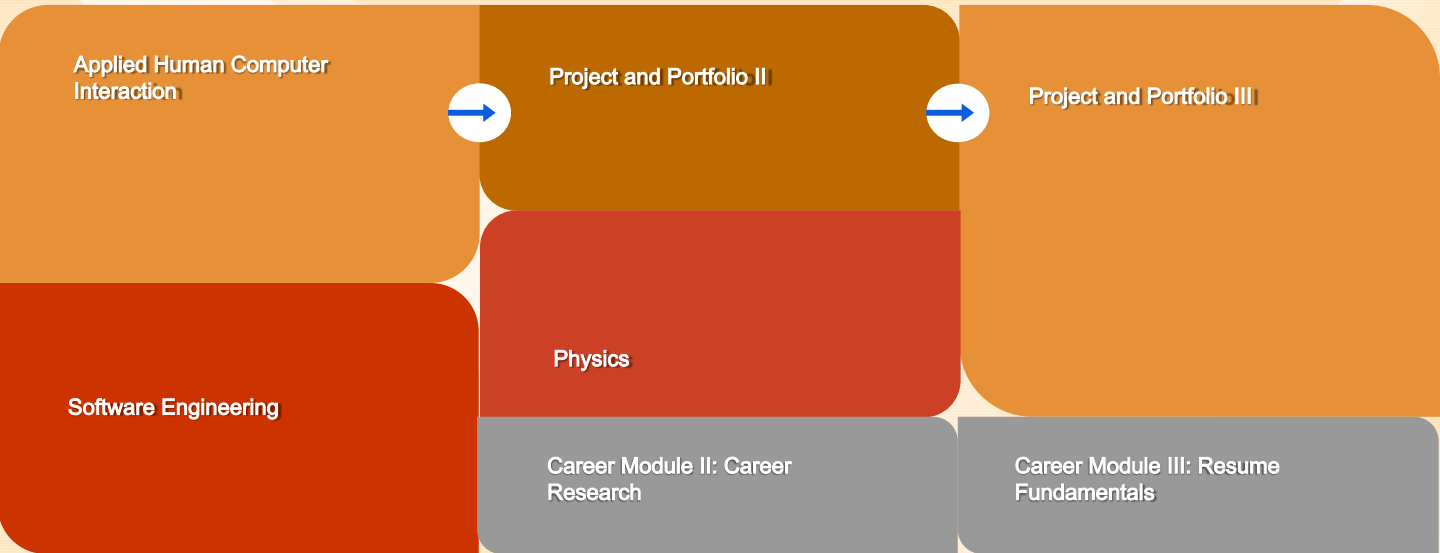
Department Chair:
Rebecca Leis
rleis@fullsail.com

CD of AHI:
Steve VanZandt
svanzandt@fullsail.com

CD of PP2:
Rod Moyer
rmoyer@fullsail.com

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

Full Midterm Project Process



Full Midterm Project Process

Applied Human Computer Interaction

- Pre Production
 - Design Document
 - Product Backlog
- AHI Topics
 - Nielsen's heuristics
 - Usability
 - UX

Project and Portfolio III

- Core Functionality
 - Critical game systems
 - Interface and UI creation
- First Use/Playable
 - Playable complete
 - Experience
 - Fun factor

Project and Portfolio III

- Alpha
 - Full Functionality
 - Example Content
- Beta
 - Content complete
 - Balancing
- Finalizing
 - QA process
 - Presentation

Project Expectations



The game we are making

Design a game with your capabilities in mind

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

Scope

- 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

Game Expectations

Platform Support

- Must support a secondary platform
 - (Keep file sizes low)
 - WebGL version exported to a webpage or published to a web portal
 - or
 - Playable on tablet or phone
- Keep platforms in mind when making design and production choices
- Keep file size low
 - < 1GB
 - Under 512 MB preferred

Game Expectations

Expo

- Games will be presented at the FPS Expo
- First Thursday of the month after PP3
 - 11am-1pm

Developer Expectations

Expectations: Problem Solving

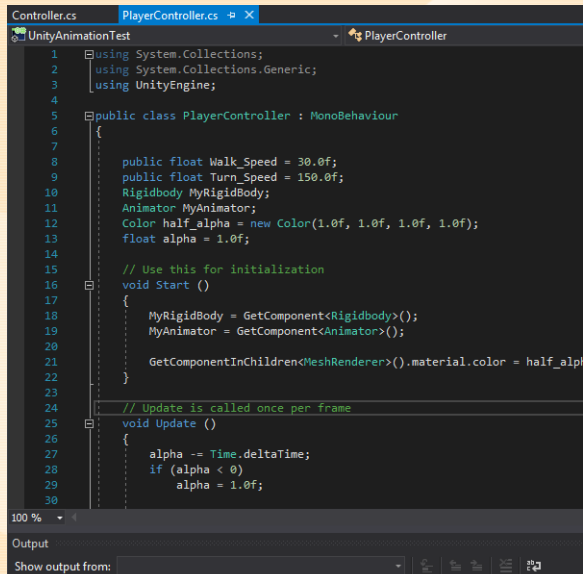
Put less importance on knowing things ahead of time.

- 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

Expectations: Be a developer

We will be doing a lot of different jobs

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

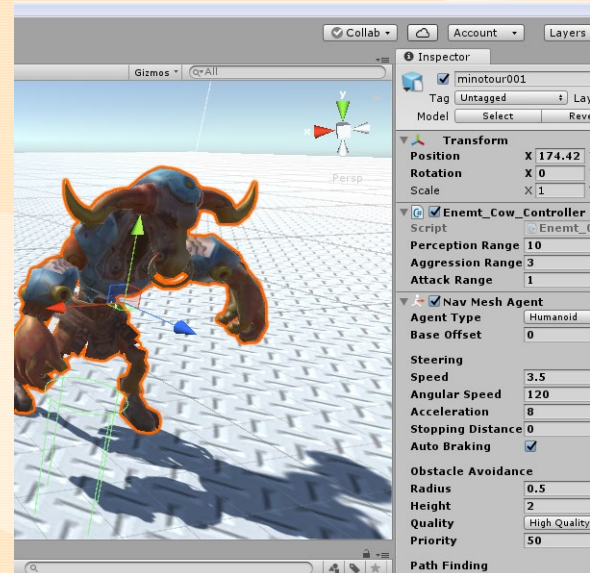
A screenshot of a code editor showing a C# script named 'PlayerController.cs'. The script is for a 'MonoBehaviour' and includes using statements for 'System.Collections', 'System.Collections.Generic', and 'UnityEngine'. It defines a class 'PlayerController' with two public float variables: 'Walk_Speed' (30.0f) and 'Turn_Speed' (150.0f). It also has private variables for 'MyRigidbody', 'MyAnimator', and 'alpha' (initialized to 1.0f). The 'Start' method initializes 'MyRigidbody' and 'MyAnimator' using 'GetComponent' and sets the color of the first child 'MeshRenderer' to 'half_alpha'. The 'Update' method, which is called once per frame, decrements 'alpha' by 'Time.deltaTime' and resets it to 1.0f if it reaches 0.

```
Controller.cs PlayerController.cs
UnityAnimationTest PlayerController
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
11     Rigidbody MyRigidbody;
12     Animator MyAnimator;
13     Color half_alpha = new Color(1.0f, 1.0f, 1.0f, 1.0f);
14     float alpha = 1.0f;
15
16     // Use this for initialization
17     void Start ()
18     {
19         MyRigidbody = GetComponent<Rigidbody>();
20         MyAnimator = GetComponent<Animator>();
21
22         GetComponentInChildren<MeshRenderer>().material.color = half_alpha;
23
24     }
25
26     // Update is called once per frame
27     void Update ()
28     {
29         alpha -= Time.deltaTime;
30         if (alpha < 0)
31             alpha = 1.0f;
32     }
33 }
```

Expectations: Be a developer

We will be doing a lot of different jobs


- Designer
 - Defining mechanics
 - Level designs
 - Balancing



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 they collide with the top of them?

☐ Do spikes not damage the player if they 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?

