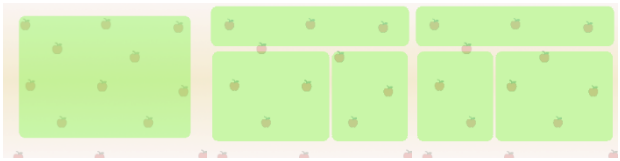
MELODY MAKER

Assets:

Top bgs:

Bottom bgs:



Objects:

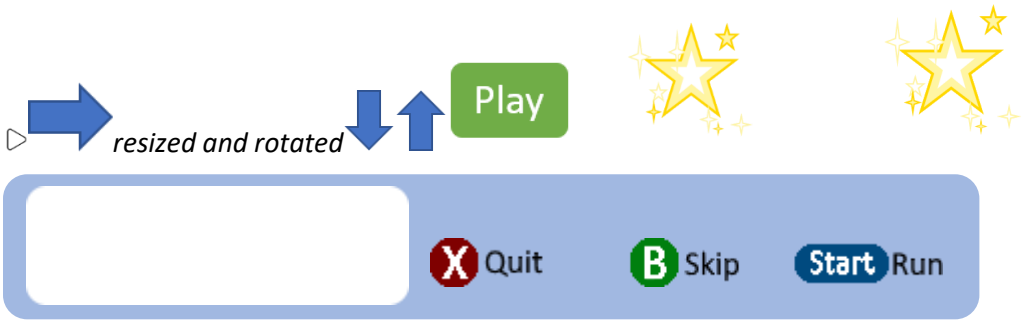
Note boxes

Notes

This asset might be unnecessary if we use the code Ernesto utilized for highlighting currently selected images, less overhead in this case but I'll let you decide.

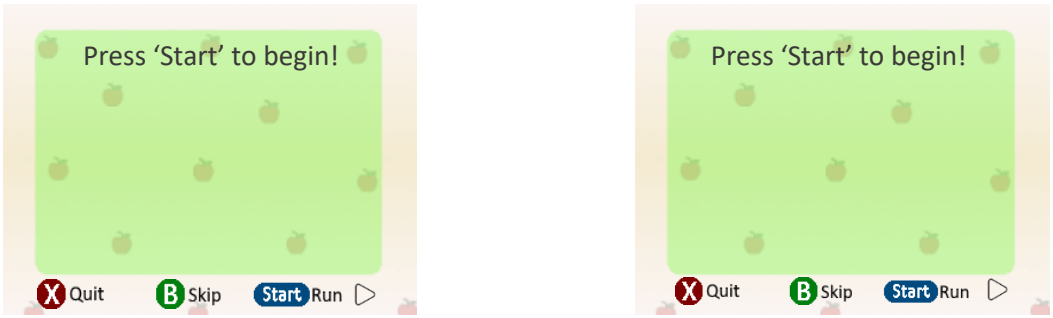
Compy

Other



LINKS OF CONTENTS

PANEL 1: TOP.1 <-->TOP.2, BOTTOM.1

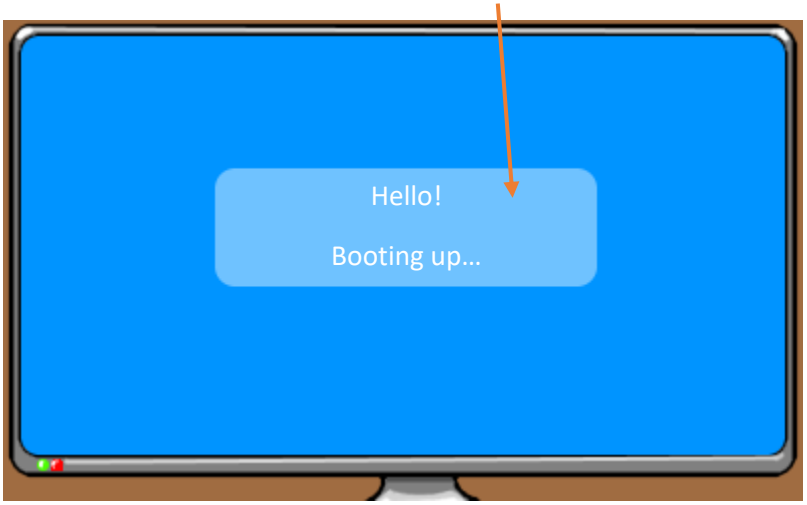


Intro top bg screen oscillates between the two for "sparkling" effect, as does 'compy's' eyes – to simulate blinking. If this looks really strange, we'll take out the blinking part, or add more in to smoothen the animation.

Compy is a separate asset laid over the backgrounds, here.

White text box is same asset from variables tutorials (small text box), opacity is even more reduced here than in prior game.

