A2 - Production Workflow Report

This task has a timeline of four weeks. It is allocated to Week 3, 4, 5, and 6.

Table of Contents

Context

The larger context of **games production** includes being able to actually **do game development activities as well**.

A dedicated role of a producer only really emerges when teams get significantly larger to result in the need for someone to dedicate their time/resources to focusing on making sure the whole process is **running smoothly**, **on time**, and is **coordinated**.

Workflows in Professions

In order to get a deeper understanding on production, it's important to actually do the skills of asset creation/development into the bigger picture of actually putting a game project together - especially understanding it from these main aspects:

- Visuals (2D, 3D, animation, shaders, post-effects, etc.)
- Audio (sounds, music, audio processing, dynamic audio/code, etc.)
- Code (mechanics architecture, systems programming, glue-code, etc.)
- Design/Testing (prototyping, mechanics tweaking, test designing and deployment, etc.)
- Writing (Localisation, dialogue systems, script markup/code, etc.)

Using Version Control Systems

One of the main ways that a project can be tracked in a *very* tangible way, is through the use of **version control systems** (such as git, perforce, SVN). Even solo devs will use these systems as they double as a solid back up *with* a usable history, and it enables projects to scale up their teams easily to over hundreds of people.

- Version control systems, game engines, and asset creation tools make up the bread and butter of games development.
- The other elements of time tracking, task tracking/management/planning tools, and knowledge base tools are what ends up fleshing out the rest of the overall picture of games production.

Aim

The purpose for this task is to undergo and reflect on activities related to development practices in games development, and to experience overseeing the process using version control tools.

The aim for this task is to be able to collect evidence of a project's progression over time, and to document increases and decreases of project activity, in order to gain insights into production practices. This can then provide evidence to base future improvement suggestions off.

The focus of this task is to understand version control in the context of games production, however a significant portion of understanding the context requires you to do various tasks within/for the game engine, from other tools.

The approach you should take with this task is research and exploration in various disciplines in games development, whilst making sure the production and tracking is done consistently (note, production and tracking is not specific to a single discipline in games development - it basically applies to all areas).

This should feed into the later parts of the semester when you will need to develop a more cohesive project more fluently.

You are required to work in teams of 3 to 6 for this task.

Task Overview

Across the team, all members will be doing the following:

- Exploring a selected game engine, and figuring out specific tasks related to various discipline areas of choice
    - Areas and tasks will be supplied in another document/template
- Using version control in the following contexts
    - Industry practices with collaborative, and/or remote working
    - Parallel, asynchronous development practices
    - Snapshotting, backing up, and full history of versions
    - Task tracking and logging

The team will need to decide on a couple of tools to try for iterations of a project (more details on this later), and proceed to use and evaluate these tools, whilst collecting evidence of both the issues/failures and successes in using these tools.

This assignment has two components. A team component and an individual component.

Steps

Game engines have such wide areas to explore since it's where most of the disciplines will converge to.

Preparation: Decide on Game Engine, and Version Control System

Think about what game engine to explore for this project. You are also allowed to change engines later for whatever reason but will need to document reasons/process in your report.

You will also need to explore and use a version control system, too.

This is one of the main components to practice using in this task, and your usage of it is directly assessable.

Recommended Game Engines: **Godot, Unity3D, Unreal Engine, etc..**

Recommended Version Control Systems (VCS):

- Git. Service providers include.
    - Github.com, Bitbucket.org, GitLab.com
- Perforce
- SVN

Depending on the engine and VCS you decide, it may influence who you team up with for this assignment.

Step 1: Form New Teams

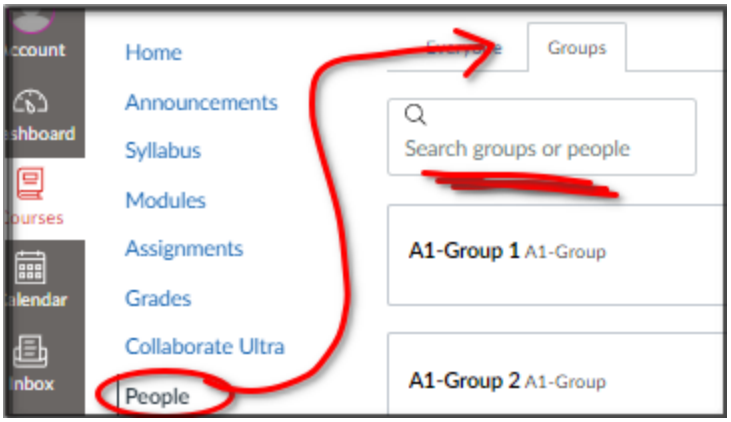
Consider a variety of professions or disciplines for the team. Even if you're all in the same area, you can try different things in this project - it's encouraged to try new areas for this assignment, in particular. The team size limit is listed above in the Aim section.