Add an item...

 **Add Comment**

Expectations: Be a developer

We will be doing a lot of different jobs

- Artist
 - Adding assets to product
 - Creating simple assets



Expectations: Be a developer

We will be doing a lot of different jobs

- Quality Assurance Tester
 - Testing and reporting bug

The screenshot shows a bug report form for a game. The title is "Player can not walk through door on second room of level 2". The form includes fields for "Found By" (John OLeske), "Category" (Design/Placement), "Build Found" (Round 1), and "Description". The description details the steps to reproduce the bug: selecting a new game, continuing to level two, and attempting to walk into a door. It also notes what was seen (player collision with the door) and what was expected (player entry). The form has a "Save" button and a "Share and more..." link. On the right, there are sections for "Add" (Members, Labels, Checklist, Due Date, Attachment), "Power-Ups" (Custom Fields), "Actions" (Move, Copy, Subscribe, Archive), and "Activity".

Player can not walk through door on second room of level 2
in list [Open](#)
☐ Recurring Add #tags Spent / Estimate

Labels: **C - Minor** + Found By: John OLeske Category: Design/Placement

Build Found: Round 1

Description [Edit](#)
Steps to reproduce:

- Select new game
- Continue to level two (Office space)
- Attempt to walk into the door in the second room

What was seen:

- Player collides with the door and wall but the door does not animate or allow the player to walk through.

What Was Expected:

- Player can enter the door or it is clear that the door can not be entered

[Add Comment](#)
Write a comment...
Save

[Share and more...](#)

Add
[Members](#)
[Labels](#)
[Checklist](#)
[Due Date](#)
[Attachment](#)

Power-Ups
[Custom Fields](#)

Actions
[00:00:00](#)
[Move](#)
[Copy](#)
[Subscribe](#)
[Archive](#)

Activity [Hide Details](#)

Expectations: Team work

Work with each other

- Have a set schedule
 - Not just during Lecture/Lab
 - Set a schedule for everyone to work TOGETHER.
 - You will always get more done as a group

Expectations: Communication

Communication will be a challenge

- Mandatory schedule
 - CDs will see you 1 or 2 times a week
 - That isn't enough to keep communication open
- Keep in contact with us
 - If something breaks, tell us
 - If there are issues with finding art, tell us
 - If something awesome changes in the project, tell us

Our Availability

FS3B 304

Monday-Thursday

9am-5pm

Either Rod Moyer or John Oleske is running
midterm project

Teams are always welcome and encouraged
to come work.

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.**"

...

"Plagiarism Defined (as in Webster's Dictionary):

- 1 to steal and pass off the ideas or words of another as one's own
- 2 use a created production without crediting the source
- 3 to commit literary theft
- 4 **present, as new and original, an idea or product derived from an existing source"**

● 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
 - Prefabs
 - Scenes
 - If it can be made in unity you are expected to make it
- 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
 - 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 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):

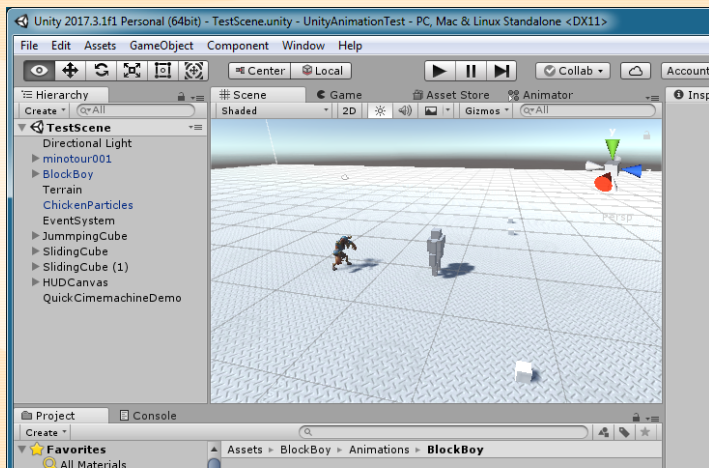
- In group work, allowing a team member to include his/her name to "tag along" on the work of other team members despite having not significantly contributed to that work.

Tools

Unity3D

Unity3d.com

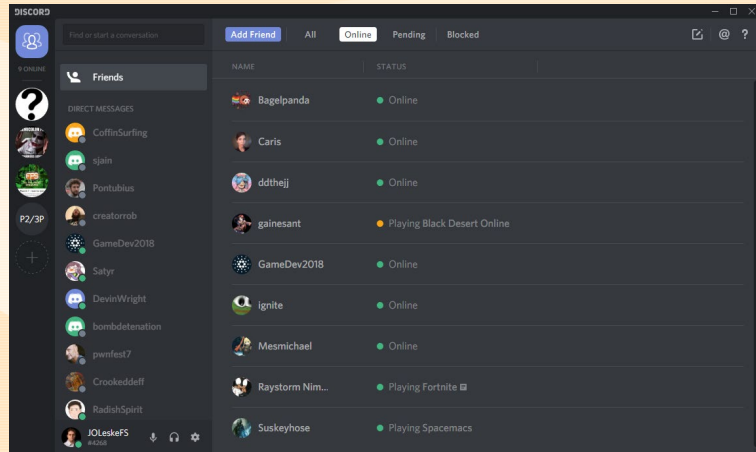
- Our dev environment
- Scripting in c#
- Ensure each team member is using the same version



Discord

discordapp.com

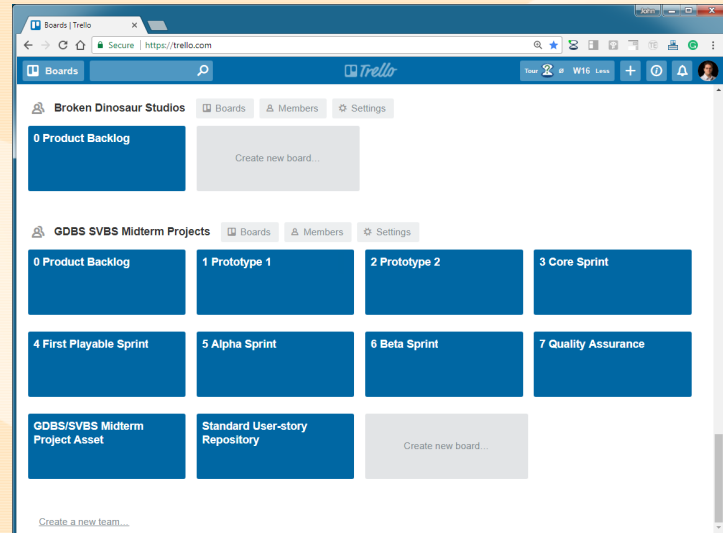
- Keep communication open while working remotely
- Share files that are not part of the project



Trello

trello.com

- This will be both our design space and our task management system



Chrome and plus for trello plugin

plusfortrello.com

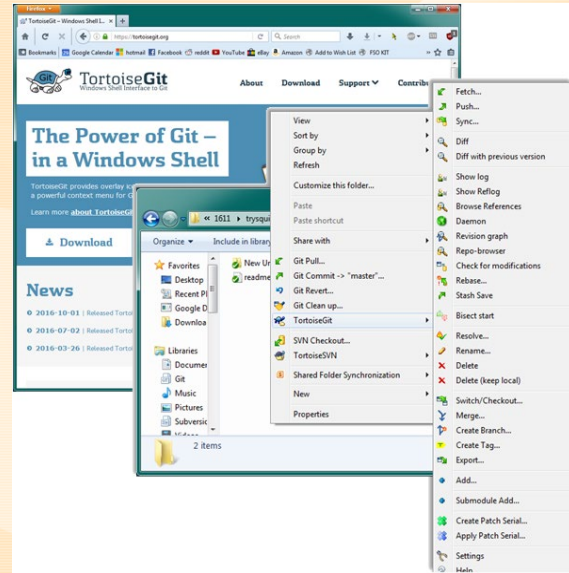
- Will need chrome for plus fortrello plug in



