Technical Design Document (TDD)

Document Revision Table

Version	Description of Change	Name	Date
2.1	Editing pass, wrote notes for future changes	Curley	3/24/2016
2.2	Alpha update	Howell	4/14/2016
2.3	Editing pass	Curley	4/16/2016
2.4	Editing pass	Howell	4/28/2016
3.0	Proofreading pass for submission	Curley	4/28/2016

Project Goals

Bullpup Games plans to build a networked First Person Shooter (FPS), Capture-the-Flag (CTF) style game in the Unreal 4 engine, supporting two teams of four players (eight players total). The team aims to deliver two maps, each with a unique game mode: "Vendetta" (standard CTF) and “Bootlegger” (neutral flag variant). Vendetta involves both teams owning one flag, and both teams trying to capture their opponents flag. Bootlegger (neutral flag variant) is capture the flag with only one flag, but still two different bases.

Scope

Bullpup Games aims to deliver a product with all the features listed in End Product section. The project’s deadline is May 13.

The team will also deliver all required materials listed in the TPG2/3 Project Requirements Document. The requirements in the document are as follows:

- **Network Server**
    - A network server on which the game is played, supports up to 8 players.
- **Implement multiplayer Capture-the-Flag mechanics**
    - The game will have basic capture the flag mechanics implemented for one of its two game modes.
- **Implement multiplayer Bootlegger mechanics**
    - The game will have altered captures the flag mechanics implemented where there is one neutral flag which can be captured by either team.
- **Integrate weapons**
    - The game will have three weapons integrated by the programming team: Pistol, Tommy Gun, and Shotgun
- **Implement Main Menu**
    - The game will have a main menu interface before gameplay starts where the player can navigate the pregame options. This menu is where players host and join games.
- **Implement HUD**
    - The game will have an in game heads up display (HUD) designed by the design team and implemented by the programming team.
- **Integrate Sound**
    - Any and all sound effects for guns and player interactions and voice overs are integrated.
- **Implement Controls**
    - All basic player controls for both the main menu and players in game.
- **Auto-Installer**
    - Delivered with the finished game build.

Task Tiers

Scope for Bullpup Games divides task priority into four categories: Tier 1, Tier 2, Tier 3, and Tier 4. Bullpup Games accomplishes tasks in tiered order. Bullpup Games does not make progress through a Tier without completing all tasks in a previous Tier. Within each Tier, individual tasks are prioritized. If the project ends in the middle of a Tier, at least the most important tasks from the Tier are completed first.

Tier 1 - Bare Minimum Project	This is required before the product before the deadline. Without Tier 1, the game does not ship.
Tier 2 - Internal Goals	This is everything that Bullpup Games wants to accomplish before shipping the game. Without Tier 2, the game loses its unique identity.
Tier 3 - Stretch Goals	This is everything Bullpup Games would like to have in the game before the deadline, but they are bonus features and the game can stand on its own without them.
Tier 4 - Wish List	All the other tasks that would also be nice to have, but will not be completed before shipping. The tasks assigned as being Tier 4 sometimes shift places with tasks in Tier 3 and Bullpup Games completes milestones.

End Product (Tier 1)

Gameplay

- Server-side blueprints containing the following functions:
    - Player joins or quits game
    - Player spawns in game as actor

- Player dies and respawns
- While a player is dead they are a floating spectator
- Core capture-the-flag mechanics
- Game score, player/team statistics
- Mechanics
    - Running
    - Jumping
    - Firing
        - Primary
        - Secondary
- Reloading
- Gun Swapping
- Picking up ammo
- Picking up consumables
- Picking up flag
- Capturing flag
- Returning flag
- Game Objects
    - Player character
    - Flask (standard flag)
    - Crate (variant flag)
    - Pistol
    - Tommy Gun
    - Tommy Gun Ammo
    - Shotgun
    - Shotgun Ammo
    - Health Pickup
    - Speed Pickup

HUD & UI

- Player in-game HUD
    - Ammo Count
    - Current Weapons
    - Health
    - Flag Status
    - Score

Menu Systems

- Main Menu
- Multiplayer Lobby Menu
    - Host Game
    - Join Game
    - Options Menu
    - Credits Screen

End Product (Tier 2)

Gameplay

- Reloading Animations
- Weapon Swapping Animations
- End of game score screen
- Death Animation on death
- Character Voiceovers
- Walking sounds on different materials
- Different bullet holes on different materials

HUD & UI

- Directional Damage Feedback
- Score Screen
    - Tracks Kills
    - Track Deaths
    - Track Scores (When you capture a flag)
- Visual Status effects
- Player names and health on hover
- Global UI Notifications
- Scrolling status
- Pause Menu
    - Audio Options

        - Quit Game

Menu Systems

- Player Names
- Options menu
    - Fullscreen
    - Audio Options

Principle Risks

Platform Risks

Players playing on their own computer need to have good enough specs to run our game. Plans for a low/medium/high quality mode is currently Tier 3.