Step 2: Register Team Members to Canvas > A2-#######

Note: This step's information is identical to Step 2 in Assignment 1, except for team names. The team name must initially start with "A2-" before renaming.

This is important to do since only one person submits the document for the whole team through canvas.

Head to **Canvas > People > Groups**, and find an available team slot to join.
If the list is too long, you can **search** for something like **A2-Group [number]**
eg. "A2-Group 1", "A2-Group 2", "A2-Group 15", etc…



For Canvas, the first person who joins the group can rename the group. You can leave it at the default name, or can rename it to a unique name for your team, be sensible with the name, but you're welcome to be creative since the team name will only be for this task.
If you want to keep a default name, that's also fine. Just be sure you're in the same group number.

Step 3: Create a Shared Document

Similar to Assignment 1, you will need a common document that you can work on together.

Two major example tools are below.

- Microsoft Office > Microsoft Word Online
- Google Drive > Google Docs

(You can use alternative tools that offer similar functionality, but make sure the whole team agrees to use it, and run it by your tutor to double check)

This document is for the group component of this assignment.

Step 4: Import Group Production Template

There should be a text template on Canvas for the group document.

You will need to follow the template and format it into a presentable report.
It will cover the decisions made by your team on what you plan to do in the coming weeks.

Keep the questions/prompts in the report. Reports should keep its full context in place.
Do not omit/remove the template questions/text from your report.

Step 5: Select Your Skills/Tasks to Do

There should be a text template on Canvas for available skills/tasks to do from various disciplines.

The template will have a number of slots that each member needs to fill by copy/pasting different skills/tasks under their name. Make sure to copy the item number as well (eg. AU01, AR01, PR02, etc.)

Note that if you have time and wish to try more, you can do more than 6 skills and add it to your submission as well.

Note that the allocations aren't set in stone. If you find a task too difficult or are no longer interested in it later, you can change what you are allocated, and edit the document to say you switched to another task.

Tips/Restrictions:

- Try to select a range of skills across the team - and some tasks will require more than one person to achieve well, so discuss with your team members what tasks to try.
- You are welcome to have members with the same item number. However all members with the same code will need to do their own version of the task/item.
- Avoid having too many people on the team doing the same task types/item numbers. As a general guide, at least 2 of the items per person should be unique to them.

Step 6: Set Up and Use a Version Control Repository

For your chosen version control system (VCS) above, search for a **service provider/host**, and create accounts to use it. It will be worth to search “how to create a repository in [VCS]”

You will also need to search/use a **desktop application/client** to connect to the **VCS server**.

For the git VCS, a comprehensive list of desktop clients that work for any git host can be found at https://git-scm.com/downloads/guis

- Some notable ones are GitHub for Desktop, Sourcetree
- You can also use the Git Bash/the terminal to work with git more directly
    - All desktop apps for git should also have terminal/command line access as well

Setting up a Repository for the Team:

1. One member needs to **create** a repository in the VCS website, and the other members need to be added to get access - explore the website!

2. Even though the repository is ‘empty’, it will need to be ‘cloned’ (i.e. downloaded on your hard drive through the VCS client).
   This has to be done for every team member’s computer.

3. Once the folder exists on your computer, you can place files in/delete files, and then manually push/pull/sync changes to/from the server. (Note that the terms may differ depending on VCS).

Before proceeding, it’s worth testing that all team members can do the following with the desktop client and the repository:

- “Push” at least 1 file to the repository, so when viewed on the cloud/website, the file appears. It doesn’t matter what the file is.
- “Pull” from the repository, so that files that others have pulled appear on your computer.

There are particular workflows on when you can push and pull - experiment to try to solve this for all members in your group! The push and pull logs will be the evidence that will end up going in your group report.