TortoiseGit

tortoisegit.org

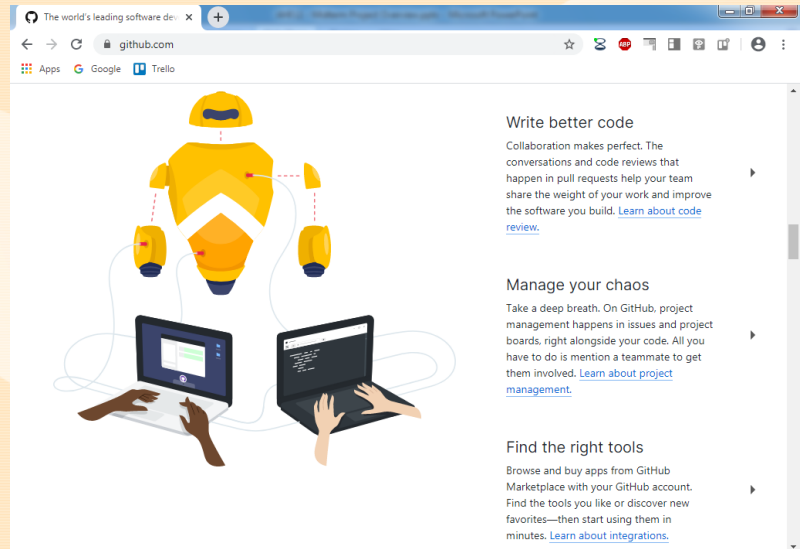
- We will be using git for our versioning system



Git Hub

github.com

- Git server will be provided through GitHub



Install or confirm

- Unity
(unity3d.com)
Everyone on same version
- TortoiseGit
(tortoisegit.org)
- Git framework
(git-scm.com)
- Discord
(discordapp.com)
- Trello
(trello.com)
Accounts created
- 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 your team

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

Collect the info

- Team
 - Team name
 - Discord channel
 - Invite me to the server (JohnOLeskeFS#4268)
- For each team member
 - Full name
 - Trello username
 - (The one in the parenthesis next to your full name on the webpage)
 - Github username
 - Discord username
 - Email

<Activity> Discuss game ideas

Think of a game

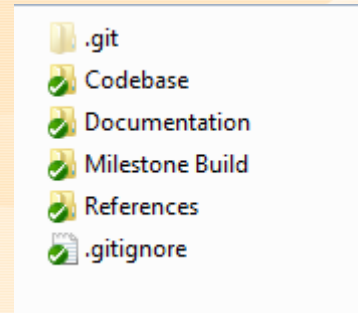
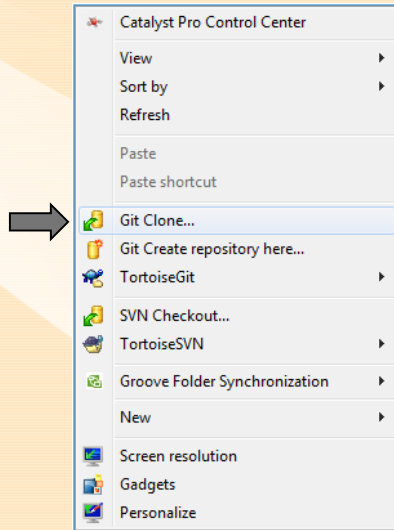
- Discuss as a team what kind of game you would like to develop
- I will be setting up servers and documents for all of the teams

Activities

<Activity> Clone the project folder

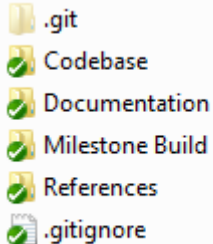
Clone the repo

- 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
- **.gitignore**
 - The file that defines what should not be pushed to the repo



Pre-Production

The purpose of Pre production

- 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
- As hard as preproduction is, it's even harder to do while you are already in production
 - And more expensive in the long run

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

Trello

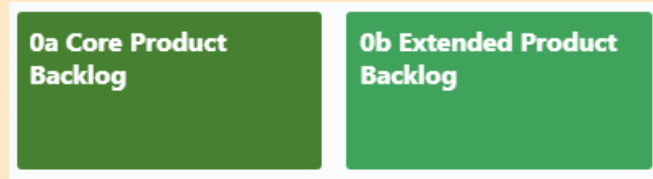
Our task management
platform

Trello: Our task management platform

- Bottom up design
- Brainstorm features
 - Create new card for each feature to be added to the game
 - Player action
 - Character
 - Enemy type
 - Item
 - Weapon
 - Power up
 - Game mode
 - ...
 - Document dependencies

Trello: Our task management platform

Our team's Trello boards

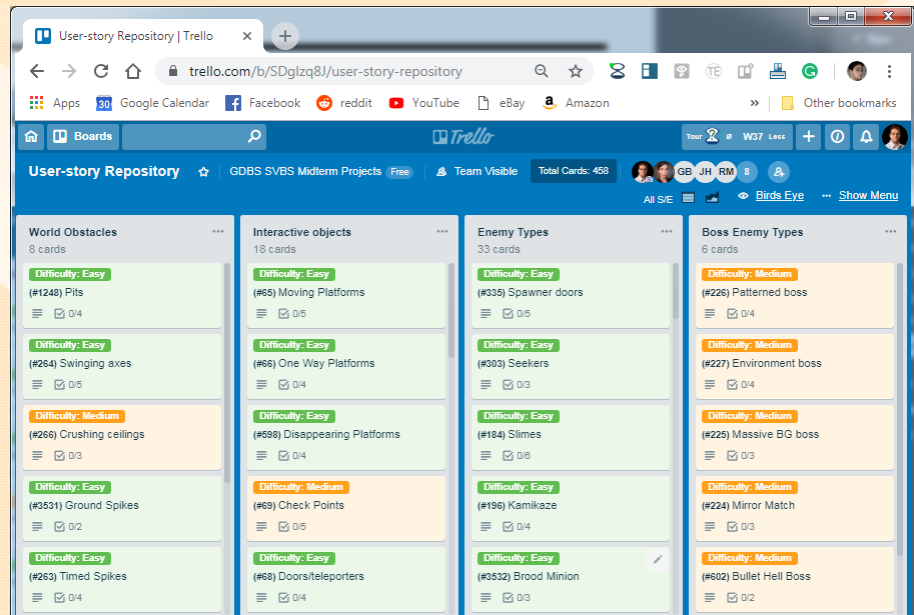


- “0a Core Product Backlog”
 - All items and features required to create the product’s vertical slice
 - M.V.P. Minimum viable product
- “0b Extended Product Backlog”
 - All the wish list, would be cool to have, stretch goals... of the product

Trello

By lecture 2 our trello boards should look something like this

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



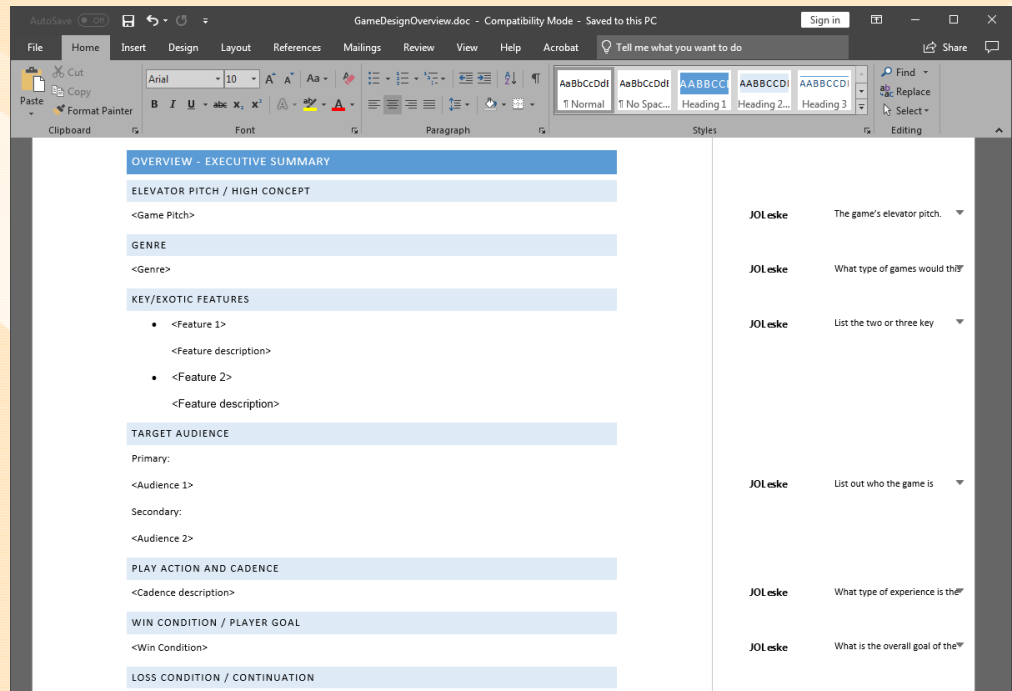
The Design Document

Design Document

- Top down design
- Agree on the core design
 - Overall goals
 - Fun Factor
 - Selling points of the game
 - Overarching systems and mechanics