Engine Risks

Due to the short development schedule, the faculty/stakeholders advised teams to avoid making custom animations. All animations must be done with the existing character rig, and thus any custom reload animations must be considered Tier 2 or 3 features. Characters currently hold each gun with two hands.

In general, Bullpup Games is also at the mercy of the Unreal Engine development team. The team has decided to stay with Version 4.10.2 for the entire project. As such, any existing engine bugs with this version are a risk.

Middleware Risks

There should be no middleware risks since Bullpup Games creates all assets in-house and does not intend on buying anything from the Unreal Store.

The team derives technical details from Blueprints built into the Unreal Engine as well as Darwin, the prototype game created by Professor Skinner. It is pertinent that the programming team becomes familiar will all aspects of Darwin.

Other Principle Risks

Cel shading and the artistic direction in general is contingent on whether or not the programming team can deliver the shading effect within the timeline. Currently, the programming team can recreate crude version of the cel-shading effect. The team agrees to wait until Tier 2 to iterate this post-processing effect and conduct further gameplay tests.

Deliverables

Consult the Asset Database for a list of programming deliverables.

System Requirements

Target System

Bullpup Games has chosen *Fallout* 4 as a benchmark for *For the Family*'s system requirements:

Minimum

Processor: Intel Core i5-2300 2.8 GHz/AMD Phenom II X4 945 3.0 GHz or equivalent

Video Card: NVIDIA GTX 550 Ti 2GB/AMD Radeon HD 7870 2GB or equivalent

Memory: 8 GB

Operating System: Windows 7/8/10 (64-bit OS required)

Recommended

Processor: Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz

Video Card: NVIDIA GeForce GTX 980M

Memory: 16 GB

Operating System: Microsoft Windows 8.1 Professional Edition (build 9600), 64-bit

Display Maximum Resolution: 1920 x 1080

Development System

Processor: Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz

Video Card 1: NVIDIA GeForce GTX 980M

Video Card 2: Intel(R) HD Graphics 4600

Memory: 16 GB

Operating System: Microsoft Windows 8.1 Professional Edition (build 9600), 64-bit

Display Maximum Resolution: 1920 x 1080

Resource Budget

Frame Time Budget

For a real-time app running at 60 frames per second (or a minimum of 30), we will have at most 16 milliseconds (or 33) per frame of total operating time (per thread, maximum, best case).

The frame time budget breaks down as follows:

Description	milliseconds
Game Logic	2
Player Logic	1
Server Communications	5
Render Environment	3
Render Players	1
Render Shaders	1
Render UI	1
Render weapons (first-person)	.5
Total	14.5

Memory Utilization Estimate

For the Family runs on 1 GB of memory, with the following component breakdown:

Description	Bytes
Mob Player Character	50MB
Pistol Weapon	25MB
Tommy Gun Weapon	25MB

Molotov Launcher Weapon	25MB
Arms / Hands	5MB
Environment	150MB
Audio	100MB
3D character models	175MB
Particle systems	20MB
Total	575MB

Assets Budgets Estimate

The theoretical maximum number of visible polygons per frame (at 30 frames per second) is 100k triangles, broken down into the following components:

Description	Triangles
Mob Player Character	10K
Pistol Weapon	5K
Tommy Gun Weapon	5K
Third Weapon (Molotov Launcher)	5K
Arms / Hands	1K
Environment	40K
Total	66K

Art Assets Budgets Estimate

Texture size limit: 8192 x 8192

Each environment assets:

- Max 1K triangles
- At most player will see 40 different static meshes on-screen

Environment:

- Max number of dynamic lights with shadows 4

Technology Sources

Acquired

Software Name	Description
Unreal Engine 4	Video game engine created by Epic Games. Used to build the core game and integrate all assets created in other programs.

Art Creation

Software Name	Description
Adobe Photoshop	Raster graphics editor
Autodesk 3DS Max	3D graphics editor (models)
Autodesk Maya	3D graphics editor (animations)
Crazybump	Imaging tool used to create normal / bump maps for textures
Mudbox	Sculpting tool to create hi-res models and concepts

Level Design

Software Name	Description
Adobe Photoshop	Raster graphics editor
Adobe Illustrator	Vector graphics editor
Autodesk 3DS Max	3D graphics editor (models)

Software Engineering

Software Name	Description
Microsoft Visual Studio	Software used to edit C++ code used in Unreal Engine 4

Audio

Software Name	Description
Audacity	Audio recording and editing software

Miscellaneous

Software Name	Description
Perforce	Version Control software, used to managed team-wide synchronous work
Swarm	A program used to decrease build times in Unreal Engine 4 by distributing the compilation work to all machines within the system
Microsoft Word	Word processor used to for major project management materials
Microsoft Excel	Spreadsheet software used to track and plan sprints
Microsoft PowerPoint	Slideshow software used for team presentations

Components

All Blueprints and descriptions can be found on the Asset Database.