Other actions to explore over the project are

- Branching
- Merging (and merge conflicts)

Figuring these out can be quite technical and finicky, but as a ***very*** common industry practice it’s extremely important to learn.

Be resourceful!
Ask other peers or your tutor for fresh eyes on errors if you're not sure how to resolve them.

Especially early on, it's worth figuring out these since the repository will be mostly empty anyway - so starting again to re-trace the problems to troubleshoot is a very effective way to figure out what is going wrong.

The usage of the service/tools (via logs) is part of the assessment, so all members must demonstrate using it.

Step 7: Setting Up Game Project and File Ignore Filters

If every team member is able to push/pull/sync from the repository, the team is ready for this step.

All game engines will require creating a project to start with. In your chosen game engine, have ONE member create the project on their computer, to be uploaded to the VCS server - **do not push the project to the server yet**. Do the next step first.

Setting Up "ignore" configurations

Many game projects have temporary files that **should not be synchronized** between team members (such as shader cache, temp libraries, etc.)

You will need to make sure your project folder is correctly set up - for the git VCS, the usage of a ".gitignore" (called a "git ignore" or "dot git ignore") file needs to be added to the base of the project.

This is a step that is not frequently done (since it's only done at the start of the project), so it's very common to simply search up the template (as it changes often when the engines update). Search up a template/how to get things set up in your engine!

Search Term: "**[Game Engine] [VCS] ignore**".. You can even tack 'how to' at the start/end.
Eg: "Godot git ignore how to", "Unity perforce ignore how to", etc.

To know if your ignore works, push/sync files to the server, and check on the website that there are some files or folders that would not be on the server but still on your computer.

Once a game project can be synchronised across the team, you are ready to work on your selected skills/tasks! Make sure you work on the same game project - and push/pull/sync the work. The push/pull/sync logs will be used for assessment!

Step 8: Create a Personal Document and Log Your Findings

You will also need to create a personal document where you will be collecting evidence of your own work and findings/answers.

There are **two** templates on Canvas to be completed for this task's individual component.

- One template is a document that should be **duplicated for each skill/task you try**
    - If you try 6 skills/tasks, you should have 6 files (not counting the reflection doc)
- **The other template** is a reflection/summary, which is to be done after you do your individual tasks

These documents are for the individual component of this assignment.

Step 9: Push/Pull Your Work from the Version Control Repository

This step is an ongoing process each step/week/session of work.

As you complete your tasks, make notes on how you did the task (and any useful links you used, whether using tools like AI helped or not). Essentially use the personal document as a log of how things were done.

This document is for the individual component of this assignment.

Remember that if you are struggling with a task and find another task more interesting, you can log information about the attempts you did, and select another task to work on in its place.

Step 10 [Optional/Extension]: Time Track Your Activities

When working on your list of tasks, a recommended activity to do is log your time. This can be a challenging task, so it's an optional/extension task you can use to bolster your report. Otherwise this is a skill you can learn to develop later on, and over time.

If you use any time tracking tools, submit screenshots of reports generated from it to bolster your submission.

Submission (Group Component)

You can resubmit the files for the assignment as many times as you need before the due date. Make sure to re-upload ALL the files each time you resubmit.

Check the following:

1. Double check that **all** the members in your team are registered on Canvas correctly
2. **Only one member needs to upload the files for the team.**
   Canvas will submit for all team members when one person uploads
3. Once uploaded, **the rest of the team should check** that the submission is openable/readable through Canvas

If you are not in a group on canvas that has submitted a file, you may end up with a zero score for this task.

You will need to submit the following files for the group submission

The individual components are submitted in a different page on Canvas (see next page)

1x PDF	This is the skill selection and group document itself. The naming convention for the file should be: **<LabDay><LabTime>_<TeamName>_SkillsReport.pdf**
Additional Files	Any appendix items, such as logs/chats with any generative AI or other tools. If you are unsure how to export the logs (if the text is so long it goes off the page), ask your tutor to assist you… or possibly ask the AI? The naming convention for the file should be something like: **<LabDay><LabTime>_<TeamName>_<ItemName>.pdf** **eg:** **Wed1130_TeamName_ChatGPTLog1.pdf** **Thu1130_TeamName_ChatGPTToolsQuestion.pdf**

** Note: If the team name is long, you can abbreviate it or shorten it for the filename. Keep the full name in the cover sheet.

