Ideas for variable minigames

Fix the scene

- More of a final situation type game, where the map gets altered based on what they choose
- My original headcannon has a player character moving through the map and things happen based on their choices – but that might require interesting things like animation and stuff
- If its more of that flat panel map, then our buddy characters should, at least, maybe have facial expressions that match a "win"
- Instead, then, we can have a user satisfy the requirements of each map, and must successfully "run" a program on each map to continue forward.
- 3-5 maps to complete the game
- Maps:
    - Getting dressed – answer a prompt about the number on the shirt, and the shirt color
        - He should get a prompt to decide what he wants to wear
        - Look into mirror and see shirt, if it isn't correct (which should be forcibly failed the first time), he needs to go to the closet to change it
        - The first map solution will be *interrupted,* because the closet is another map to be solved.
        - Once "organizing his closet" map has been satisfied, he can return to the mirror and simply change his shirts variables to the desired outcome. This isn't meant to be hard, or even that fun, but it almost helps pave the way to show functions (kind of).
        - The organizing map is described below, and I have some thoughts to make it a minigame that can be revisted. If we can implement the randomness of map generation, we should prompt users to "check in with mom" which may trigger another Closet event.
            - This does not need to be forced, they do not need to check with mom and reset the pattern required of the closet map being satisfied.
    - Organizing his closet first – pattern matching
        - Have him remember that his mother decided on a pattern for colors one time, or maybe for numbers the next.
        - Have the different scenarios possibly randomly choose everytime they "enter" the closet
            - This will require randomly pulling a scene that has him "thinking about what his mother said" with her pointing at a series of shirts that repeat a pattern at least twice. A pattern may be limited to 3-in-a-row of any given color or number, but can be also 2-in-a-row, or all one color, then all the next color, or 3, or maybe *they all have to be the same color*
                - Note "at least twice" means that we'd need to illustrate up to 6 shirts in this screen.
            - Coding wise, this requires a screen that prompts the mother, then panes over to the pattern schematic that its randomly selected
                - This schematic will then set the goal/win for the next frame
            - This screen drops and returns to the closet

        - Closet might also randomly generate its blank state, coding restrictions might require that the blank state is randomly generated without any weight from the "win"
    - Closet must be coded to "win" based on the chosen scenario
        - For example: if the win schematic was R-G-B x 3, then 9 shirts would be available (not fully fronted, we'd need to create assets that are as if shirts are hung up), and the program would wait for a user to change all 9, hit "run" and see if they succeeded
  - The shirts available are interspersed with ones that are already colored/numbered, and those that are "blank"
    - They can assign the unassigned ones
    - Or CHANGE the assignment of others (brute force methods anyone?)
- Choosing a backpack – color and size(capacity)
  - Needs to have x many things because the teacher said so
  - Needs to match his shirt, or needs to be a different color than his shirt, or matching his best friend's backpack
    - If we reference a "friends backpack", we would have a similar mechanic of our buddy envisioning it in his mind, much like what was described for the closet organization.
    - He would envision the backpack, and it would have a color, of course, but also a descriptive text that says packColor = "----". This will help with recognizing the link between writing in the assignment and seeing it resulted. Normally I'd say don't have extra text, but this text doesn't necessarily need to be read to accomplish its goal, and stands to help visually access the idea of "writing in" a variable's value.
- Loading up the backpack – adding items to its capacity
  - Luke needs to check his backpack's capacity first, then choose from *n* items – apples, most likely. Maybe pencils too.
  - Again, pose a puzzle – his teacher really likes green apples, but his friend wants a red one. Luke wants to give at least one apple to each. COOLMATHGAMES™
  - Load in apples that don't have a color, or only one does, and play a pattern matching meets word problems game
    - Luke needs to choose how many he can put in the backpack, since it has a capacity, then load in *n* many apples that do not have a color
    - The backpack will update its numApples variable, then
    - User then assigns colors each apples appleColor variable to match the problem statement, and hits RUN
  - We could spice it up even from here, and then have him remember that he needs pencils, *x* many to be precise. Does his backpack have enough room to add these pencils?
    - If the user added way too many apples, it won't. They would need to reduce the number of apples. Give an option to update the numApples variable, *then* prompt them to check the colors.

        - This might become very confusing for a younger audience if we do not introduce this very step-by-step.
        - The problem-statement about *x* many red and *y* many green should be a screen that is easy to reproduce and have it make sense, it would be ideal to have that problem statement reintroduced exactly the same, but since we've already solved that problem, we don't need any extra information or setup to it – just the final prompt.
        - Users can go into the apples directly and change their color.
    - Now that there is enough room, time to add pencils
        - Again, they can have a color attribute, or maybe a type – pen vs pencil vs crayon. Or a name "my favorite pencil" "my friends pencil" "a pencil I found" "colored pencil" – this might help introduce how strings can be values. Sure, we've introduced colors, but those have a visual representation, these identities aren't immediately obvious, and need to be "said"
        - Same setup as adding apples, at this point its simply a rehash, but they need to be thinking about the numPencils attribute of their backpack – picking a number of pencils, and then selecting which ones based on some imagined prompting
            - To avoid having too much text, since the variables themselves will be strings, we could have an envisioning-thought-bubble that simply *shows* the pencils he wants to take.
            - For this to work, we'd need a set of pre-determined sprites that have values already assigned. This might be a case where we do not allow the values to be mutable, they simply have to solve the prompt.
            - The thought-bubble would show the 2-3 sprites that he wants to bring, and perhaps have them interactive by forcing the user to panel through them and see their descriptions in the bottom screen, to understand the string concept.
        - Coding wise, we must be careful in this scenario – the required number of pencils must be strictly gated between a low number, like 1 – 3. And the randomly selected amount will likely be done by the game itself, which then affects how many sprites are loaded into the thought-bubble. (I need a name for the envision thought-bubble thing).
        - User adds pencils in a drag-and-drop style. There was discussion about variables and values being introduced in the tutorial like a drag-and-drop. If that's implemented, we should use it here.
- The backpack loading map is pretty complicated, and is actually more like 2 maps to be solved. I imagine that the outer scene will focus in on the two inner scenes – apples and pencils, which both require a user "run" their solution on them. The outer scene may or may not have its own puzzle – need some ideas to polish it off.

- The goal: get Luke ready to leave and go to school. Maybe that's the final outer scene idea listed above. Prompt the user "Is luke ready?" and maybe as a big finale, they hit "yes", then "run" and the game completes.
    - I wouldn't mind some magical coding prowess here with *some* kind of animation, something like him celebrating, walking out the door, or some combination of those ideas.
    - Its likely we'd need to directly modify sprites, or find another set of assets that has different poses/structures that represent our human
    - Note: if we change the character representation, the shirts will also have to change.
- For the final run, lets not make it like a real program and potentially crash the shit out of it, they hit yes and that's it, no backchecking that all the variables are correctly set. Like, unless you really want to.
    - I mean.. there is a point to that. We could illustrate the program running by showing the "code" that a main function would execute in the top screen after they hit run.
    - It wouldn't be interactive, and it might be confusing, but its kind of cool.