PANEL 2: TOP.3, BOTTOM.1

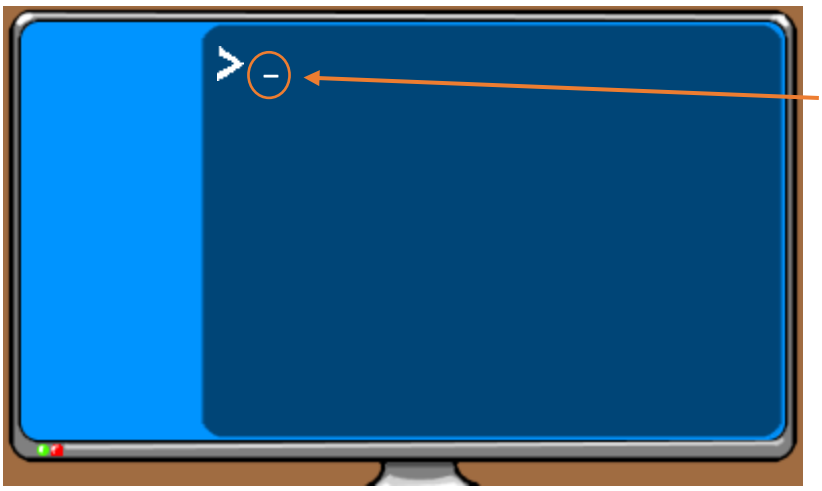


I wanted to simulate a computer to really integrate the idea of programming at this point. Stretch goal is to have the three dots in the ellipse ('. . .') appear one at a time for 1-2 timed cycles (1-1.5s in between, total 6-9s)

*** Even bigger stretch goal: have the blue screen appear first, then the white text box, then the text: "Hello! Booting up." -> "Hello! Booting up.." -> "Hello! Booting up…"*

Note: can text color be changed to white? If not, the color scheme of the upcoming assets will need to be tweaked.

PANEL 3.0 (OPTIONAL, STRETCH GOAL): TOP.4; BOTTOM.1



This is a stretch goal panel:

Blinking input-cursor "_" and then building the text:

"run" → "run H" → "run He" → etc until it reads "run Hello.exe" (can be very fast)

As you would imagine a cliché and old-fashioned way of representing terminal prompts.

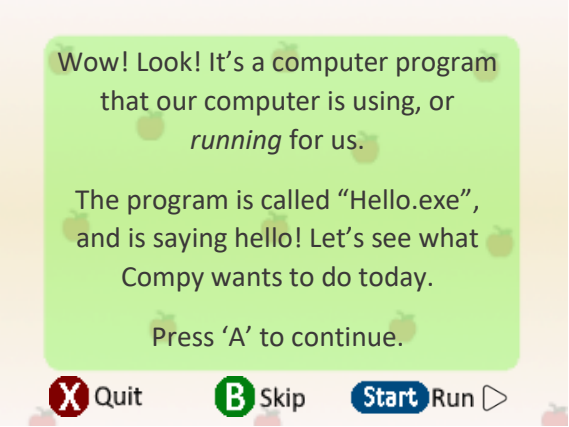
Once the string "run Hello.exe" has completed on the screen, it can populate the next screen below.

The bottom screen, at this stage, would be blank (BOTTOM.1), transition would be timed and not triggered by user input.

PANEL 3.1: TOP.4; BOTTOM.1

First time running this game: Users must go through the entire series of panels, up to PANEL16.

*After "tutorial" is complete, this screen should be replaced with PANEL 3.**2***



The "quit_skip_start" asset should really just be loaded at all times, since different stages of the game have use those differently.

If we decide we have the time to go through and fine tune when things are available, then we can separate the buttons and adjust opacity per PANEL set.

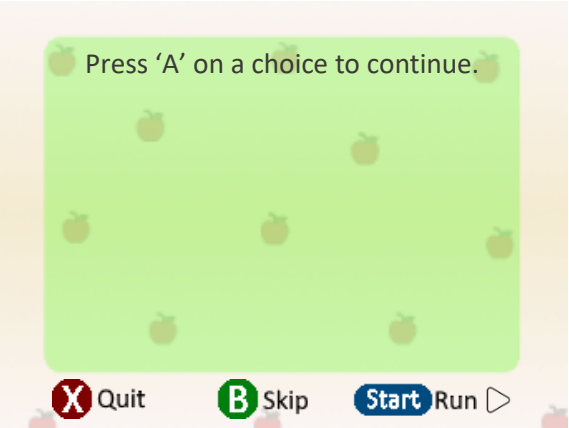
PANEL 3.2: TOP.4; BOTTOM.1

Selection of the first option will run through all screens until PANEL16.

