Project Evaluation

Success Criteria Review

Criteria #	Criteria	Result
1	**Player Movement:** the player can use arrow keys to move about the world. There are animations for running in each direction, as well as idle animations for when the player is still. When an overlay or menu is present movement is disabled.	Success
2	**Pause/Settings Page:** the player can press "P" to open the pause menu and quit the game or return to main menu.	Success
3	**Dialogue System:** the player can initiate dialogue with an NPC. Dialogue can be different depending on the state of an attached quest.	Success
4	**Quest System:** the player can begin and finish multi-step quests of at least two types (item collection and combat). Quest-giver dialogue reflects state of quest. Player is rewarded in some way for finishing quest.	Success
5	**Inventory System:** player can collect items by colliding with them and have them added to a persistent inventory. Item quantity and description is tracked.	Success
6	**Quest Journal UI:** the player can open a quest journal UI by pressing "Q" to see a list of all quests, their states, and what steps they have completed for each.	Success
7	**Player Journal UI:** the player can open a character journal keeping track of character responses to story events by pressing "J", which is updated as certain story beats are hit.	Not implemented
8	**Inventory UI:** the player can open an inventory visual UI by pressing "I" and view icons representing the items they have in their inventory. By hovering over an item with the mouse or using arrow keys the player can read a description of the item.	Success
9	**Combat UI:** the player can see their health and the enemy's health during the fight. The player can type an answer into the response field and see feedback if it is correct via the enemy's health decreasing.	Success
10	**Combat System:** the player can initiate combat with an enemy through collision with it. By answering randomly selected calculus problems correctly the player can deal damage to the enemy. A timer is used to determine how much damage the player does.	Partial success- the timer idea was not implemented. Speed at which the player answers problems has no effect on damage delt.

11	**Animations and Music:** Enemies have movement, idle, and attack animations. NPCs have idle animations. Looping soundtracks for each room.	Not implemented. Only player has animations.
12	**Level Design:** the player can explore five fully fleshed out rooms with quests in each one, culminating in the collection of five of the calculus pebbles.	Partial Success - only one level was finished. Five was WAY too ambitious
13	**Story Beats:** the player can hit major story beats by completing certain quests, which trigger cutscenes/special dialogue. The player can "beat" the game by collecting some number of pebbles (or at least preview the end if not all 10 levels complete).	Partial Success - completing quests can trigger special dialogue and add items like pebbles to the inventory. Cutscenes were not implemented.

Program Effectiveness

The game is effective in that most of the systems I desired to implement are fully functional - movement, questing, inventory, dialogue, and basic combat features are all fully functional and work reliably, with most of the bugs ironed out. Player movement is smooth and disables/enables reliably as needed. Questing is very scalable and supports a wide variety of quests, actually far beyond the two (combat and collection) I had originally planned to implement. The inventory system is persistent and accessible for integrating with the quest and story systems.

The game is less effective in that the level design and playability of the game could still be greatly improved by finishing the rest of the levels so that the game has a complete arc and experience, adding more quests, and generally doing user quality of life improvements like UI re-designs and focusing more on playability and an enjoyable story.

Client Follow-Up Interview

[I met with my client on Monday, May 13 (a later date than intended because AP exams got in the way, apologies for the subsequent late submission)]

I show Kiersten the game demo, demonstrating the undertaking of a quest, the way the quest journal updates as you go, collection of items and population of inventory, defeating two enemies by solving calc problems, and finishing a quest and exploring the level. This takes a while, but it's mostly just me pointing out the different things as we went without any substantial feedback during the playthrough

Me: So, what do you think of the final version, of this project at least? I know it doesn't have all the levels and polish but what is your opinion on the basic gameplay, quests, and calculus integration?

Kiersten: I like it! It looks really cool, and the gameplay- like the movement and talking to characters and stuff- looks pretty nice and easy to understand. I noticed during the playing and wanted to ask- the calculus problems, are they themed? Like, are they integrals because the quest is related to Newton?

Me: Yeah, this level is the "Forest of Fundamental Calculus" so every problem in this level is integral related, because they're based of Unit 6 of calc.

Kiersten: Oooooh cool. Yeah, I like that, and I liked how the level is themed with the quest being for Newton. It's like calc *lore*.

Me: Do you have any feedback on the combat / calculus solving system?

Kiersten: Uhh, give me a second. Maybe adding something to make it feel more high stakes? Like add a penalty for answering problems incorrectly, by reducing player health and reducing it by more each time the problem is answered incorrectly.

Me: Oh, like how the points work in the stressful CTF problems!

Kiersten: Yeah!

