EZRPG Post-Mortem

Successes

What went well?

- The energy of everyone working at the beginning was good, but it fell quickly after week 4
- Lucas thinks his Settings menu looks pretty
- The separation of everyone's "isolated mechanics" via folders and/or assembly definitions was good.
    - It forced those isolated mechanics to be truly isolated
    - The branch separation was good too.
- WE STILL HIT THE DEADLINE RRAAAAHHHHHH
    - And completed most of what was outlined in the GDD + more!

Failures

What went wrong?

- detailed world building is much much more tedious than anticipated
- Unity's Terrain Details don't support COLLIDERS. **AT ALL.**
- Kenney.nl assets are… "you get what you pay for" (which they are free)
    - this one in particular has horrid UV maps, textures are missing in some spots, and the alignment of objects causes Z-fighting a lot.
- Using the term "Things" is a horrible way to plan implementing Mechanics, don't do that in a GDD
- The boss fight was definitely under-thought (in theory) in the MVP
    - Boundaries of fight?
    - Player escaping?
    - Stages of fight?
- Using just docs to manage who's on what is somewhat bad.
- Didn't account for time taken to gather what assets were used and setting up the credits area of those assets.
- Didn't account for making a debug menu or test menu for all aspects of the game.

    - Though in my opinion, Test Driven Development (TDD) is dead for game dev due to there being too much user modification to TDD everything in game design - it would be worthwhile to save future time by building "Debug Menu"s or cheat tools for devs to use. We didn't account for making those in the GDD, and it caused dev time to suffer.

Conclusions

What should be done in the future?

- For a large scale premium 3D game, use premium assets (duh).
    - This should be self explanatory, but either make assets yourself or use premium $$$ ones. Free assets tend to have deeply rooted issues that are hard to fix / not worth fixing.
- Don't use Unity Terrain's Details and/or Grass for complex objects, or interactables at all.
    - The <u>Details</u> tools should be for ***purely visual*** things that a player can pass-through, like grass, ferns, bushes, etc. The <u>Trees</u> tools seem… okay, because they support *primitive colliders*. But I bet 99% of the time that you'll end up wanting to scrap using Unity's Terrain tools for anything mesh-related. E.g.: you want the trees choppable? Time to place them manually in the scene hierarchy buddy.
        - Seriously, if you attach a script to the prefab of the used trees in terrain, Unity will strip the component off of the tree when it uses it for the terrain.
    - *However,* it seems great for prototyping visuals or small scopes. I'm sure a solo indie dev who wants some wavy windy grass would adore Unity's Grass in the Details section once they tweak it enough. And it seems relatively cheap on performance out-of-the-box since you can GPU instance it all.
- Use the word "Features" instead of "Things"? To namely define where mechanics get glued with Assets to form something game-specific. Features seem to be wrap-all for game-specifics, whereas Things is still too general for game-specifics.
- Reflection Probes in Unity can be very costly to performance if they're refreshed often (or every frame). "Baking" reflections on awake seems to be the way to go, or using shader hacks.
- A Trello or Kanbo board *would* be more decent for tasking, rather than just docs.

    - Perhaps try out https://www.codecks.io/ sometime.
- This is irrelevant to any previous bullets on this document, but I think using Obsidian (perhaps with git) for documentation is better. Google Docs (which is what was used) can be a pain to audit, and it has no dark mode!!!