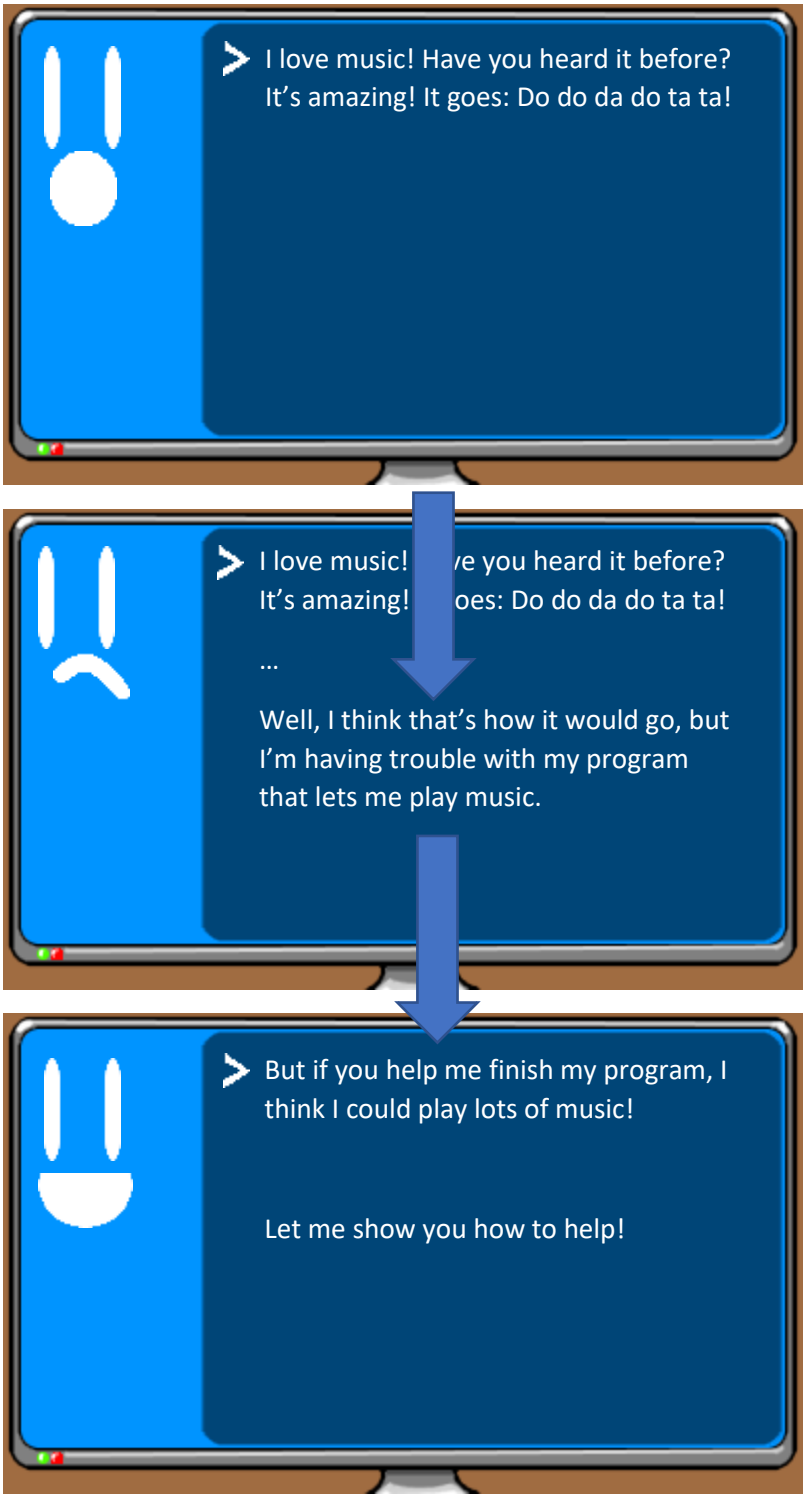
Selection of second option will jump right to PANEL16

If this branching is too hard right off the bad, we can instead turn on the skip immediately, even if users haven't "saved" a tutorial win yet.

In this case, Panels will run sequentially, (instead of replacement).



PANEL 4: TOP.4; BOTTOM.1



The text is a little silly, but I actually had my kids read/test different dialogues, and this one got a giggle out of them – so it won.

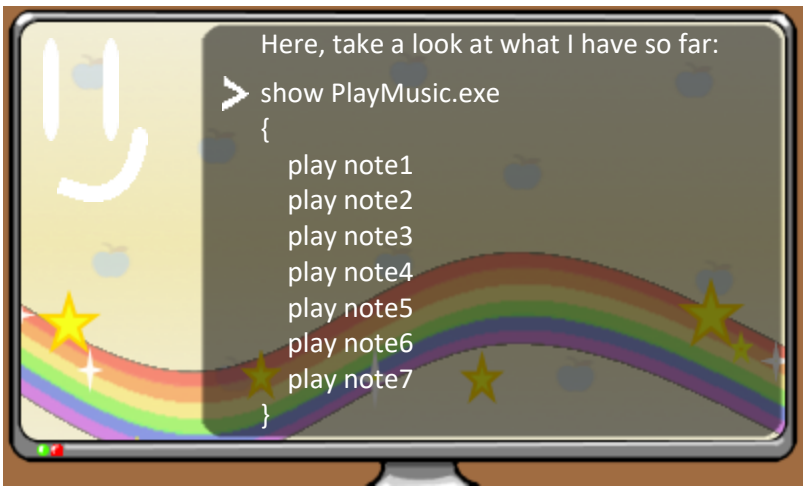
Top screens build text as the face changes expressions.