Me: That's a good idea. The combat does feel pretty low-stakes right now. Do you have any suggestions for other systems? Quest? UI?

Kiersten: Maybe for the UI… I'm a big fan of having a little thing in the top of the screen that tells you what quest you're on and stuff, like a little reminder.

Me: Like a little circle with health bar, current quest name, player picture, and stuff?

Kiersten: Yeah exactly

Me: Oh man I thought about doing that because the Witcher UI reminded me of something like that but I totally forgot. That would be pretty easy to add.

Kiersten: Also for the UI, maybe you could redo the quest journal? I really like how the level is themed around the unit and the mathematicians like Newton and stuff, so maybe you could keep doing the theming with the UI? Like for a fundamental calculus have the UI be like a scroll and pen and then the more modern stuff be like a student notebook, if that makes sense?

Me: Probably beyond my current abilities in terms of UI scope, but that's a great idea! Maybe one day… Well, it's getting pretty late. Do you have any final thoughts?

Kiersten: Idk, I can tell its not a fully finished game like you said, but it made me happy. The idea is really cool. It was pretty math-tastic.

Me: *Math-tastic?!* Pff okay. That's awesome. Sorry I wasn't able to finish everything I pitched in our original interview, but I'm glad you liked it. Thank you! Bye!

Future Development Recommendations

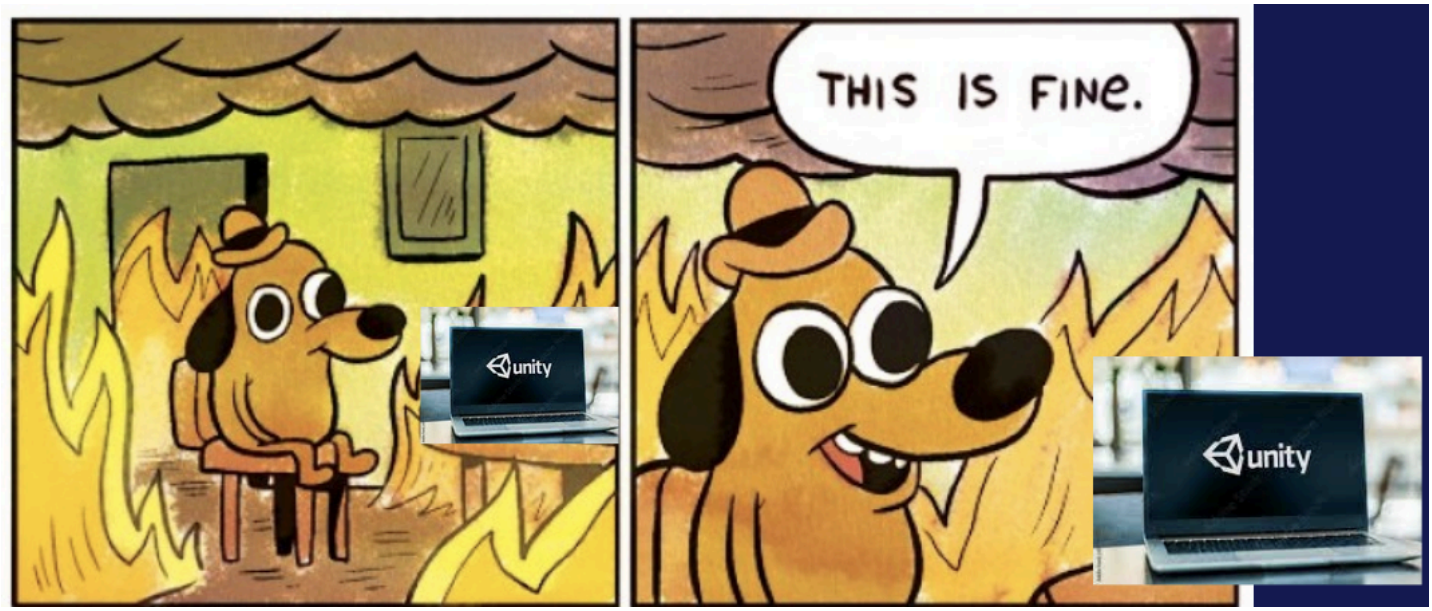
Minor Improvements:

1. **UI Adjustments:** download a unique/custom font to use in place of Unity's standard Arial equivalent. Redesign UI panels to have a pixel style look in the style of the other assets that is more pleasing and navigable for the user.
2. **Animations:** add idling animations for all NPCs and enemies, as well as fighting/hurt animations for enemies during combat.
3. **Movement:** add mapping for WASD keys if user prefers to use those in play of arrow keys.
4. **Quick UI:** add small circle UI at top of screen with player sprite in the circle, health bar underneath, and name of current quest under it.