Software Configuration Management

Naming Convention

- All filenames must conform to those printed in the Asset Database
- Use CamelCase when naming files
- All filenames must only contain alphanumeric characters
- No spaces in filenames
- Use underscores to separate between DataType, Name, and Asset Type
    - For example: “T_HealthBar_Diffuse”
    - The Asset Type only needs to be denoted if there is more than one asset type to differentiate between
- For assets with more than one variation, use sequential numbers to denote different versions
- Use 1 as the first variation. If there is no second variation, you do not need to use a number.
- Names begin with a noun, followed by up to two modifiers, then version number. The modifiers should be in order of importance and should be consistent amongst other similar files. [Noun][Modifier][Modifier][#]
    - For example: “M_CarSmallBlack2”

Category/File Formats

Category	Kind of file
T _	Textures *Example: T_HealthBar_Diffuse*
M_	Materials *Example: M_CarSmallBlack2*
BP_	Blueprints
LVL_	Levels
SK_	Skeletal mesh
SM_	Static mesh
A_	Animations
SFX_	Sound effects
CUE_	Sound cues
MUS_	Music
P_	Particles
HUD_	HUD widgets
FileType_	Catch-all for any type not included above. Usually there are very few of these. *Example: GameMode_CTF*

Build Plan

The Lead Programmer (Clay Howell) is the build master. Every week on Saturday, at the end of core hours, the build master makes a clean build of the game in its current state. The build master ensures there are no critical bugs or problems with the build before submitting it to Perforce, in a folder under the root directory name ‘Builds’.

At the end of each milestone, a milestone build is made, and is a playable demonstration of the project at that point in time. All team members must play and test the milestone builds before the lead programmer submits the build to Perforce.

Backup Routines

In case Perforce were to fail, the lead programmer makes a local copy of all the contents of the team’s Perforce repository on an external hard drive. This backup is made every Saturday at the end of the work day.

PC Images

Art	Level Design	Programming	General

Autodesk Maya 2016	Adobe Photoshop 2015 v8.1	Microsoft Visual Studio 2015	Unreal Engine 4 4.10.2
Autodesk 3DS Max 2016 18.0 sp1	Adobe Illustrator 2015		Microsoft Word 2013
Adobe Photoshop 2015 v8.1	Autodesk 3DS Max 2016 18.0 sp1		Microsoft Excel 2013
Adobe Illustrator 2015			Microsoft Visio 2013
			Microsoft PowerPoint 2013
			Audacity 2.1.1
			Perforce P4V 2014.3
			Cisco AnyConnect VPN
			Swarm
			Slack

Personal Backups

Team members encountering issues with Perforce must immediately talk with the lead programmer. The goal is to avoid the loss of significant progress on work and to facilitate collaborative work with Perforce.

During the workday

Team members will regularly back up all the content that they work on for *For the Family*.

1. When the workday starts, all team members will Get Latest from Perforce
2. After a team member completes a task, they are expected to commit that content, and update the respective documentation and scrum boards

At the end of the workday

Team members back up all the content that they work on for *For the Family*.

1. Always commit any work you're currently working on before you leave core hours
2. Do not have files checked out for long periods outside of core hours

Per Milestone

1. A current working build of the game will be uploaded to Perforce

Coding Standards

- Use OnTick as little as possible (it's costly)
- Put all code inside comment boxes
- Make liberal use of comments on individual functions themselves
- Do not cross connection lines unless absolutely necessary
- Keep connection lines organized, keep functions aligned
- Try to be consistent with blueprint organization across all blueprints

All code adheres as strictly as possible to the UnrealScript coding standards.

Version Control

How to set up a new workspace:

1. Click the workspace drop down
2. Click ‘New Workspace…’
3. Change the workspace name to [username]_BullpupGames
4. Paste this into the workspace mappings:

-//depot/... //[username]_BullpupGames/...

//depot/C25/Students/[username]/TGP2/ForTheFamily... //[username]_BullpupGames/...

//depot/C25/TGP/TGP23/ForTheFamily/... //[username]_BullpupGames/...

-//depot/C25/TGP/TGP23/ForTheFamily/Config/... //[username]_BullpupGames/Config/...

-//depot/C25/TGP/TGP23/ForTheFamily/Intermediate/... //[username]_BullpupGames/Intermediate/...