Can have text on screen update with each press of 'A' so that slower-reading learners have a chance to get through the dialogue and have the same experience as everyone.

The last top screen here has finally replaced the starting strings.

Also note: apparently this is their favorite face that Compy makes for whatever that's worth.

PANEL 5: TOP.5, BOTTOM.1

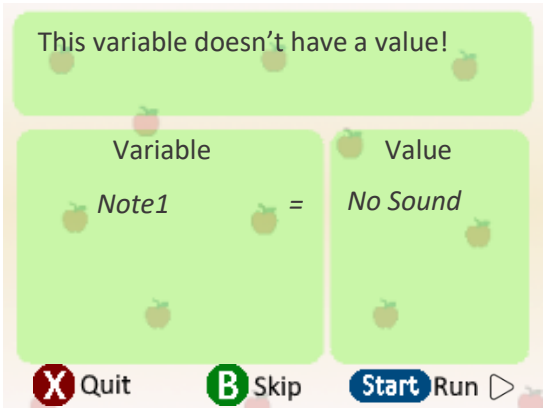


Getting all 7 “play note” strings was bit of a stretch formatting-wise, and this is in word. I’d like to offer up to 7 notes per song in the game since the spacing in the active screen allows for it. If we can get a little creative with fitting this text into the box on live as well, that would be ideal.

STRETCH GOAL: just like the stretch goal PANEL3, having “_” blink, then “show” → “show P” → “show Pl” → … → “show PlayMusic.exe” on a timed animation of the string growing is a polishing goal.

*In that brief testing I did: the sequential nature of commands did *not* need to be explained. The kids picked up, intuitively, that the notes were played one by one simply by the naming convention and the way its written.*

PANEL 6: TOP.5; BOTTOM.2

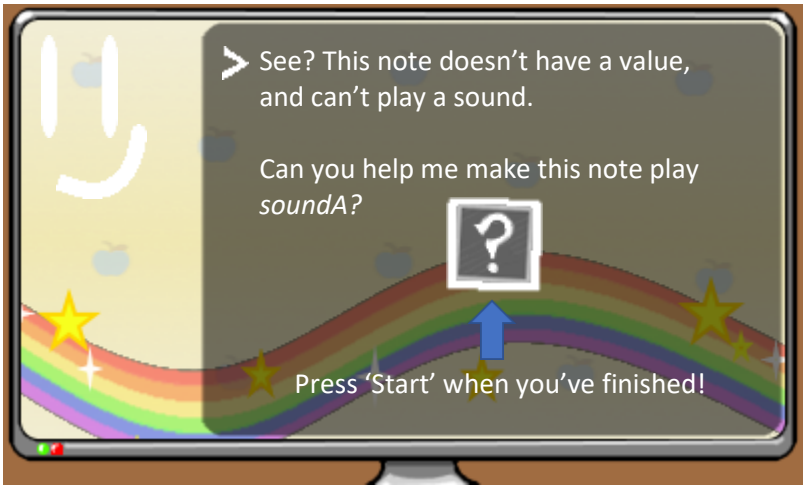


The empty_note asset here is in front of the highlight_note asset, since the gray on black look was kind of hard to see.

The highlight_note image is meant to go behind these box assets in the active play to help indicate (along with the arrow) which note is being given a value (sound).