Design Document

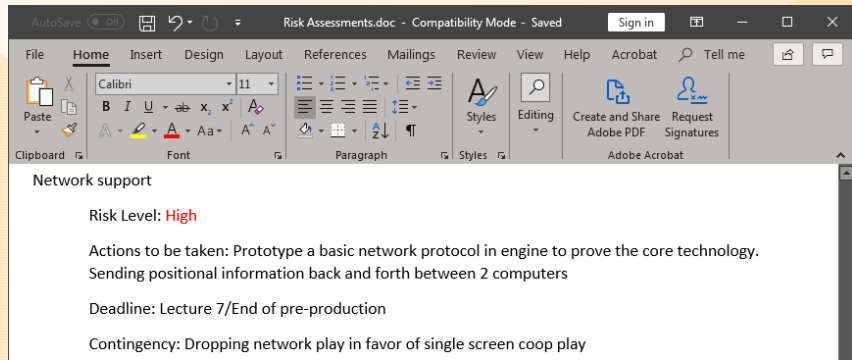
- Template document has been provided for you
 - In your repository's documentation folder



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

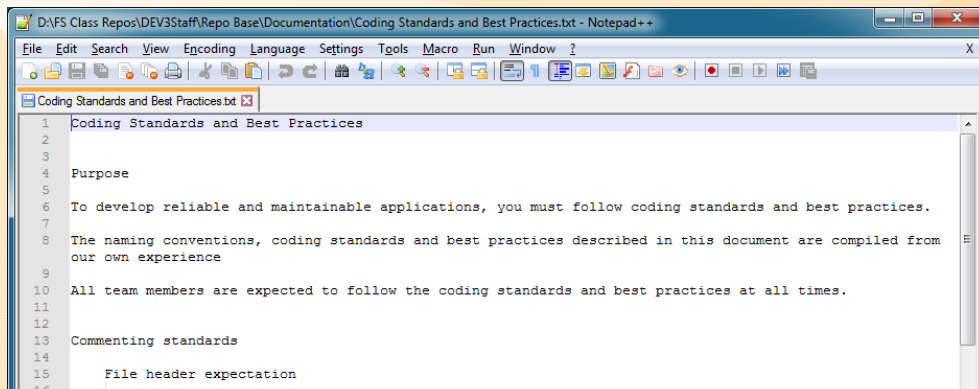


Coding Standards and Best Practices

Coding Standards and Best Practices

The team agreed to a standards

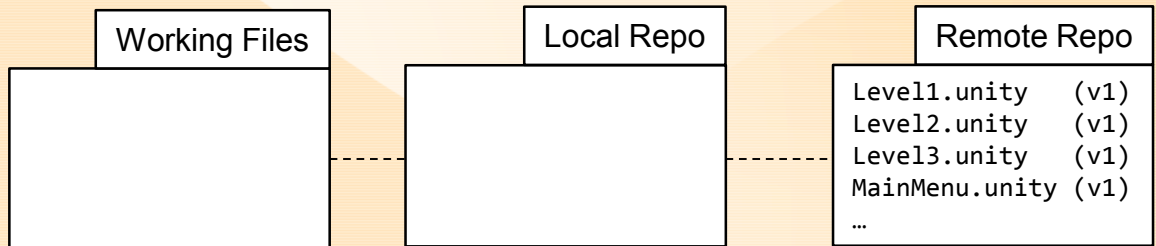
- Commenting
 - Naming conventions
 - File Formatting
 - Indentation and Spacing
 - General Programming practices
- Template document has been provided for you.
 - In your repository's documentation folder



Version Control

Git Basics

- Understand the system behind the interface
 - Three main sections to pay attention to
 - Working files
 - Local Repository
 - Remote repository

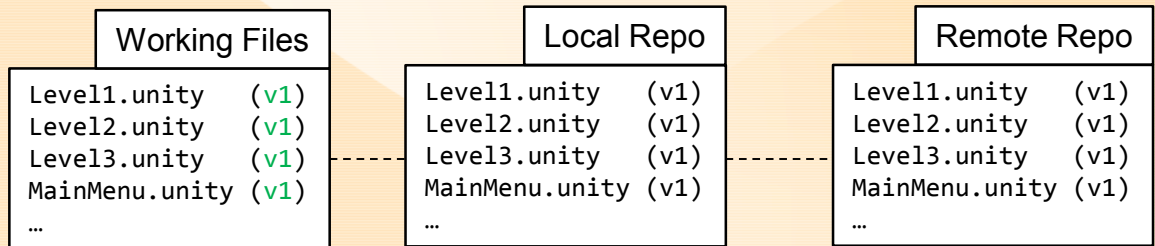


Clone

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

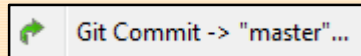


- Get a bring remote repository into the local repository
- Populate working files from local repository

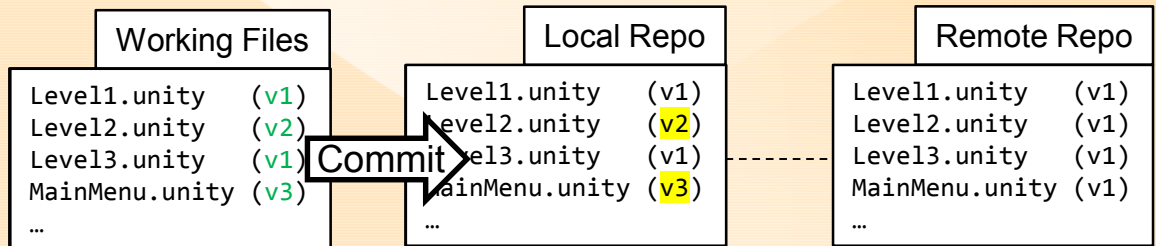


Commit

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

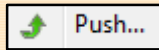


- Commit saves the changes to the local repository
- Once committed there is a timestamp of the files that can always be returned to