-//depot/C25/TGP/TGP23/ForTheFamily/Saved/... //[username]_BullpupGames/Saved/...

-//depot/C25/TGP/TGP23/ForTheFamily/Saved/* //[username]_BullpupGames/Saved/*

//depot/C25/TGP/TGP23/ForTheFamily/Content/... //[username]_BullpupGames/Content/...

5. Click ‘OK’

How to get the current data off the Perforce Server

1. Open the Perforce Visual Client
2. Open up your current workspace
3. Right click the root file for the repository
4. Choose ‘Get Latest Revision’

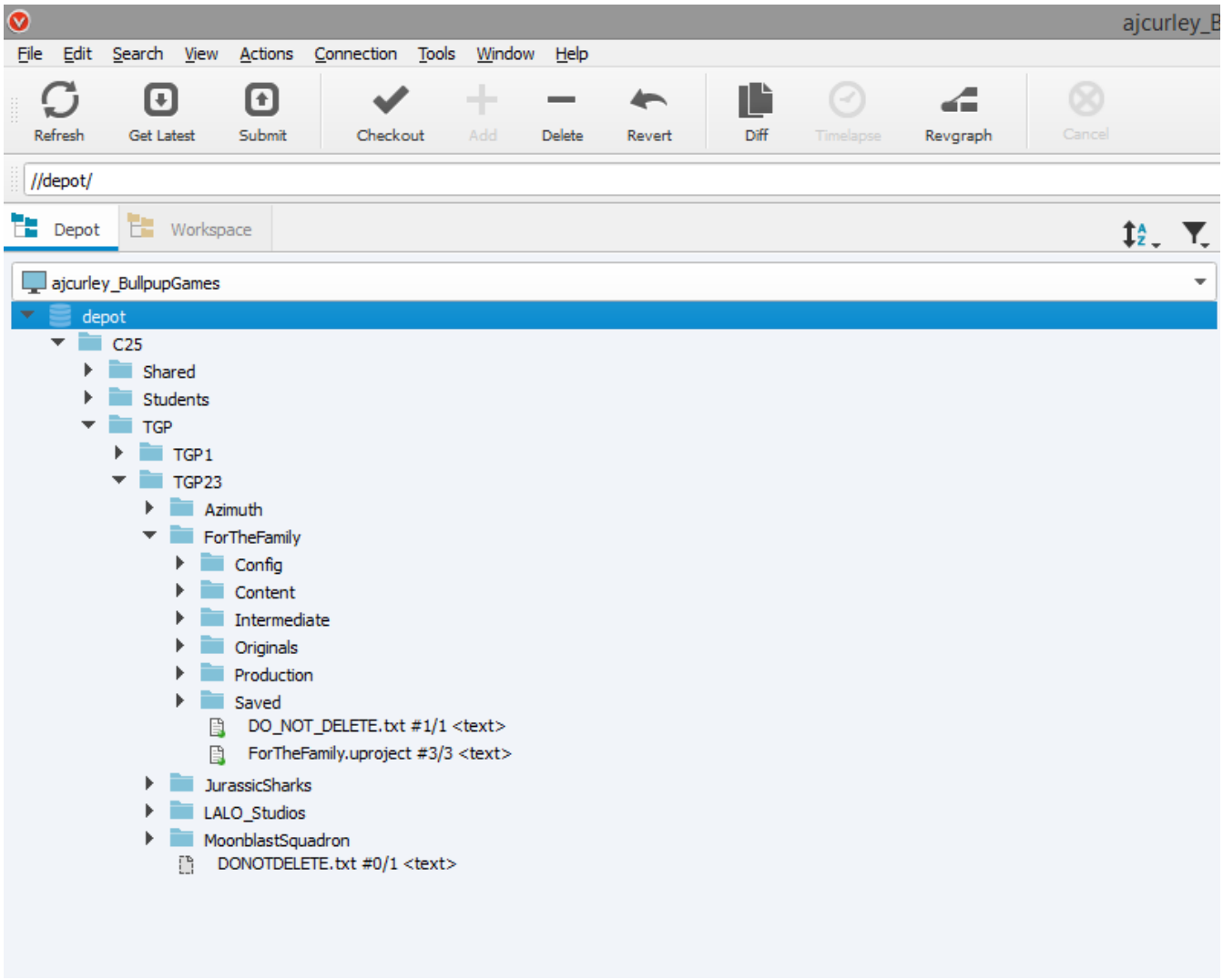
Perforce File Structure

Bullpup Games uses the following file structure in Perforce:

- Config
- Content
    - Animations
        - EXP
- Blueprints
    - Enums
    - EXP
    - GameModes
    - GameplayEntities
        - Projectiles
    - Pickups
    - Player
    - UI
- Collections
- FirstPerson
    - Animations
    - Audio
    - Character
        - Materials
            - MaterialLayers
            - Mesh
            - Textures
    - FPWeapon
        - Materials
            - MaterialLayers
            - Mesh
            - Textures
    - Meshes
    - Textures
- FirstPersonBP
    - Blueprints
    - Maps
- Geometry
    - Meshes
- HUD
    - EXP
- Levels
    - EXP
- Materials
    - EXP

- World
- Meshes
  - SkeletalMeshes
    - EXP
      - AnimCharacter
        - Animations
        - SkeletalMeshes
  - StaticMeshes
- Music
  - EXP
- Particles
  - EXP
- SoundEffects
  - EXP
- Splash
- StarterContent
  - Architecture
  - Audio
  - Blueprints
    - Assets
  - HDRI
  - Maps
  - Materials
  - Particles
    - Materials
  - Props
    - Materials
  - Shapes
  - Textures
- Textures
  - EXP
  - World
  - Intermediate
    - Config
      - CoalescedSourceConfigs
      - ReimportCache

- Originals
  - Concepts
  - Meshes
    - SkeletalMeshes
    - StaticMeshes
- Production
- Saved