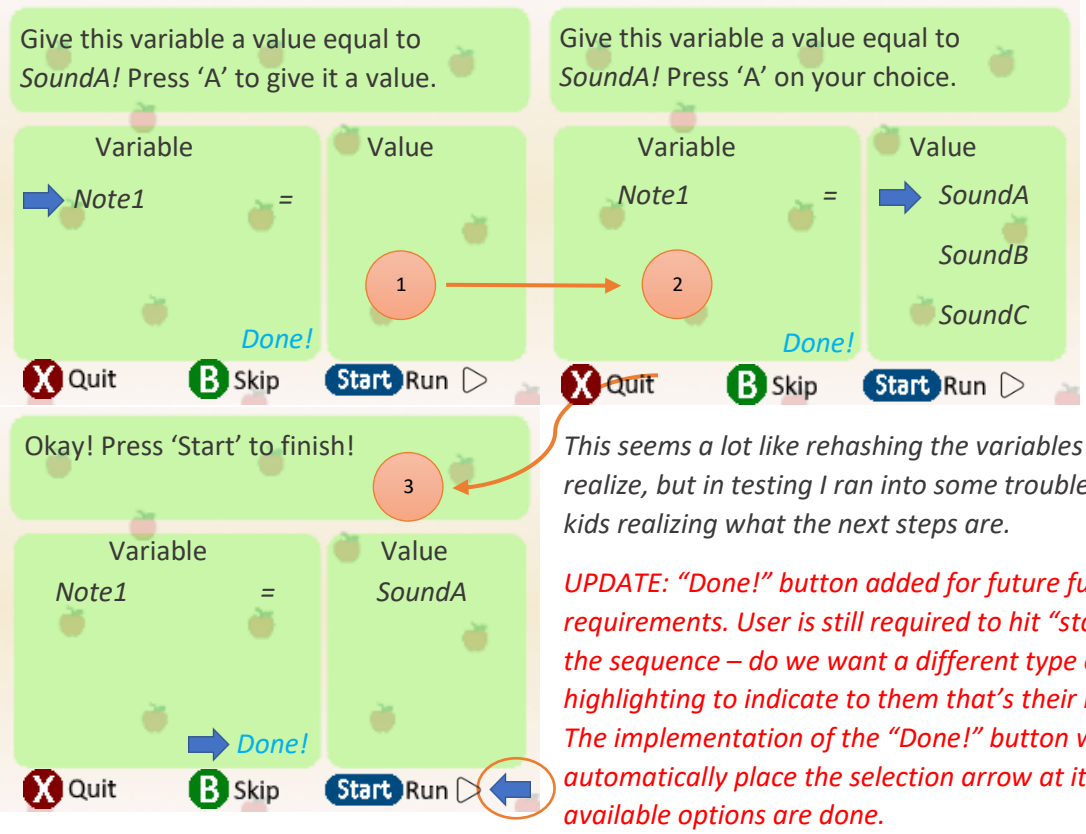
Its not much of an outline, but I didn't want to take up too much real estate since getting 7 notes in has a pretty precise look to it.

PANEL 7: TOP.5; BOTTOM.2



Below is a sequence of textual population updates that the bottom screen should cycle through on each press of 'A'.

The nature of leading them through it, and having the screens populate based on their button pressing is necessary feedback to confirm the right actions.



This seems a lot like rehashing the variables tutorial, I realize, but in testing I ran into some trouble with the kids realizing what the next steps are.

UPDATE: "Done!" button added for future functionality requirements. User is still required to hit "start" to run the sequence – do we want a different type of highlighting to indicate to them that's their next step? The implementation of the "Done!" button will automatically place the selection arrow at it once available options are done.

PANEL 8.0: TOP.5; BOTTOM.2 (keep prior bottom screen)

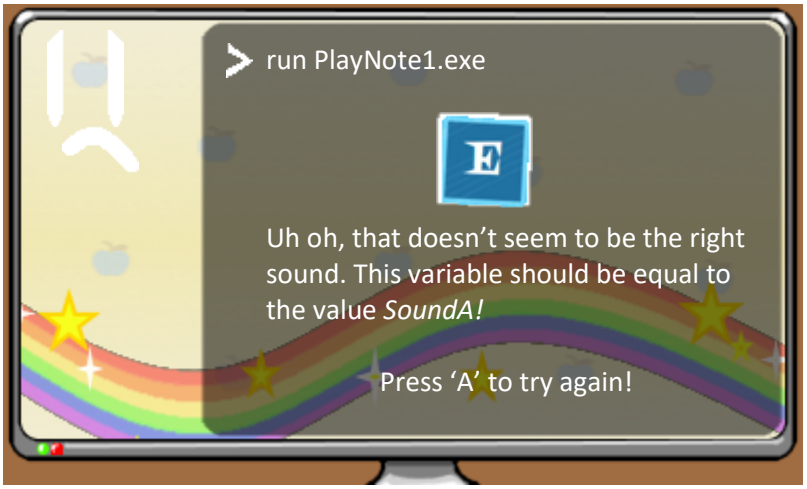
Ultimately: my goal is to have the note actually play after a brief pause, so that "run PlayNote1.exe" can be written to the screen.

Timing for the active play is important, so I hope this is a feasible request.

After the note has played, we have two options:

1. *Give player option to press 'Start' to hear the note again (simulating re-running the program)*
2. *Automatically moving to next screen with no input or control from player.*

PANEL 8.1 *ERROR WRONG NOTE:* TOP.5; BOTTOM.2 (keep prior bottom screen)

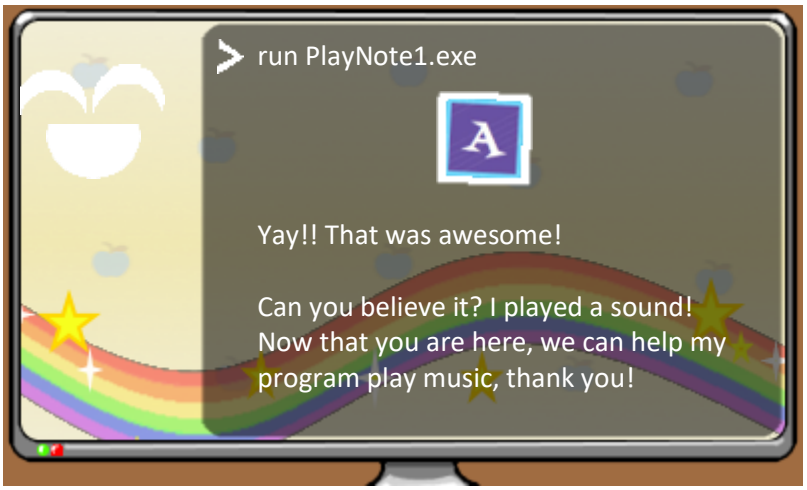


The bottom screens for both PANEL8.0 and PANEL8.1 could simply be the bottom screen in P7 with the latest populated data for that variable.