Submission (Individual Component)

You can resubmit the files for the assignment as many times as you need before the due date.
The group component is submitted in a different page on Canvas (see previous page)

You will need to submit the following files for the individual submission

Multiple PDFs	Your task reports, one for each 'skill/task' you wrote about. The naming convention for the files should be: **<LabDay><LabTime>_TaskReport1_AB##.pdf** **<LabDay><LabTime>_TaskReport2_AB##.pdf** **<LabDay><LabTime>_TaskReport3_AB##.pdf** **…etc.**
1x PDF	The Workflow Reflection report. The naming convention for the file should be: **<LabDay><LabTime>_WorkflowReflection.pdf**
Additional Files	Any appendix items, such as logs/chats with any generative AI or other tools. If you are unsure how to export the logs (if the text is so long it goes off the page), ask your tutor to assist you… or consider asking the AI. The naming convention for the file should be something like: **<Day><Time>_<YourName>_<ItemName>.pdf** **eg:** **Wed1130_AName_ChatGPTLog1.pdf** **Thu1130_AName_ChatGPTToolsQuestion.pdf**

** Note: If the team name is long, you can abbreviate it or shorten it for the filename. Keep the full name in the cover sheet.

Production Skills Template (Group)

The text below is what you will find in Canvas. It is mainly here for reference.

```
# Assignment 2.1 - Production Workflow Report (Group Component)
## Cover Sheet
___

### Team Name
- (ID) Name
- (ID) Name
- (ID) Name
…

Class Time: Day-Time
Tutor Name: Name
___

# Section 1: Group Skill/Task Selection
List each team member and their selected skills they intend to explore for this project.

(Team Member 1)
1. XX##: ((copy/paste the whole description + item code on the left from the skills/task list template))
2.
3. ...
4. ...
5. ...
6. ...
…

(Team Member 2)
1. XX##: ((copy/paste the whole description + item code on the left from the skills/task list template))
2. (skipped) – XX##: ((copy/paste the whole description + item code on the left from the skills/task list template))
3. ...
4. ...
5. ...
6. …
7.  (yes you can add,or change skills as you go, but make notes if you skipped or stopped a skill attempt)

(Team Member 3)
etc....
___

# Section 2: Tool Selection Questions
- Q: Which Game Engine did your group decide to use for this task? Why? Also mention if you considered other engines and why.
> A:

- Q: Which Version Control System did you decide to use for this task? Why? Any considerations for other VCSs?
> A:

- Q: Who contributed to the process of setting up the initial repository? Have they worked with any VCSs before?
> A:

- Q: How was the process of setting up the 'ignore' configuration in your chosen VCS? Easy/hard? Makes sense? Still confusing?
> A:

- Q: In terms of scheduling, do you have any plans about when people will work together? Or will you be mostly working in your own schedules? Or just going free form?
> A:
```

Task Report Template (Individual, multiple docs)

The text below is what you will find in Canvas. It is mainly here for reference.

 # Assignment 2.2 - Task Report (Individual Component)
Cover Sheet
___

Team Name
- (ID) Name
- (ID) Name
- (ID) Name
…

Class Time: Day-Time
Tutor Name: Name

___

 # Section 1: Skill/Task Summary
- Q: Which skill/task is this report for? (Copy/paste the skill/item code here. It should really only include one skill per report. For multiple skills, have multiple reports)
> A:

- Q: Did you have any experience or assistance relating to this task?
> A:

- Q: How easy/hard did you find this task? Was it how you expected/was it weirder than you thought, going in?
> A:

- Q: Attach a screenshot of the Version Control system showing that the work assets have been 'pushed' to the server (i.e. all git commit/push logs, perforce logs, etc.) - the work can be finished/unfinished. If you also did time tracking on this task, put a screenshot of the time logs here as well.
> A:

___

 # Section 2: Logs and Notes
Tips:
- Compile notes/screenshots on how you did this task, and what you explored. Whoever reads your notes should get a sense of how you approached this task
- Report about key steps/troubles/solutions you came across doing the tasks.
- After each section/skill, comment about how easy, or difficult (or weird) the steps may be, to do the task/skill.
- Consider taking screenshots of important settings instead of just writing all the steps out - but if you take screenshots, at least caption them with 1-2 sentences of what's going on in the screenshot.

