Conventions

Intro:

The following are the conventions that will be used throughout our project. It is advised that you follow these conventions as strictly as possible. This is not only to help your own organization, but sense we will be sharing a repo of which the total structure will not be known by any one person, it is to help give an understanding to all users of where things are located and the current progress of the game.

Files:

Repo:

Sense the repo will be shared by many people, files and assets in the repo should be ones that will only be included in the final build of the game. This means final assets, temp assets, and place holders are all ok to place in the repo, however assets such as concept art, sketches, and so on should not be. If you still wish to share these sorts of files it should be done so through an Asana comment on the appropriate task.

Location:

For this project we will be organizing assets by **role**. This means all assets that share something in common should be placed in the same folder. To give an example the player object may have the following files:

PlayerTexture.png
PlayerMod.obj
PlayerVoice1.wav

All these files should be located in a single player folder sense they are all used on the player. Assets should be sorted in this way by what they share in common. This will help other users easily find the assets they need.

For folder paths you should use as many sub-directories that are needed in order to give an accurate description of what your asset is. For example if you have the file PlayerTexture.png then here is how the following folder paths would rate

HORRIBLE
/Content/PlayerTexture.png

BAD
/Content/Gameplay/PlayerTexture.png

OK
/Content/Gameplay/Player/PlayerTexture.png

GOOD
/Content/Gameplay/Player/Textures/PlayerTexture.png

It is acceptable to organize by type for the final folder for an asset. Sense all types will be bundled under the role this is simply to make the Player folder a bit easier to maintain and look through. It should be noted that over using sub-directories can be just as bad as not using them at all. For example the following is a bad use of subdirectories:

/Content/Gameplay/Player/Weapons/Attacks/AttackSounds/AttackSound1.wav

The following path is an overuse of sub-directories that can cause another user to have to search through folder after folder for what they want. An easy fix for this is simply to move the weapons folder up to Gameplay, remove the Attacks folder and rename AttackSounds to Sounds.

Naming:

Files should be named in such a way that it describes what that file does without any other context. So a texture file should be named PlayerTexture.png or if it has a more specific use such as a normal map PlayerTextureNormal.png and such. Some tools will not display an assets full path so this will allow for an understanding of what the file is without having to look further into it. Likewise assets that are part of a set (such as multiple wav files that are variations on the same sound) should be named with the same convention, followed by _NUMBER, and then the file convention.
PlayerAttack_1.wav

Repo:

Commits:

Game assets should be committed when a task is either stable (can compile and run) or finished. **DO NOT COMMIT BROKEN CODE OR ASSETS THAT WILL BREAK THE ENTIRE PROJECT.** You should also only commit related things in a single commit. Generally this will mean one task per commit. The only situation where multiple tasks should be in a commit is if those tasks are inherently related (such as two bugs that were really one problem).

WHEN PUSHING ALWAYS USE RE-BASE, NOT MERGE.

Commit Procedure:

Before committing you should always pull from whatever remote branch you're on and test to make sure that nothing new breaks. If something new does break then fix it and commit that fix **FIRST** followed by your commit. (These commits in some situations can be the same if they are similar enough)

Commit Messages:

When committing to the repository you should **ALWAYS** have a commit message. This message should follow the following outline

Task(s): #ASANA_TASK_ID separated with space. This will place your commit in Asana as a comment under the task(s). If the commit is relevant to multiple tasks, then list them all here.

Write a comment stating what this commit does/changes/adds.

Issues: If there are any issues with this commit (such as a known bug) document it here

Tested: Comment here how this commit was tested. There should always be some form of testing

Branches:

Our project will use several different types of branches as listed below.

Master:

The master branch is the current development branch. This is where you will do most of your commits. The current project state will always be the head of this branch.

Version:

A version branch will branch off from the master branch and never merge back. These branches are created whenever a release build is needed. This branch should be as *Version-[VersionTag]* where the version tag is what this version is for. If this is a full release this should be of the form Major.Minor.Minor (such as 1.1.0). If the build is for something specific, then the version tag should reflect this (so for a BFiGs demo it should be Version-BFigsDemo).

When creating one of these branches you should add a tag to the master branch with the version as well so it can be easily found.

Feature:

A feature branch is a branch that will always branch from master and merge into master. This branch will contain a specific feature that is being worked on that can be worked on separately from the main game. An example of when this would be used is if you were writing some editor extension, then this would go in a feature branch sense it is not needed by anything in the master branch. They should be named as *Feature-[FeatureTag]* where the feature tag describes the feature.

Asana:

Priority:

A task can be given one of five priorities.

Very Low:

Low priority tasks are tasks that can be removed from the final project with very little impact. These tasks are mainly for features that are polish to the game and will be the first to be removed if time does not allow.

Low:

Low priority tasks are tasks that can be removed from the project if needed. This would be things like extra levels or items. They would change the overall game, however a workable game is still able to be made.

Normal:

Normal priority tasks are tasks that need to be finished by the end of the games development (or by a certain build number). These features have to be finished but are not hugely time sensitive and can be done in any order.

High:

High priority tasks are tasks that other tasks are dependent on being finished before they can progress. These features should be finished before normal tasks in order to not block other peoples work.

Very High:

Very high priority tasks are tasks that need to be finished ASAP. Very high tasks always need to have someone assigned to them, and if you are assigned a very high priority task you should drop everything as soon as you are able to work on it. These tasks should come up almost never, only when a feature is needed immediately in order to make the entire project work (such as when the repo is broken and such).

Claiming a Task:

When claiming a task, you should assign it to yourself so people know who is working on what. Tasks with no one assigned will be assumed to not be being worked on.

Assigning Tasks:

Likewise if you need a task done you should assign that task to another group member. It is also recommended that you inform them of such through other means so they know what tasks they have.

Finishing a Task:

When you finish a task you should **NOT** mark it as complete. Rather make a comment that says “Needs Review” and assign the task to another group member.

Review Task:

When reviewing a task you should check through all the changes done by that task. Make sure it works and everything is in check. If there is something you should feel should be changed either comment so and leave the task unassigned to be picked up later, or re-assign it to the original owner. If the task is ok, leave a comment saying so and then mark it as done.

Task Dates:

If a task needs to get done by a certain date, mark it as so on the calendar. It’s recommended that you do tasks as needed when assigned to you.