Since user input is restricted to 'Start' in P8.0 at the worst, the error screen would enable 'A' now as an option, and loop all the way back to PANEL7.

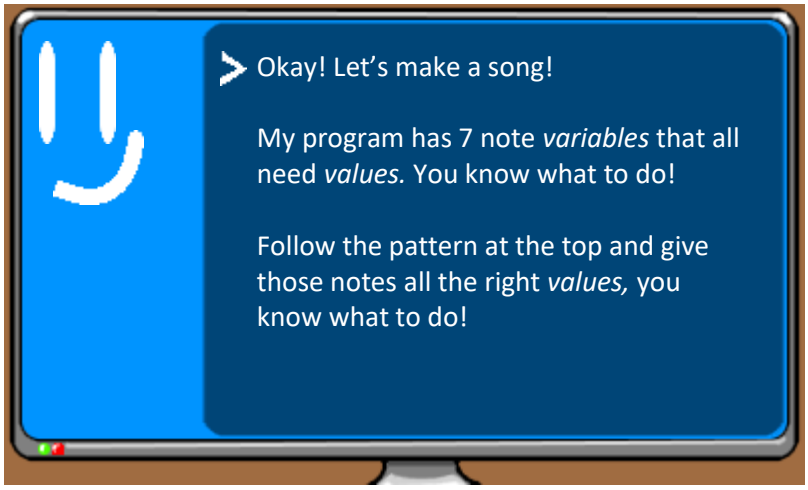
The return to P7 is not 100% logical, dialogue-wise, but there is so much going on with this game that I didn't want to add another set of text with only minimal difference. We can revisit this for polishing if needed.

PANEL 8.2 *SUCCESS:* TOP.5; BOTTOM.1



PANEL 8.2 ***does*** *have an updated bottom screen.*

PANEL 9.1: TOP.4; BOTTOM.1



Finally, no more long expositions on Compy's part – time to actually start the game.

As of right now, the game should immediately launch into the active gameplay screen (no selection on a users part) with a pre-concluded song pattern – likely just a scale or a 3-note repeat (I'll figure it out).

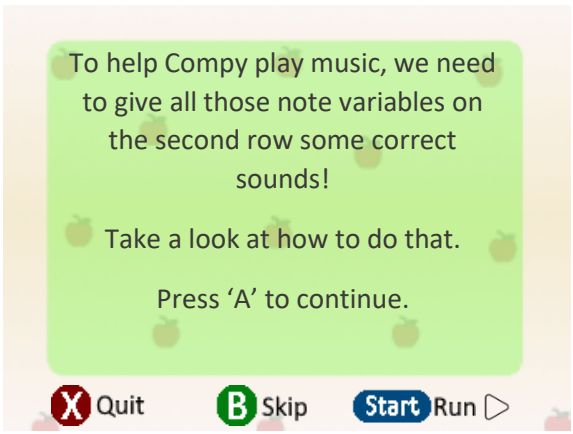
Bottom screen for PANEL9.1 is probably better left without text, then populate with the given dialogue at 9.2.

PANEL 9.2: TOP.4; BOTTOM.1

PANEL 10: TOP.6; BOTTOM.1

We talked about the overlapping containers that split the screen, but un-assigned notes shouldn't have a scalar, now that I think about it. Having them flat across the bottom is a nice solution for now.

The bottom screen is a loader type screen, we probably won't keep it in once users are selecting their own songs to write.

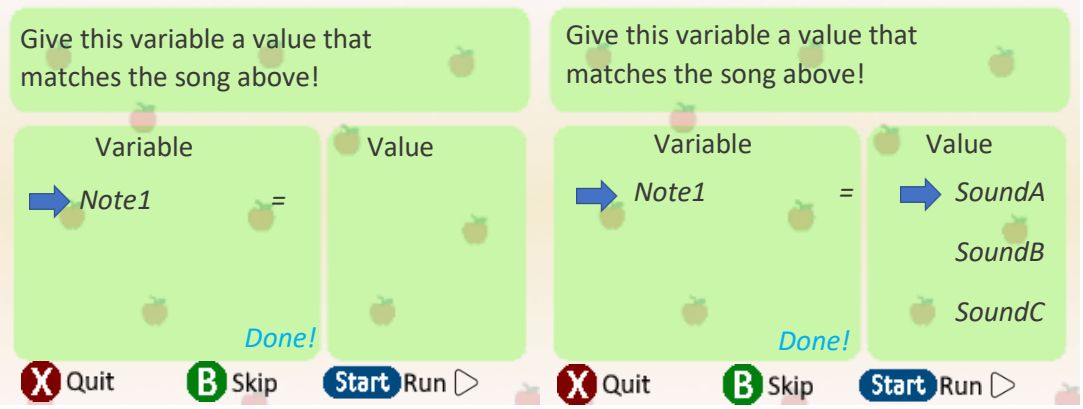


PANEL 11: TOP.6; BOTTOM.2

Not displayed: highlighting the "?" note that the cursor is hovering over.