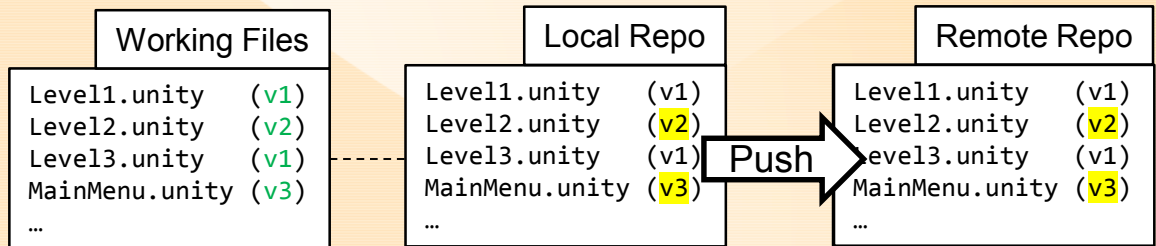


Push

Push integrate changes onto the remote repository



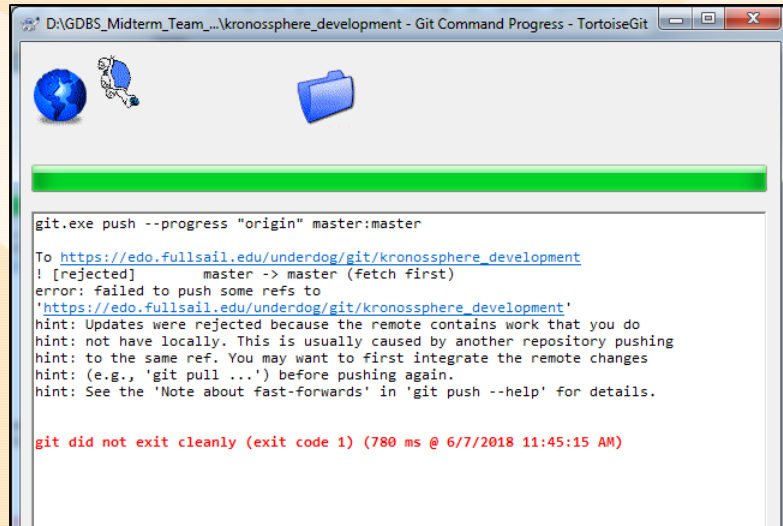
- Non committed work does not get pushed
- If a file hasn't been added it doesn't get committed



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)

A screenshot of a TortoiseGit window titled "D:\GDBS_Midterm_Team_... \kronossphere_development - Git Command Progress - TortoiseGit". The window shows a green progress bar at the top. Below it, the command "git.exe push --progress "origin" master:master" is displayed. The output shows a rejection of the push to the remote repository "https://edo.fullsail.edu/underdog/git/kronossphere_development". The error message states: "error: failed to push some refs to 'https://edo.fullsail.edu/underdog/git/kronossphere_development'". It includes several hints: "hint: Updates were rejected because the remote contains work that you do not have locally. This is usually caused by another repository pushing to the same ref. You may want to first integrate the remote changes (e.g., 'git pull ...') before pushing again." and "hint: See the 'Note about fast-forwards' in 'git push --help' for details." At the bottom, a red line indicates "git did not exit cleanly (exit code 1) (780 ms @ 6/7/2018 11:45:15 AM)".

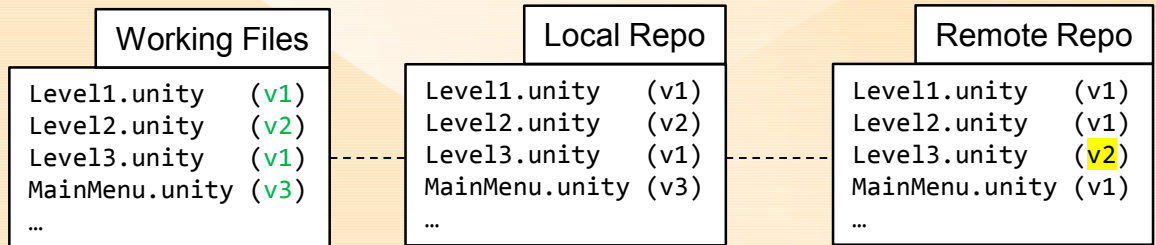
```
git.exe push --progress "origin" master:master

To https://edo.fullsail.edu/underdog/git/kronossphere_development
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to
'https://edo.fullsail.edu/underdog/git/kronossphere_development'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

git did not exit cleanly (exit code 1) (780 ms @ 6/7/2018 11:45:15 AM)
```

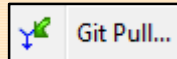
Push

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

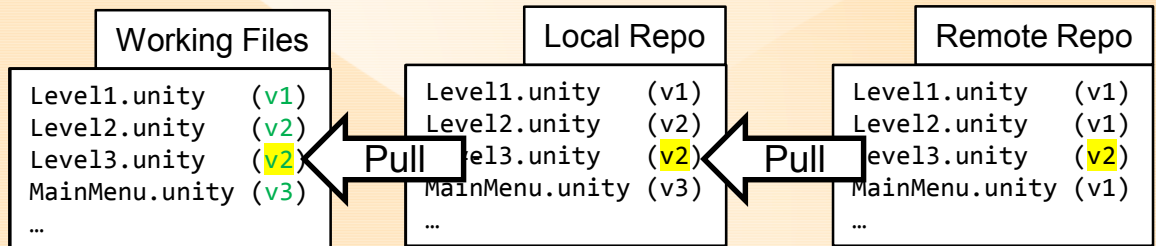


Pull

Pull changes from the remote and integrate them onto your build

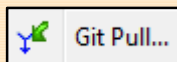


- Changes get integrated into the local repo
- If integration is good the working files change to match

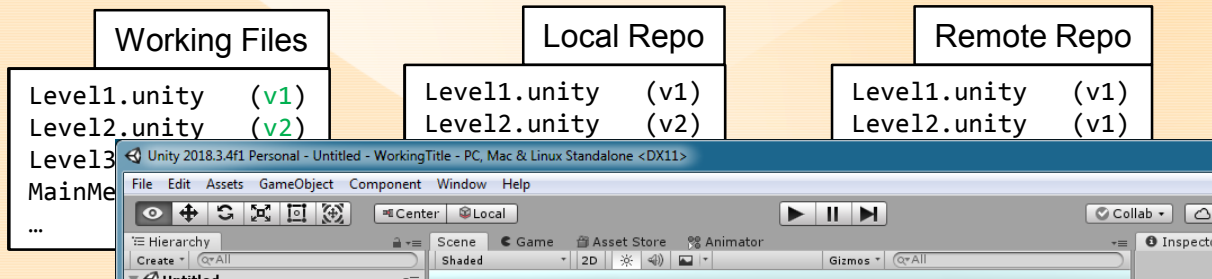


Pull

Do not pull with unity open

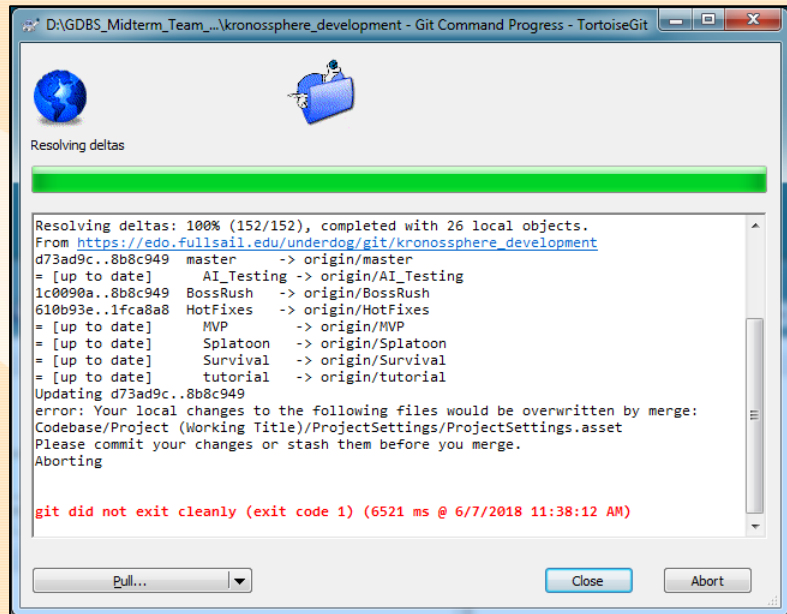


- Unity is always updating and recompiling based on changes in the files
- If you pull with unity open unity and there is an error unity will attempt to recompile with the error and break



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.”