Major Improvements:

1. **Save and Load:** use a fully fleshed out save and load file system instead of Unity's player prefs for serialization and persistence. This would allow a greater amount of data to be stored and retrieved more efficiently.
2. **Character Journal UI:** create a character "journal" that the user can read after certain story events happen in order to keep a log of those events and provide insight into the main character's perception of the story.
3. **Calculus/Combat Function Improvements:** revamp combat system to work with a wider variety of calculus problems including integration, better acceptance of exponents, and other higher math symbols. Consider investing in TEXdraw asset/library in order to render math symbols on screen (this is really what I would need to implement the vision I had in mind for this project, but it was a $60 investment- I might get it if I decide to fully finish this project). Create higher stacks either through use of a timer as originally planned or penalty for incorrect answers.

Development Process Review



^I am reusing this meme again because… it's accurate. I think I have melted my computer running this program and melted my mind as well.

At the end of my calculus class last year, I had to complete a final project incorporating calculus in some creative way. I decided to make a game! It was my very first experience with Unity and game development, and the grand vision I had of an educational calculus video game was realized only by a blocky character being able to walk around and kill little red slimes by solving derivatives correctly. But I enjoyed my brief first attempt at game development and resolved to resume the creation of "The Legend of Calculus" after I learned more about Unity.

Enter the second semester of this year- after spending a month learning more about game development and C# during our November/December unit, I was feeling pretty good about my improved skills and was ready to reinvent TLOC: start from scratch, with fully functional typical game systems like questing and inventory, and actually create a real, finished product! However, as I quickly learned, despite my improved Unity skills, I was, and probably still am, an absolute beginner at game development, and many of the things I desired to implement like events, serialization, quests and inventories built on scriptable objects, and complex adaptive UIs were *way* above my skill level.

I struggled a lot to stick to the timeline I had created during the planning phase, as systems I had charted to take only three or four class periods dragged into full weeks spent working not just in class but on gold days as well as I balanced trying to *learn* how to do things I had never done before, like managing events subscriptions or using lifecycle methods or creating player pref keys, with trying to *do* those things in my project. I let myself get thoroughly wrapped up in learning and creating the code of my game that I definitely neglected the actual game of my game, which shows in the failure to create five full levels as I fully intended (probably an overambitious goal to begin with even with my optimistic timeline, but alas), and what I would consider a sub-par user experience in terms of story and actual gameplay.

However, despite the fact that my game falls short of my original expectations, I am still quite proud of what I have managed to create. I learned SO much during these last few months. While diving into a project above my skill level might not have been the wisest choice in hindsight, it did challenge me to learn more advanced Unity concepts that I will benefit immensely from having pushed myself to learn: the Unity lifecycle, events, scriptable objects, persistence, abstract classes, the dreaded input system (shudder), C# dictionaries and lists, enumerations, animations, and much, much more. I know that the next time I go to make a game, at the very least the learning curve should be a bit shallower.

I am also quite proud of the logic and actual code underlying my game. While I relied heavily on tutorials and articles found online to help me with the creation of the questing system, everything created afterwards- the dialogue, inventory, story, combat, pause, and sub-quest steps - were created by myself. When I started, I never would have imagined being able to do that without regularly consulting YouTube and StackOverflow, but by the end of development, I was coding new features in a single afternoon, finally able to use what I had learned without relying on outside help as a crutch. The systems I created might not be the most elegant or efficient as they could be, but to my bouts of testing they have proved (mostly) bug free, reliable, and scalable with my game.

If I could rewind time and restart development, I would massively scale back what I wanted to create, giving myself time to learn what I needed to learn so that I didn't feel like I was falling way behind once I moved past basic movement/UI/scene management and starting creating the more advanced systems. Instead of trying to complete five full levels, I'd probably commit to only two, giving myself time to really focus on creating quality code and levels instead of fretting about if I was going to be able to complete everything. Despite the challenge, I'd still try to learn and implement the features I did, as I feel that learning the new skills that I did and pushing myself is what made this project experience valuable and those skills are ones that I will use in the future.

All in all, I find myself in a position strangely similar to the one I found myself in last year- having bitten off far more than I could chew, learning a lot in the chaos, displaying my calculus inspired RPG to friends and family (able to show off A LOT more features this time), but still feeling like there is much more to do to finish the project. I really would like to actually finish this game, which probably means this summer I'll be sitting down once more, ready to start TLOC v3.0. I guess I really have entered the development *cycle*, which while frustrating, gives me hope that as each iteration improves, eventually I will have a fully polished, finished project, that can one day be published for real and hopefully entertain a couple calculus nerds.