The associated bottom screen is meant to be generic for this and future active gameplay, so you can save it as a template or however it's likely to be stored.

Note: the "start"/"run" button is available right from the get go. I'm still worried that younger users won't know to hit "start" at the end when they're done, though.



UPDATE: Added "Done!" button functionality. See P22 for explanation.

We have a choice here.

When a user selects what they want the empty variable to be equal to, do we:

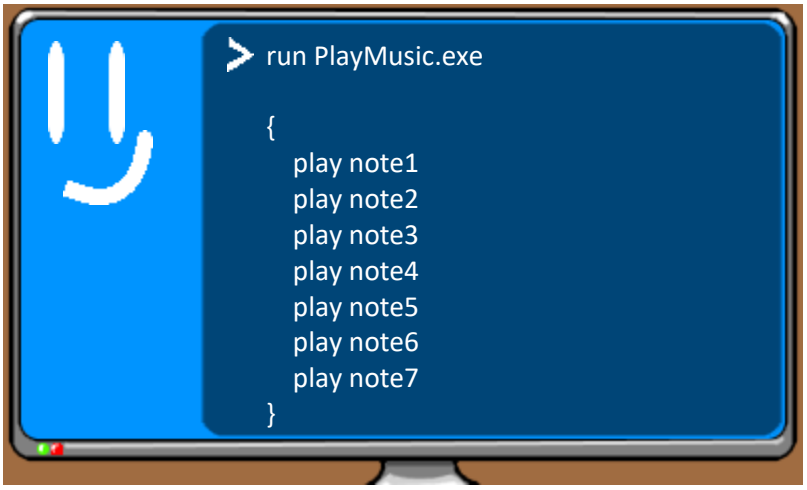
1. *Populate the asset in its correct x,y coordinate on the scale*
2. *Simply replace the "?" asset, in its current position, with the new note asset?*

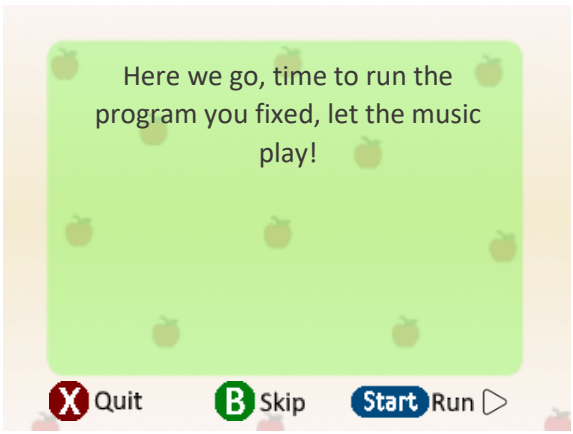
Its probably six of one, half a dozen of the other, but we can debate it a little. My thoughts currently err to #2, an incorrect input might result in asset overlap.

Note: The colored asset that replaces the empty "?" should match what the user actually chose, even if it doesn't match the pattern.

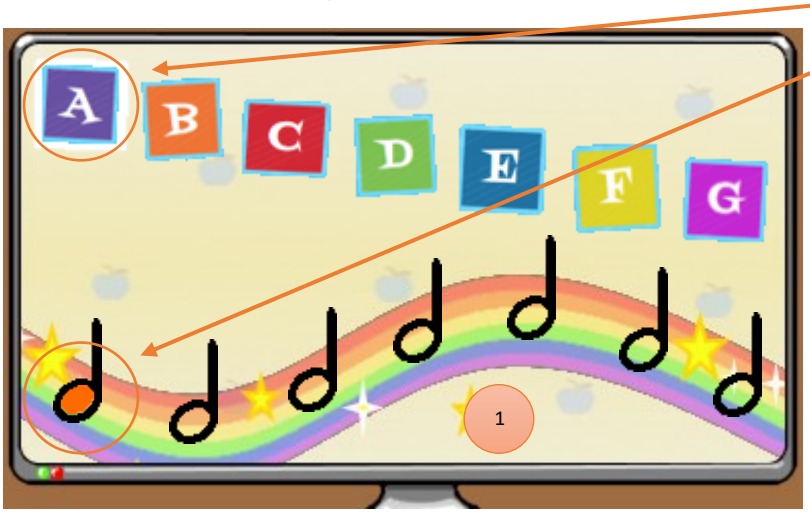
NOTE: HIGHLIGHTING behind selected notes is missing!