Workflow Reflection Template (Individual, 1 doc)

The text below is what you will find in Canvas. It is mainly here for reference.

```
# Assignment 2.3 - Workflow Reflection Report (Individual Component)
## Cover Sheet
___

### Team Name
- (ID) Name
- (ID) Name
- (ID) Name
…

Class Time: Day-Time
Tutor Name: Name

___

# Section 1: Skill/Task Selection
For easier reference, copy/paste your selected skills here from your group submission.
1. XX##: ((copy/paste the whole description + item code on the left from the skills/task list template))
2.
3. ...
4. ...
5. ...
6. ...
7.  (yes you can add,or change skills as you go, but make notes if you skipped or stopped a skill attempt)

___

# Section 2: Self Reflection and Evaluation
- Q: What tasks did you find the most challenging out of the ones you tried/listed? Why?
> A:

- Q: What tasks did you find the least challenging out of the ones you tried/listed? Why?
> A:

- Q: Any reasons behind your task/skill range selection? i.e. Did you want to explore, were you curious, or specialise? Or just wanted to collaborate with someone on some skill? (note: very open ended question)
> A:

- Q: Version control (VC) is one of the most core tools used in any software development industry (games included). Which did you use, and how did you go with using the VC system? Did you have previous experience with this software?
> A:

- Q: Did you or anyone in your team have previous experience with your chosen VCS previously? Did you have to do a lot of troubleshooting with others in the team with the VCS?
> A:

- Q: Would you consider switching VCSs in the future? Why? What do you think you learned well, and what else do you feel you should learn more about in the future?
> A:

- Q: Did you pick up any new skills or software during this assignment? How was that experience, and will you continue using the tools in the future, or not? Why?
> A:

- Q: Any other feedback/thoughts/reflections?
> A:
```

Available Skills/Task List

The text below is what you will find in Canvas. It is mainly here for reference.

 # Assignment 2.2 - Available Skills/Tasks to Choose from

___

 # 2D/3D Visual Artist Skills
VA01: Import 2D and 3D assets from an external program into a game engine, the asset must show in a level and be visible in the 'game world'. 2D Image must show transparency. 3D Asset must be unwrapped, textured, and also show transparency.

VA02: Import and work with sprite sheet animations into a game engine. Sprite sheet must loop in its animation, and demonstrate transparency. Also look into how to make sprites not loop and only play once. Spritesheet must be visible when the game is running.

VA03: Import a 3D animated object, and have its animation looped in the running game. Also look into how to make the animation a 'one shot' (plays only once).

VA04: Import a rigged and skinned/vertex weighted 3D object, and have its animation looped in the running game. Also look into how to make the animation a 'one shot' (plays only once).

*VA05: Animate using keyframes for 2D and 3D objects *within* the game engine itself (not using an external program). These animations need to be able to play in the running game. (Note that not all engines support an in-editor animator, so this option may not be available to you)*

VA06: Create one permanent/looping particle system, and one 'one shot' particle system, within the engine itself. The particles must demonstrate transparency, no shadows, and a custom texture/image, and motion. This needs to be visible in the running game.

VA07: Create a 3D 'arena' or room, using only the engine, without any external programs. Plugins or addons are allowed. There needs to be a minimum of 5 materials/textures created in the engine that are applied to the arena/room.

VA08: Create a UI using UI specific objects in the game engine. Some engines have "Canvas" objects or "UI" objects. Create a game user interface using these UI specific objects.

VA09: Create a 3D dark room, and demonstrate various different lighting objects. This needs to be visible in the running game. Explore ways to achieve volumetric light to see if your engine supports it. Also explore light emitting materials.

VA10: Explore 'Rigid Bodies' in the engine and build a structure that, when the game is played, has objects falling to gravity and show collisions and physics. Eg. A see saw/balance beam with objects of different sizes/masses falling on them at run time. Also demonstrate rigid bodies that are 'kinematic' (i.e. they do not move but things still collide with them.

VA11: Search for generative mesh plugins and create a scene/level with it. Eg. Forest generators, terrain generators, greeble generators. This must be a plugin for the game engine, and not something like a 3d modelling software like Blender/Maya/3dsMax.

___

 # Audio/Sound, and Music

AU01: Import sound files into the engine. Figure out a way to trigger one-shot sounds, and have looping sounds. You will also need to figure out how to trigger the sounds when the user interacts in the game with either a keypress or mouse click.