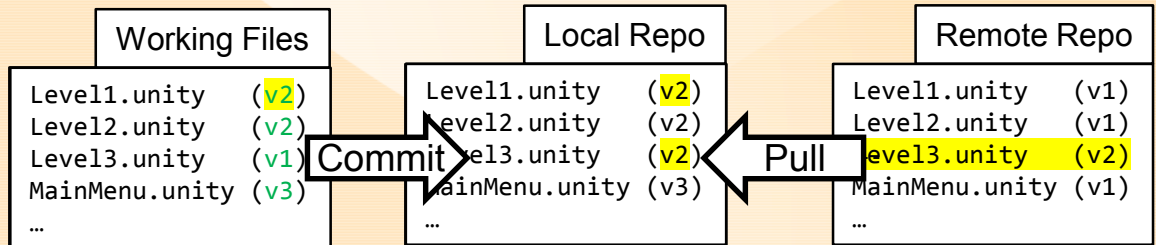
Uncommitted work error

Resolution

- Once you commit your changes you can pull and get the remote changes

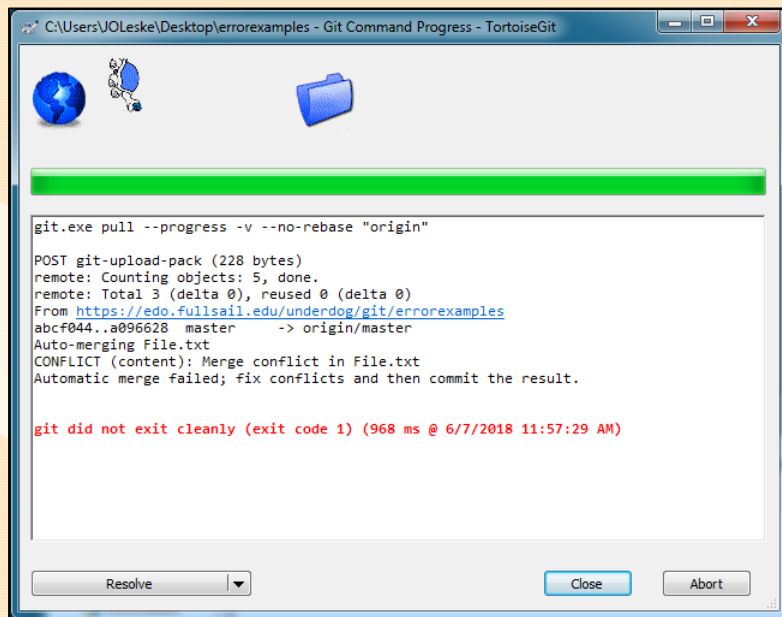
 Git Commit -> "master" ...

 Git Pull...



Conflict

“Automatic merge failed: fix conflicts and then commit the results”



A screenshot of a TortoiseGit command progress window. The window title is "C:\Users\JOLeske\Desktop\errorexamples - Git Command Progress - TortoiseGit". It features a green progress bar at the top. Below the progress bar, the command "git.exe pull --progress -v --no-rebase 'origin'" is shown. The output text includes: "POST git-upload-pack (228 bytes)", "remote: Counting objects: 5, done.", "remote: Total 3 (delta 0), reused 0 (delta 0)", "From <https://edo.fullsail.edu/underdog/git/errorexamples>", "abcf044..a096628 master -> origin/master", "Auto-merging File.txt", "CONFLICT (content): Merge conflict in File.txt", and "Automatic merge failed; fix conflicts and then commit the result.". At the bottom, a red error message states "git did not exit cleanly (exit code 1) (968 ms @ 6/7/2018 11:57:29 AM)". The window has "Resolve", "Close", and "Abort" buttons at the bottom.

```
git.exe pull --progress -v --no-rebase "origin"

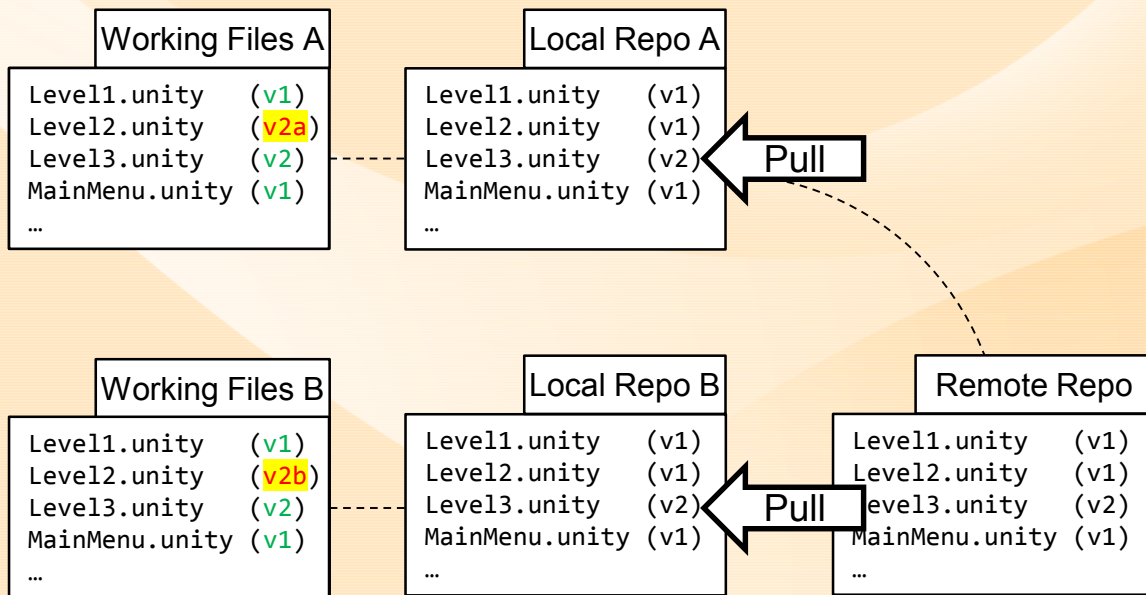
POST git-upload-pack (228 bytes)
remote: Counting objects: 5, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
From https://edo.fullsail.edu/underdog/git/errorexamples
abcf044..a096628 master -> origin/master
Auto-merging File.txt
CONFLICT (content): Merge conflict in File.txt
Automatic merge failed; fix conflicts and then commit the result.

git did not exit cleanly (exit code 1) (968 ms @ 6/7/2018 11:57:29 AM)
```

Conflict

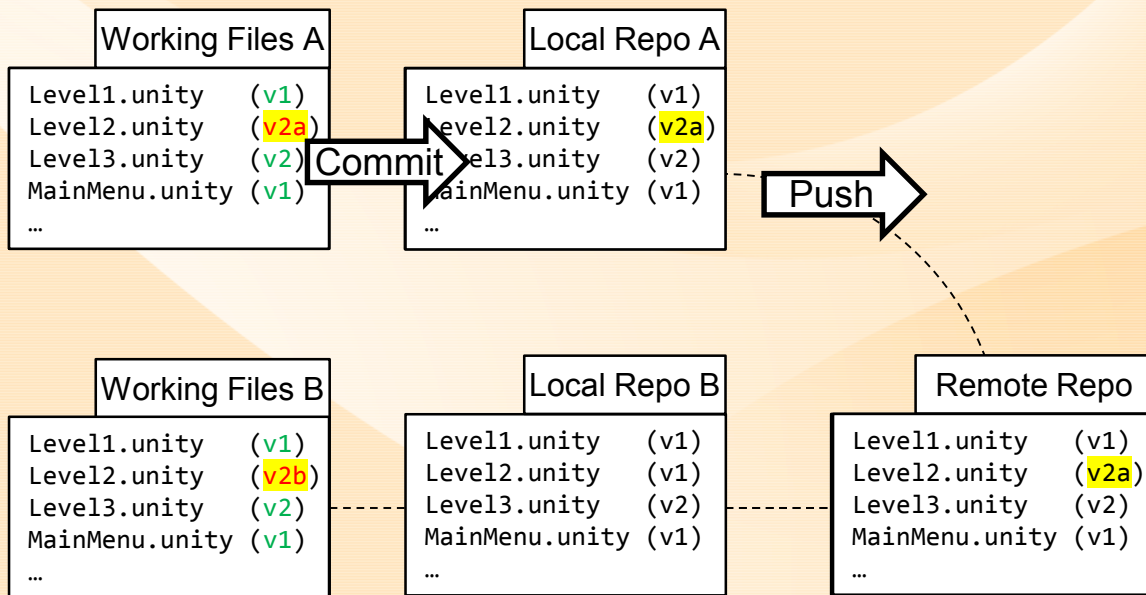
Conflicts are created when 2 pull and then modify the same file

