

GAME3023 Term Project

Progress Exercise DevLogs

Weight: 10% x (6 + 1) (60pts to course grade + 10 bonus)

Group size: Individual

Prerequisites:

1. Experience with 2D Unity game development
2. Experience using Git and GitHub in a collaborative Unity project
3. Familiarity with the associated 2D Pokemon-Like RPG Term Project assignment details.

Purpose:

1. Practice essential programming, game development skills through use of Unity to build a functional game prototype.
2. Learn about building scalable game systems.
3. Design core turn-based RPG-style game mechanics that create space for interesting gameplay dynamics mostly revolving around meaningful player decision-making rather than space and timing.
4. Develop systems/tools in a way that provides room for designers to create interesting experiences without needing much new code if any.
5. Use your own systems to create an interesting gameplay experience with meaningful decisions.
6. Document progress in vlogs, practicing skills in communicating design intent and functionality both as a programmer and as a game designer. You should be good at explaining what your game features are, how they work, and why players should care about it.

Task:

- Implement features from the list to follow
- For each one, create a simple **DevLog (Developer Blog/Vlog) video** briefly summarizing how the feature works. You may also discuss the process of development
- **Flexible target length of 2-8 minutes.**
- Upload the video and get a public or unlisted YouTube link, or a shared link to the video on another platform. Post it as a DevLog on an itch.io **page for your Term Project game.**
- **Submit the links** to the commit on your repository and link the DevLog page.

- If you feel it is needed to make the explanation clear, you may optionally submit a supplementary document (e.g. written report, the video script and/or annotated diagrams)
- **The work you do in these DevLogs can be directly used in the paired final assignment!** Both group members must do this work independently, implementing it themselves. Then, you can choose to add one or the other to the assignment project, combine ideas, or make it again from scratch. All DevLogs have A and B options, so you can coordinate with your partner to do one of each.

Grading criteria

Grades will be earned for:

1. participation in the activity and handing it in,
2. level of completion, and
3. quality of the completed work.

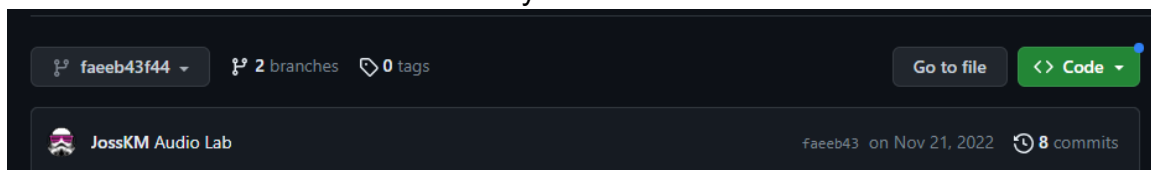
Grades will be **scaled based on**:

1. clear demonstration of understanding of the material you present, and
2. level of ambition, including attempts beyond the listed requirements.

You have some creative freedom to meet the requirements within a reasonable margin that also pays respect to your own game design plan. These objectives are to encourage a feasible timeline and show a history of work on your Term Project. On top of the grades earned in these Vlogs, you will receive additional grades for these features in the Term Project itself. For what to submit and what to discuss in your Devlogs, see the following breakdown:

Source Code - Mandatory

- Source code is submitted via GitHub which is the same as the one which is presented.
- Create a new BRANCH for each DevLog DevLog named “DevLog_X”. The TIP of each of these will be graded, or the direct commit link provided.
- **In Blackboard:** Provide a hyperlink the COMMIT representing the submission, not the whole repository. How to: (<https://vanderbergh.com/tips/&/tricks/2019/12/20/posts-Link-To-Specific-Commit-On-GitHub.md.html>) If you are not sure what this means exactly, **ask**. Failure to do this correctly can result in a 0.



- Optionally include the [commit hash](#) in addition to the direct commit hyperlink (but not in place of it)

Presentation - Mandatory

- Video has sufficient quality to have legible text, and clear audio

- If applicable, optional supplementary document(s) are legible and organized

Explanation - Mandatory

- Discuss the process of development.
 - How you studied
 - What you learned
 - What think should come next
- Summarize how the feature works in an easy-to-follow way
- Show relevant code and objects, and how they relate
- Demonstrate a higher level understanding of how various systems interact

Implementation

- Prove that the feature works properly
- Analyze and prove robustness and/or reveal bugs, flaws, weaknesses
- Describe how the entire feature works from a technical standpoint. Any portion not properly explained may be considered “not implemented”

Penalties

- Assignment will not be accepted without Mandatory components
- If the video extends beyond the target length or does not use the time well, significant portions of it will be ignored and the grade will be lower
- Instructor provided code is allowed, however using external or AI-generated code without reference and understanding could result in a mark of zero or further action