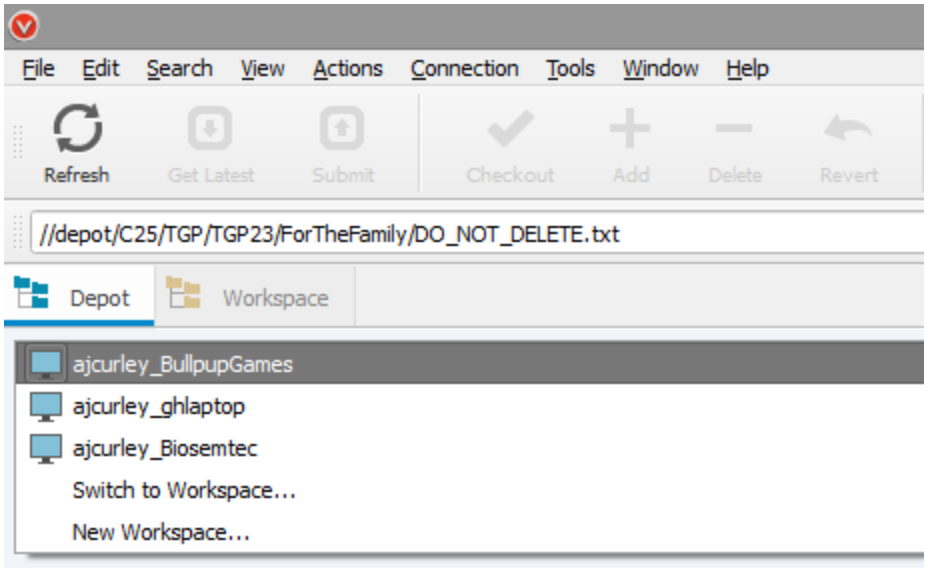


Sample SVN Repository

Perforce Tutorial

Connect to the Network Repository

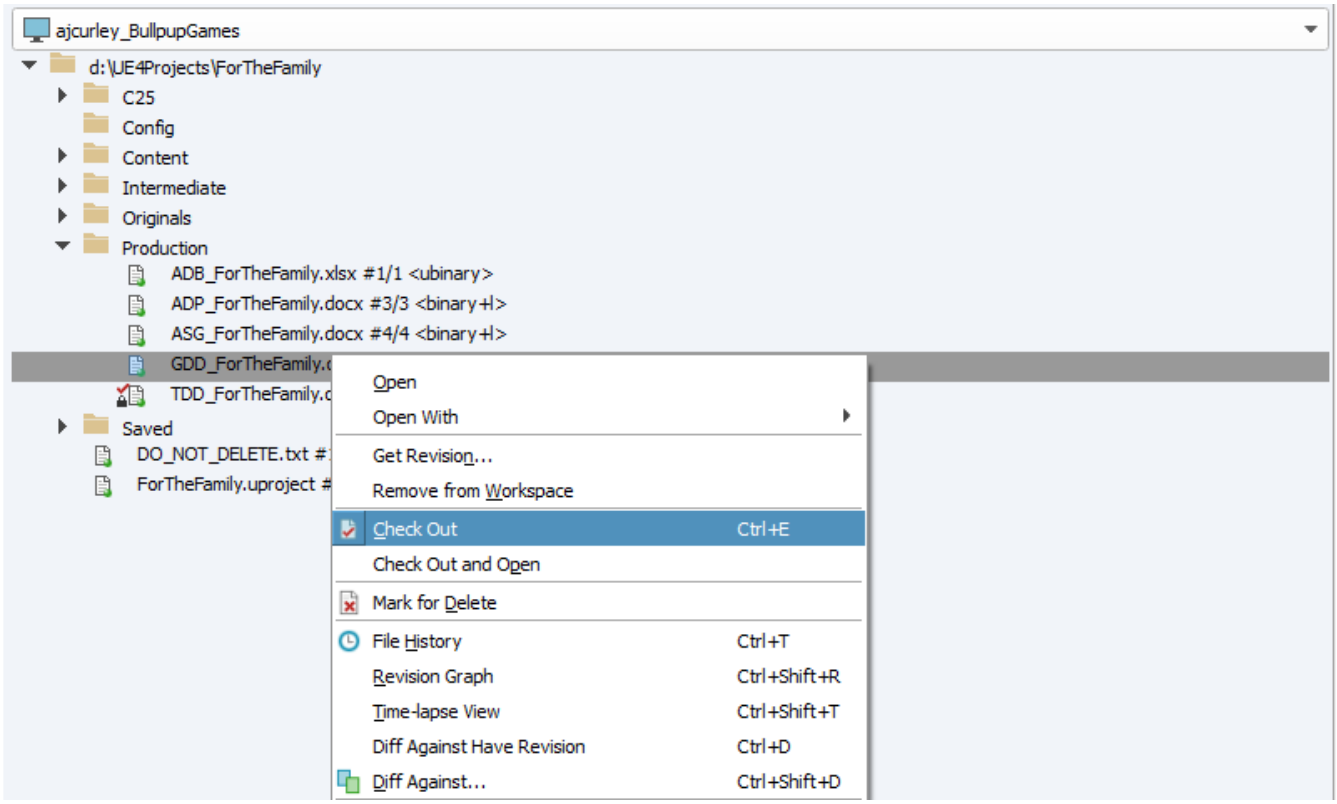
1. Open Perforce Visual Client
2. Enter SMU username and password
3. Make sure you are using [user]_BullpupGames for your workspace
4. You can change your workspace by clicking the drop down at the top of the repository



Workspace dropdown menu

Initial Check Out from the Network Repository into the Local Working Directory Repository

1. Find the document that you want to work on in the repository
2. Right Click the file and click ‘Check Out…’
3. Confirm the check out by click ‘OK’
4. The file should now have a red checkmark in the icon on the left

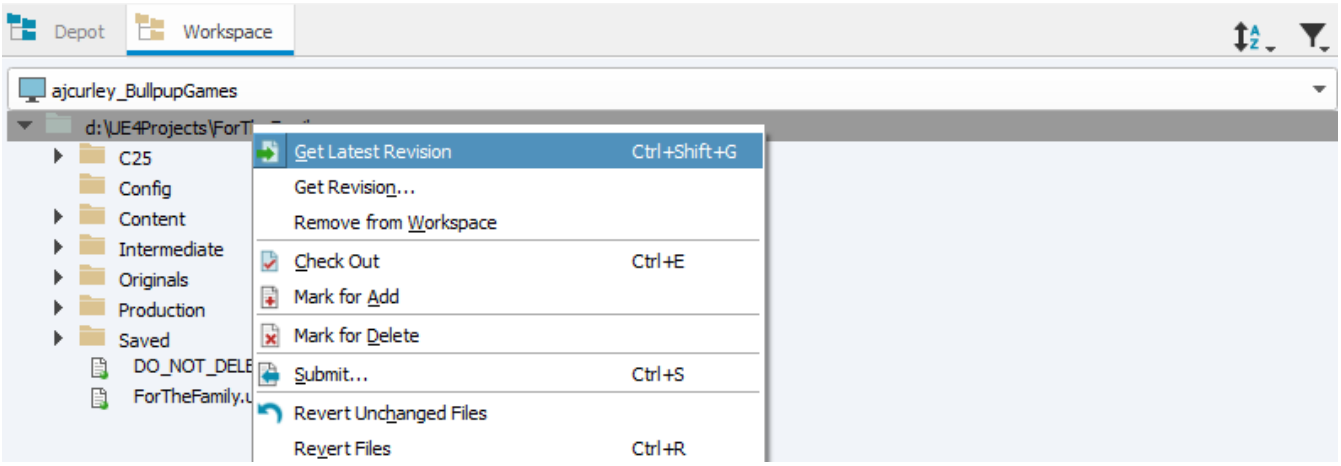


Check Out command

Updating to the Latest Files from the Network Repository

1. With Perforce Visual Client open, click on the Workspace tab at the top next to the Depot tab

2. Right click the root folder and choose ‘Get Latest Revision’
3. Alternatively you can also left click the root folder, then click the at the top under the toolbar labeled ‘Get Latest’