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



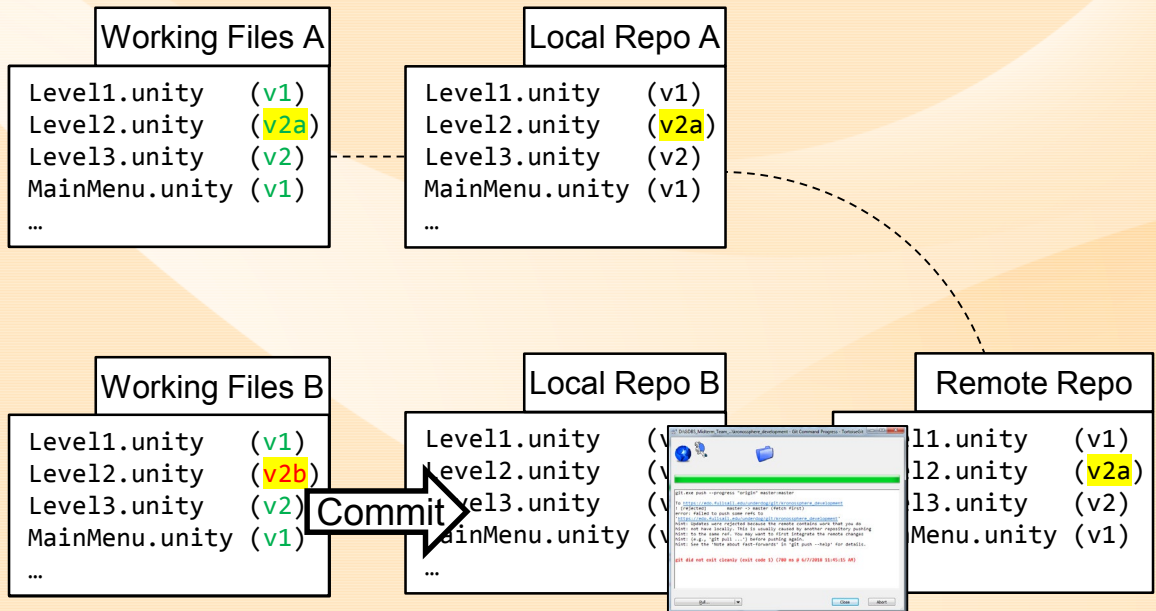
Conflict

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



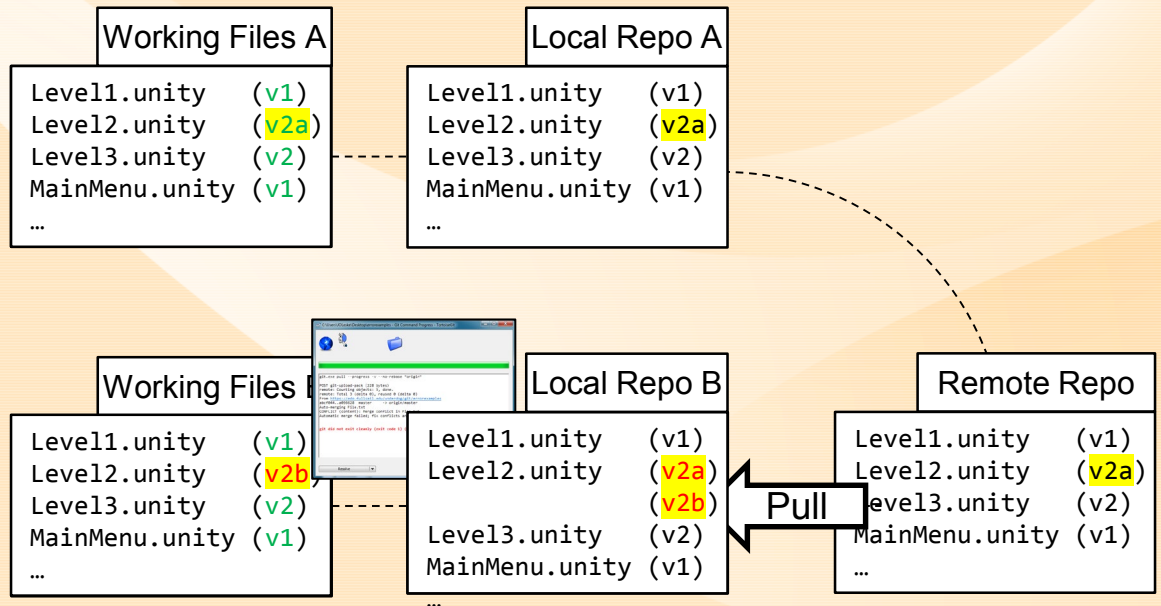
Conflict

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



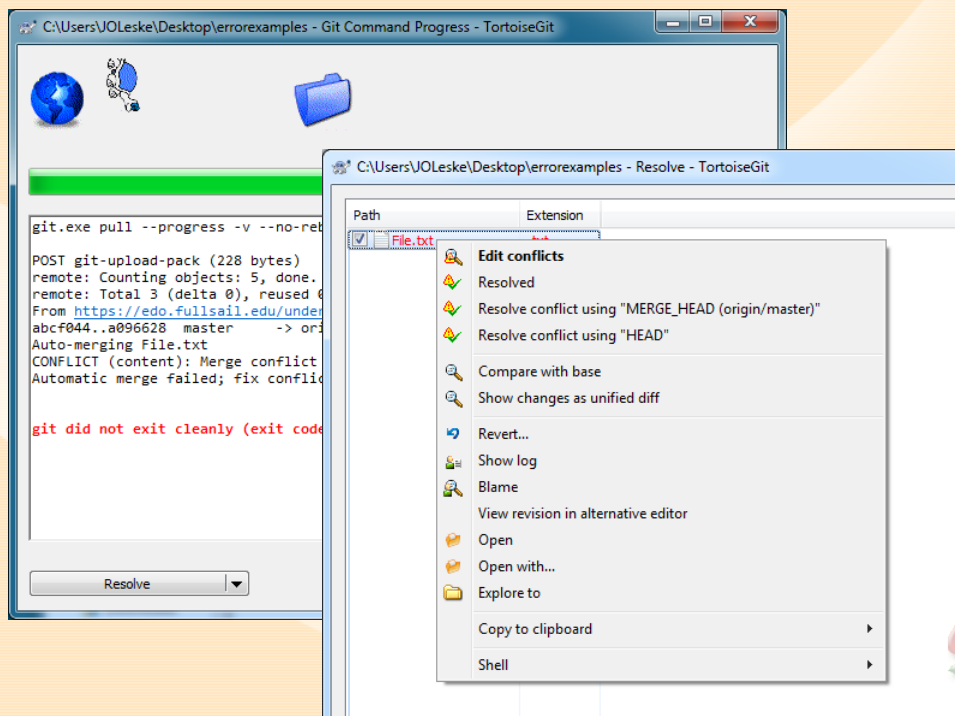
Conflict

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



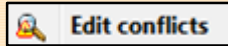
Conflict

There are 4 main options to resolve a conflicted file

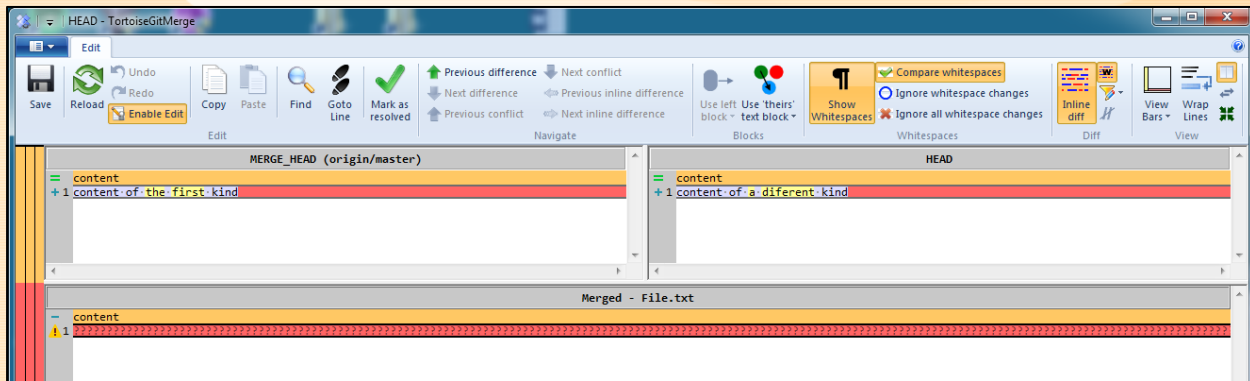


Conflict: Resolution

Edit Conflicts

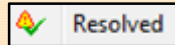


- A merge tool can be used to pick and choose between lines and choose what in each should be saved
- Only works on text based files
 - Code
 - Text based assets