AU02: Explore the engine and/or find plugins to enable audio mixing and live-audio effects, such as reverb, delay, pitch shifting, EQ, and other effects. Find at least two effects that are working when the game is running.

AU03: Explore the engine and search for ways to get MIDI files to play in a running game. You may need to also search for plugins, and might need to learn about soundfonts as well.

AU04: Explore the engine and search for ways to stream audio into the running game (such as streaming from an online radio station). You may need to search for plugins to support this feature.

AU05: Explore the engine and search for whether VST or LV2 plugins can be loaded into the engine. You may need to search for plugins to support this feature.

AU06: Explore the engine and search for audio visualiser plugins, such as audio spectrometers, waveform visualisers, and mic level monitors, that can be visible in the running game.

 # Code and Programming

PR01: UI Programming. Set up a basic canvas, and hook into the button click events (with the mouse, but you're welcome to explore keyboard or gamepad events too).

PR02: Object manipulation. React to user inputs (such as mouse click or keyboard/gamepad) to do each of the following, with different events for:: A) Hide/show a game object. B) Create/instantiate a game object. C) Delete/destroy a game object.

PR03: Scene and settings: React to user inputs (such as mouse click or keyboard/gamepad) to do each of the following, with different events for: A) Switch screens/levels/scenes/worlds (and switch back). B) Exit the game, C) Change game settings such as volume, resolution/window size, fullscreen/windowed

PR03: Input handling. Most game engines have input managers. Instead of hard coding things like "if KeyPressed(Key.Space)", use the input manager for your event handling (i.e. 'if ButtonPressed("Jump")). You must be able to demonstrate how to get analog inputs (eg. X/Y/Z axis) as well as button inputs (ABXY etc.)

PR04: Physics programming: Manipulate and move an object/player character using forces in the rigid body system of the engine. Must be able to move around such as WASD, and also jump, and 'respawn'/reset position.

PR05: Game mechanics: Create or import a template project for a genre of your choice. You're welcome to ask your team what kind of genre if you're not fussed about what type of game to look into. (eg. First person, 2d platformer, 3d platformer, etc…) Explore and manipulate the template to demonstrate being able to customise the mechanics - this skill MUST be combined with other PR## skills of your choice.

PR06: Physics programming: React to a collision event and create an object at the collision location. A sound effect needs to also play when the collision happens.

PR07: Timers and delays: Make an object delete itself after a few seconds after being created. Implement this two ways: one with coroutines/threads/timers provided by the engine. The other way with a float variable that counts down based on time passing.

PR10: Game mechanics: You can discuss a mechanic with your tutor and come up with your own! Objects deleting when they have zero health, increasing a global score variable when certain conditions are met, etc…

PR11: Technical programmer: Assist a team member with setting up a project/task that involves working with a template/example project, and hook that project into one other programming.

 # Design and Testing

DT01: Create or import a game template, and manipulate the character's properties to simulate different 'player effects': A) 'slowed movement', B) 'doubled/halved jump height', C) Tweak cooldowns or animation speeds, if any.

DT02: Usability Configuration: A) remap controller or keyboard/mouse buttons to different actions, B) Change axis sensitivity for mouse looking C) Change axis sensitivity for gamepad inputs. Note that you may need different templates to fulfil mouse/gamepad sensitivities.

DT03: Build/export the game, so that it's able to run without the need for the game engine to run. Export and test for at least two different platforms that you can get access to yourself (eg. PC, Linux, Mac, Web, Android, iOS, etc..)

DT04: Testing and mechanics balancing: Explore and search for plugins for your engine that can emulate/simulate game inputs for automated scenario testing.

DT05: Testing and user experience: Explore and search for game-input loggers that shows or logs user inputs when the game is running. See if there are features to export the data to a file for analysis later.

 # Writing

WR01: Localisation: Explore how your engine handles localisation, and demonstrate at least two different localisation entries. You may need to search for plugins or templates to demonstrate this.

WR02: Dialogue system plugins: Explore and find dialogue system plugins, and try at least one template/example project.

WR03: With a project and dialogue system of your choice, create/implement at least one dialog tree or dialog variation. This needs to combine with another writer's skill.

WR04: Explore and search for plugins or templates for NPCs that 'bark' (or say dialogue over their heads, for example) periodically. Edit the dialogue that the NPCs say, and edit the name of the NPCs as well.