Workflow Checklist

1. From Master/Main branch, make a new branch DevLogelled DevLog_X (X=1,2,3) etc.
2. Complete and present ONE of the two options for each week
 - a. Make and push commits to the DevLog_1 Branch as you go.
 - b. It is recommended that you complete all tasks on time or ahead of time, as they include features required in your assignment.
 - c. Since DevLogs have two choices, it is recommended, *but not required* to coordinate with your assignment partner to have each person do a different one of them.
3. Record/update what helped you into a file named **Acknowledgements.txt** in the root directory of the repository. T
4. Record/update licenses for each asset used in **Licenses.txt** in the root directory of the repository
5. *IF you did the work first before branching, you can make the branch now and push to it, but it is recommended you branch FIRST so the DevLog branch shows your history of work on this DevLog.*
6. **Record** and upload your YouTube video.
7. **Start a submission**
8. Add link to the latest commit on the DevLog_1 branch to submission text. **Hyperlink** it so it is clickable.

9. Add link to YouTube video to submission text. **Hyperlink** it so it is clickable.
 - a. Ensure it is marked as Unlisted. If it is marked as Private (the default) it cannot be viewed.
 - b. Make sure you click the “Get shareable link” button. Do NOT copy-paste the URL from your browser. It will likely be a studio.youtube.com/ domain and will NOT be viewable.



10. SUBMIT the text



AFTER submitting

11. Merge the branch into your Master/Main branch so it has your latest changes
12. When it's time to work on another DevLog, go back to the latest in Master/Main and make a new branch (e.g. DevLog_2) from there.
13. Do not touch DevLog_1 branch again. It should stay that way for course submission historical/auditing reasons.

DevLog Tasks By Week Due:

Week DUE	Feature
1 - Sep 6	(Ungraded) Find your team and brainstorm your Term Project work in preparation for DevLog 1
2 - Sep 13	DevLog 1: <ol style="list-style-type: none"> A. Project Setup: Set up a GitHub repository and Organization for the project, and start a 2D URP Unity Project. Source and import <u>your own assets</u> into a new scene for your game, add a Player Character Sprite with movement. Make your own character animate based on movement in the Overworld. The character should change to show different states when moving and stopped, as well as showing movement in different directions. Describe how the C# scripting system in Unity works. Include the asset licenses for everything found, and ensure that you have assets which cover everything or almost everything in the game plan for DevLog 1-B OR B. Project Plan: Prepare a pitch for your Pokemon-Like RPG-style game including: <ol style="list-style-type: none"> a. the overall vision b. what specific features it will have, c. how it will meet the project requirements in its own way, d. any requests for alterations to the requirements to suit your vision e. Appoint a Project Lead and present a detailed project plan, including <ol style="list-style-type: none"> i. how the finished product will look with fake screenshots (mockups you make in Photoshop) ii. S.M.A.R.T tasks and timeline breakdown and work assignments for the whole game, using THIS Gantt Chart Template: https://docs.google.com/spreadsheets/d/16Yos3N1FnMeiAPoAU3wV4tmKe-wtHG N-mfwfaKi-Ccc/edit?usp=sharing. Be realistic, and pay attention to exams and major assignments!
3 - Sep 20	
4 - Sep 27	DevLog 2: <ol style="list-style-type: none"> A. Random Encounters: Make a ‘Random encounter’ battle scene that the player can enter <i>sometimes</i> while walking over ‘Tall Grass’ tiles (they do not need to be grass). The scene should show the player, an opponent, and a UI button which allows the user to Flee (leave). There does not need to the full battle code in the Encounter. The encounter rate should be based on movement, not time. In addition, players should have a minimum period of safety

	<p>after exiting an encounter before another encounter may occur. In the random encounter scene, add space for at least 4 different options for Abilities, which do not need to be implemented yet and can just print log messages.</p> <p>OR</p> <p>B. Overworld & Menu: Using Tilemaps, & Tile Palette, paint an Overworld Scene. This is the place players walk when not in an Encounter. Using URP and 2D lighting, add 2D lights. Using Tilemap Colliders, add collisions on tiles. Show the character walking through the scene and colliding with the world. Create the start Scene, with the names and student numbers of your team, and buttons: New Game, Continue, Credits, and Exit. "New Game" & "Continue" should lead to the Overworld scene, with the intent that in the future they will have different behaviors. Players should see the button change when their mouse hovers over. When they click, it responds visually and then changes scenes. The Exit button should quit the application, and when in Editor it should stop Play mode.</p>
5 - Oct 4	
6 - Oct 11	<p>DevLog 3:</p> <p>A. Ability System base: Upgrade the Random Encounter scene from your game and add multiple selectable Abilities to the menu. The abilities should be in a form that is usable by both players and enemies. The scene should implement a simple turn-based action order and a dummy opponent that passes the turn back to the player (no AI). Add 2 Abilities that the player can use to win the Encounter. When the player wins their first encounter, have a progression system to add a new, third Ability. The system should populate clickable buttons in the battle UI based on what Abilities the character currently owns.</p>  <p>OR</p> <p>B. Procedural Animation/Sequencing: Add procedural animations of some kind using Coroutines or other code that runs over time, that works procedurally based on code and does not require hand-authoring. Add:</p> <ol style="list-style-type: none"> Text boxes animate to write out information one character at a time Characters jump/move in different ways when acting/being acted upon Battle menu UI proceeds slowly to allow players to see the sequence of events (e.g. Player turn begin, player action, player action result dealing damage to enemy, enemy turn begin, enemy action, enemy action result dealing damage to player) 
7 - Oct 18	
8 - Oct 25	INTERSESSION WEEK
9 - Nov 1	<p>DevLog 4:</p> <p>A. Enemy AI: Add an Enemy AI to the Encounter scene that uses Abilities. It should choose in such a way that makes sense for the game. The AI must follow the same rules as the player in</p>