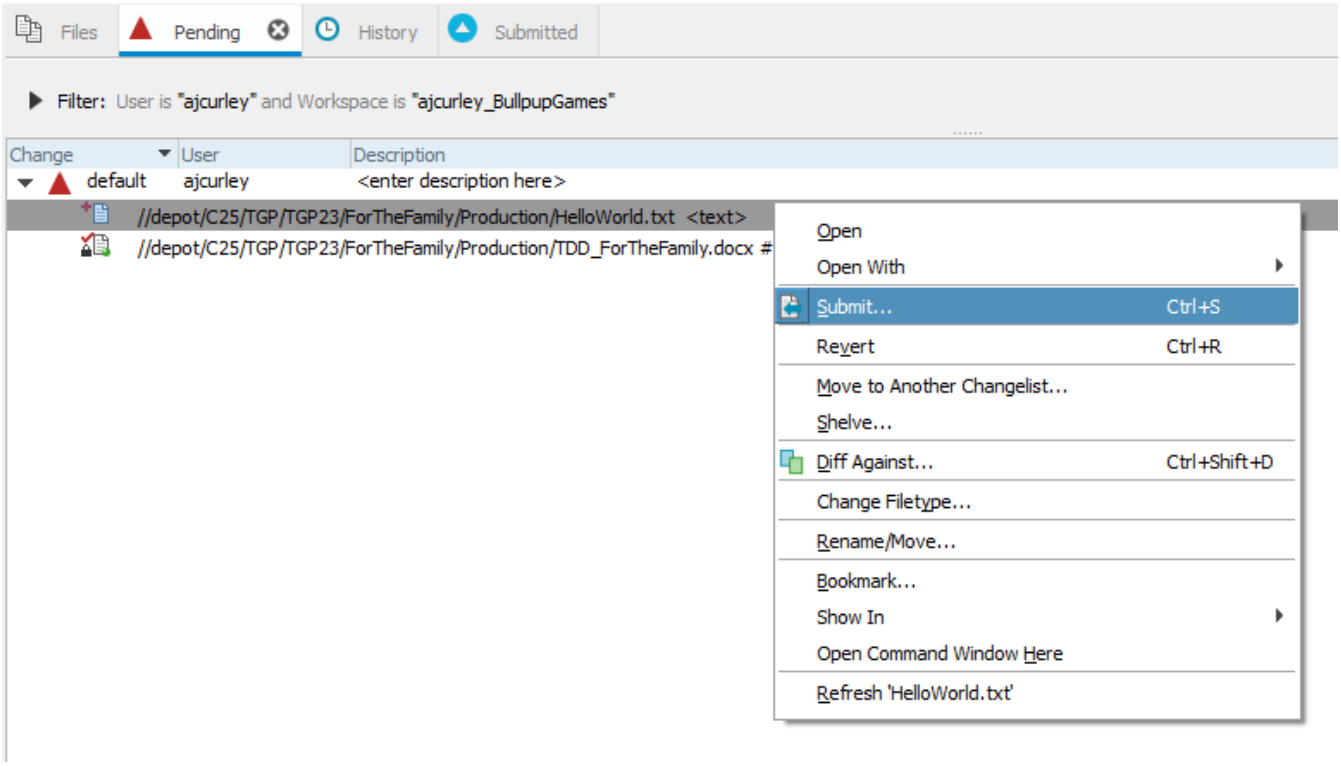


Get Latest Revision command

Adding a Single File to the Local Repository

If you’ve create a new file within the your Bullpup Games workspace, you need to add it to Perforce so it can track the file and all changes made to the file