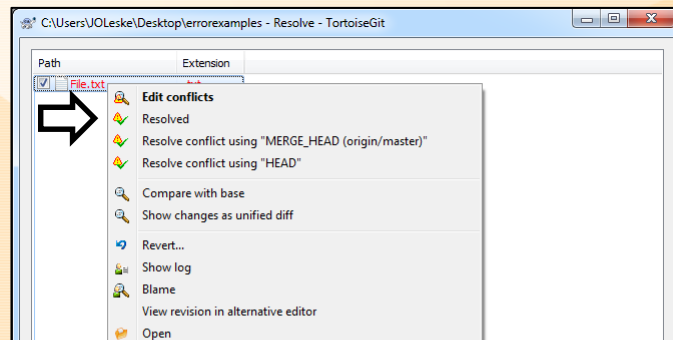


Conflict: Resolution

Resolved

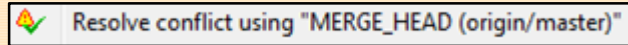


- If the conflict is resolved manually without using the git interface they can simply be marked as resolved
- DO NOT do this unless you have actually fixed the conflict elsewhere

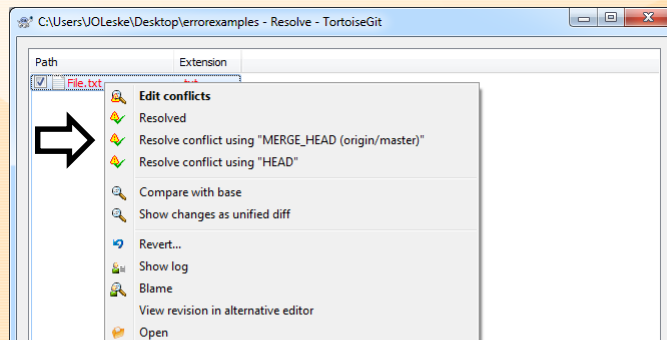


Conflict: Resolution

Resolve using “MERGE HEAD”



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

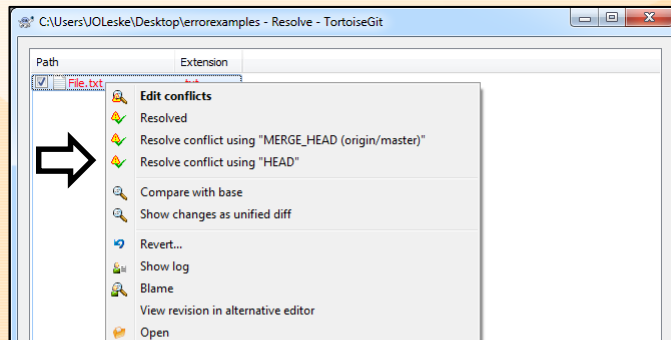


Conflict: Resolution

Resolve using “HEAD”

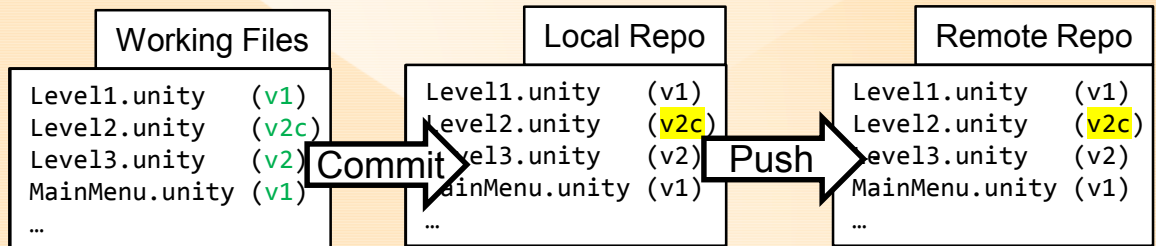
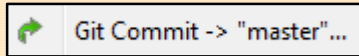


- Take all of the changes from the remote repo and throw them away to keep all of mine



Commit

After conflicts have been resolved you have to commit those changes



Git 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
 - The server space is shared with multiple projects across multiple degrees and file space is limited
 - When the server is out of space no one can push or pull until the server space is freed and reset manually

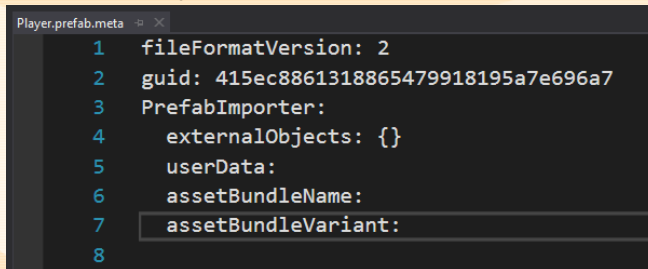
Tips: Learn from errors

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
 - Don't blame the hammer when you hit your thumb

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 through that GUID
 - Deleting or regenerated a meta files makes all objects lose their references, causing functionality to disappear/break



```
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

Activities

<Activity> Git crash course

Let's use git on a team

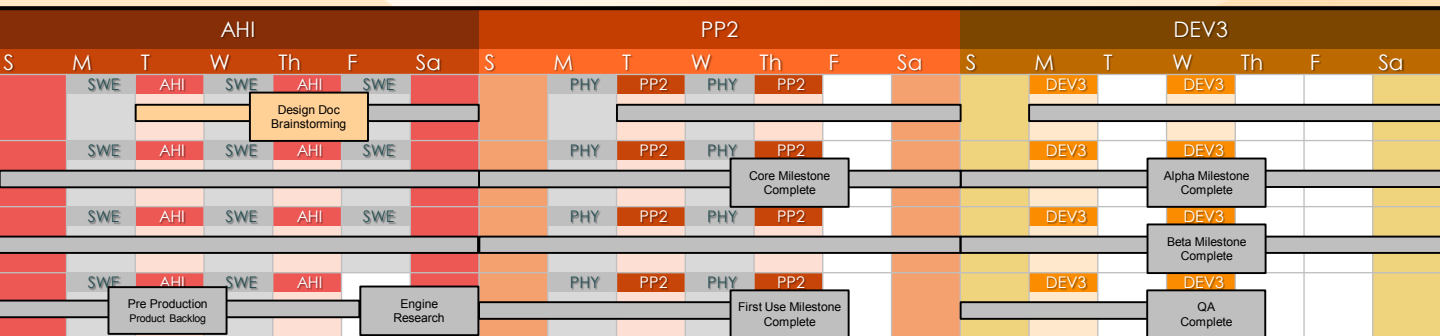
- Create sandbox scenes in the scenes folder
 - Each team member create their own
- Coding Standards and Best Practices
 - Each team member add your name to the file
 - (text document)
- Design document
 - Each team member add your name to the file
 - (Binary file)
- Sync versions to get everyone's changes
- You will get conflicts and errors
 - That's the point of the activity

Assignment

Plan

By Lecture 2

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



Contact Info

- John OLeske
 - JOLeske@fullsail.com
 - Discord: JohnOLeskeF\$4268
 - Skype: joleske
 - Trello: johnoleske
 - Work Phone: 407.551.2024 x 8926
 - Google: joleskefs@gmail.com
- Office hours
 - Mon 1-5 Friday 1-5
 - By request available