	<p>Encounters to seem fair. I.e. if the player must have enough mana to use a spell, the AI must also follow that rule. Its actions should be less predictable than exactly the same each turn, and more predictable than entirely random.</p> <p>OR</p> <p>B. Enemy/Encounter : Devise some process for developers to easily create new types of Enemies with different sprites/attributes/abilities etc and incorporate it into the Random Encounter system so that players may meet different types of Enemy in different locations. Consider the code architecture, using Nested Prefabs, Interfaces, and/or ScriptableObjects. Consider ways to design it that provide you, the developer, with options to design specific themed encounters, themed areas, tweak encounter rates in different zones, etc.</p>
10 - Nov 8	
11 - Nov 15	<p>DevLog 5:</p> <p>A. Items: Add Items to the Overworld which can be picked up outside of battle. Scatter some of these in the world. When you approach them, a popup should appear to show what button you need to press to pick it up. When picked up, it should show another prompt describing the item for a few seconds, which can be dismissed early with a button. In the future they may be integrated into the term project to be used to affect your Character(s) e.g. Vitality Potions which heal characters, or Pokeballs found scattered in the world in Pokemon games.</p>  <p>OR</p> <p>B. Inventory Management: Prototype the inclusion of a drag-and-drop Item-slot system in the game, similar to the kind used in Diablo 2 or Minecraft. You do not need to finish the system, just show the ability to drag and move objects on a grid.</p> 

12 - Nov 22	
13 - Nov 29	<p>DevLog 6:</p> <p>A. Implement a Save feature for the player's location. Click an on-screen save button to save. At any time later after stopping the game, if the player clicks "Continue" in the main menu, then start the game and automatically return the player to their last saved location with the correct state. If the player clicks "New Game" in the main menu, then start the game without loading. All of the game's state should be saved. If you have a Party of Characters, the state of each Party Member should be saved.</p> <p>OR</p> <p>B. Implement an Achievement/Trophy feature. Add an in-game Achievement screen you can open from a pause menu in the Overworld. The screen should show Locked and Unlocked Achievements, with descriptions of the requirements for each one. When the requirements for one are met, a popup prompt should appear for a few seconds to notify the player that they have unlocked that achievement. Add at least 2 unlockable Achievements:</p> <ol style="list-style-type: none"> 1. Take 10 steps in the Overworld 2. Open the Achievements Screen <div data-bbox="380 911 771 1050" data-label="Image"> </div> <div data-bbox="888 491 1466 953" data-label="Image"> </div> <p>(Ungraded) Term Project should be more than 60% complete</p> <ul style="list-style-type: none"> • For example, repository should show significant activity from both developers; • Game should already have progress <u>beyond</u> what was covered in the DevLogs; • Game scenes should have detail and not look like placeholders; • Special mechanics or extra features should appear at this stage in some form. Do not forget to do work outside of the DevLog requirements! • At this point, the feature set should be enough to support FUN in the game. If there are not, you need to push for them • Game should be play-testable
14 - Dec 6	<p>(Ungraded) Term Project should be at or nearing 100% functionally complete</p> <ul style="list-style-type: none"> • ...With room for balancing, and extra polish. • Extra features should be finished. • The Itch.io page should be fully set up, awaiting a final build • A draft build should be already up • A draft submission should be already made • Game should be in play-testing, level design, and/or balancing
15 - Dec 13	<p>DevLog 7 (Bonus):</p> <p>A. Audio Polish: Create a music system which fades in and out different tracks when you enter and exit an Encounter, or enter and exit different areas of the world. Add a footstep sound system that plays steps as the player character walks around the Overworld, and changes sounds when they walk over different types of ground. Should support <i>at least</i> two different sounds – 'dirt' and 'pavement', and ideally is easily extended to any number of sounds. Add sound effects to different Abilities used in the Encounter.</p> <p>OR</p> <p>B. Visual Polish: Add a full-screen effect to mask scene transitions from your Start, Overworld, and Battle scenes e.g. fade to black or wipe. Add particle effects to walking around in the</p>

	<p>Overworld. Add custom particle Effects to Abilities used in battle. Play a different effect for each Ability.</p> <p>Term Project Due</p>
--	---