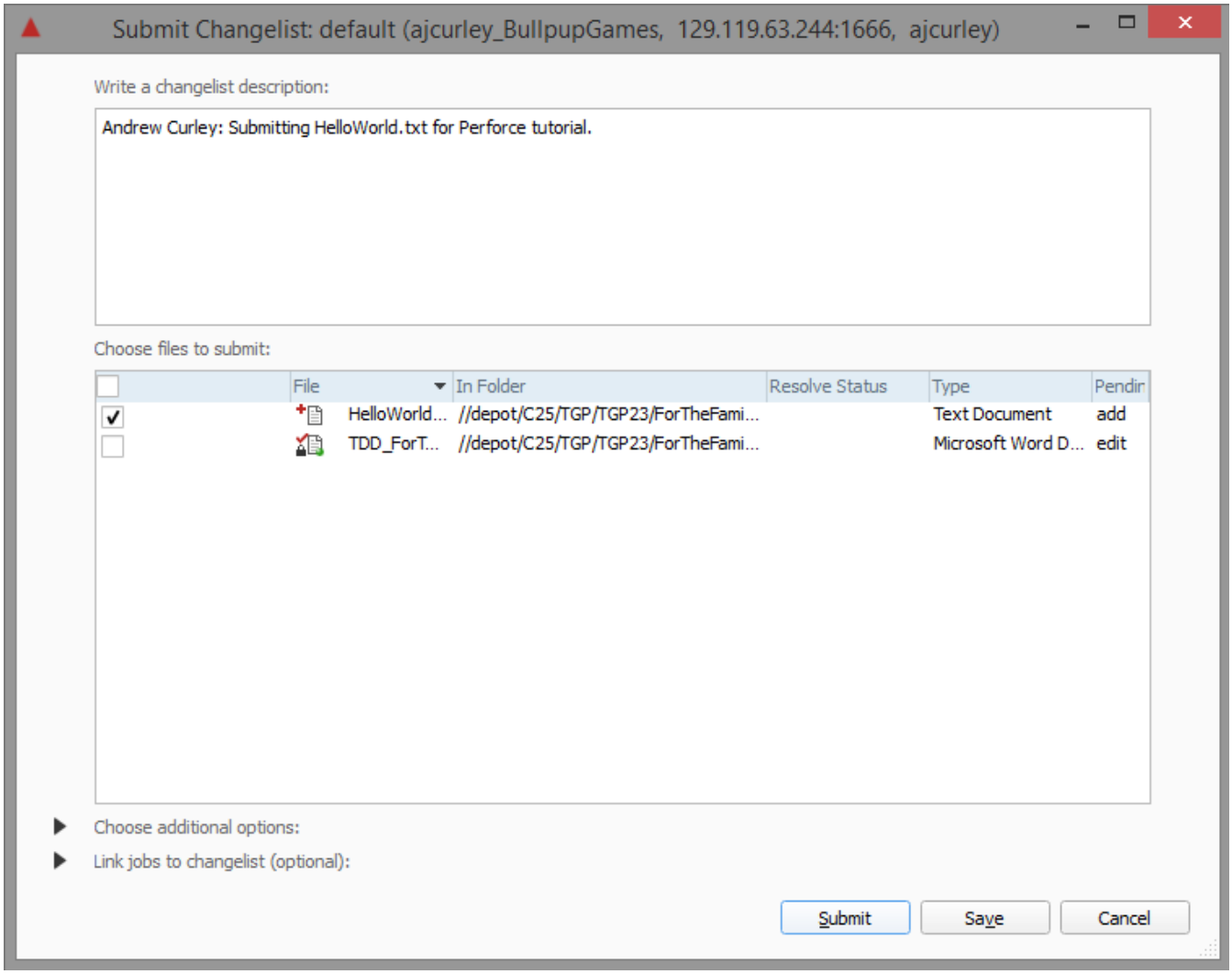
1. First, make sure you’re on the Workspace tab and not the Depo tab
2. You should see the file in your repository but the icon will be faded, indicating the Perforce is not currently tracking the file
3. Right click the file and choose ‘Mark for Add…’
4. In order to confirm this action, you must navigate to the right side of the client where it shows your pending items
5. If you do not see your pending items, click the tab with the red triangle called Pending
6. Select the change list that contains the file add action or select just the add action under the change list if there are multiple actions listed under the change list
7. Right click the selected item and choose ‘Submit…’



Submitting a local file to Perforce repository

Committing Files to the Network Repository

1. When your Perforce Visual Client is open, on the right will be a list of all pending files ready to be checked in to Perforce
2. Right click the change list you want to commit to Perforce and choose ‘Submit…’
3. A confirmation box will then appear
4. Fill out the description
5. Uncheck any items you do not want to commit
6. Click ‘Submit’



Submission confirmation box

Handling Version Conflicts

1. When a conflict occurs, the person who is checking in the object last is at fault
2. Find the person who checked in the asset before you, and ask them what was changed
3. If important work was done, revert your work, get latest on the assets, and the redo your work
4. If the person before you confirms that the work they did was not important, you can overwrite their work with your work
5. To prevent conflicts, submit your work often, at least once an hour

Quality Assurance

Component Testing

When a developer believes a component is ready to be implemented in the game, they must first test the component’s functionality within the programming zoo in their local editor. The developer then tests the component over networked play in order to determine if it works under a range of likely circumstances. After the component is proven to work locally, the developer moves the asset to the appropriate folder in preparation for integration testing.

Integration Testing

Integration testing occurs once per week, during which the entire team suspends all work and playtests the current build, with specific attention to the new components and their interactions with all other assets in networked play.

Any and all components successfully tested must be moved to the appropriate folder and committed to Perforce according to the correct naming convention outlined in the Software Configuration Management section.

Useful Tutorials

https://wiki.smu.edu/display/guildhall/Networking

https://docs.unrealengine.com/latest/INT/Engine/Audio/index.html

https://wiki.smu.edu/pages/viewpage.action?pageId=86311344