PANEL 12: TOP.4; BOTTOM.1





PANEL 13: TOP.5 <-->TOP.6; BOTTOM.1



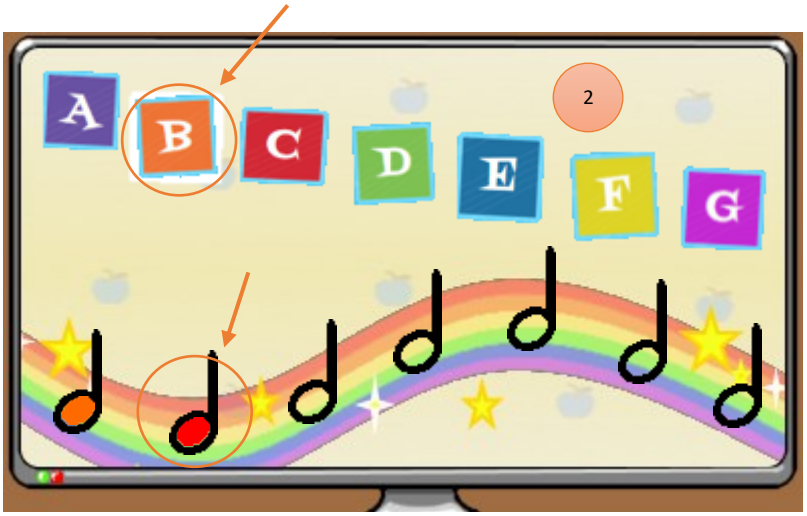
Goal: for each note being played

a) Highlight the note in top margin
b) Fill in the center of the empty quarter note skeletons
c) Oscillate the background with TOP.5 & TOP.6

The quarter notes along the rainbow do not need to be matched in any way. They can even just be in a pre-defined order from a list that cycles. There are 5 colors, so with 7 notes you'll have overlap anyway.

The fill-ins for the quarter skeletons are individual assets that are fitted, so they can be overlayed with the skeletons without any worries about clipping.

My thoughts are: the skeletons are an array of x,y positions that place a note per, and never change, while the fillins are cycled through another predefined set of x,y coords associated with a number that calls the asset.

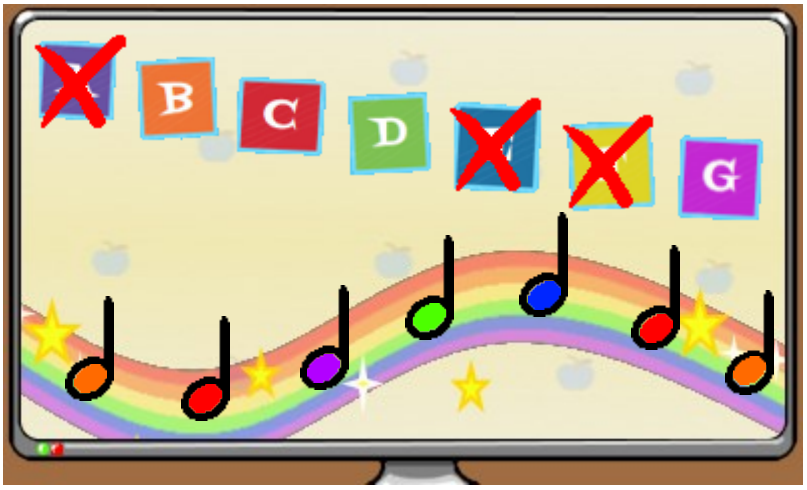


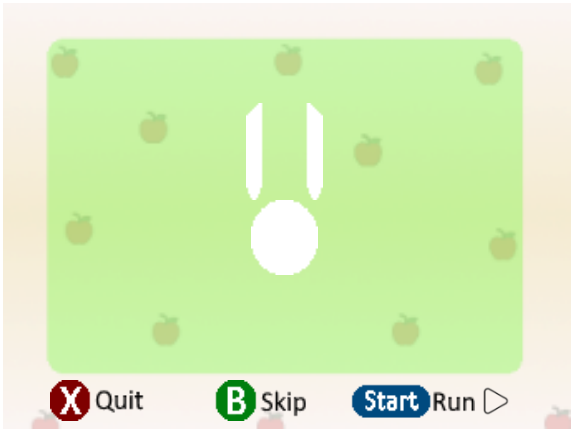
For the duration, user input is not accepted. For the bottom screen, I've had it so compy is bouncing up and down with a different mouth asset to simulate him dancing, lol. This can have an independent timer, or be on the same one as the top screen – whichever is easier.

Stretch goal: have Compy perform a randomly selected dance per each song a user is conducting.

Extra stretch goal: have Compy make an unhappy face when a note is played incorrectly.

PANEL 14.0 *ERROR, STRETCH GOAL:* TOP.5; BOTTOM.1





*Stretch goal: Have the notes that are incorrect to the pattern, **after** the song has finished, be marked abruptly with a red 'X'.*

The "X_note" asset is the exact same size as the note boxes, and can be aligned on the same coord it exists in.

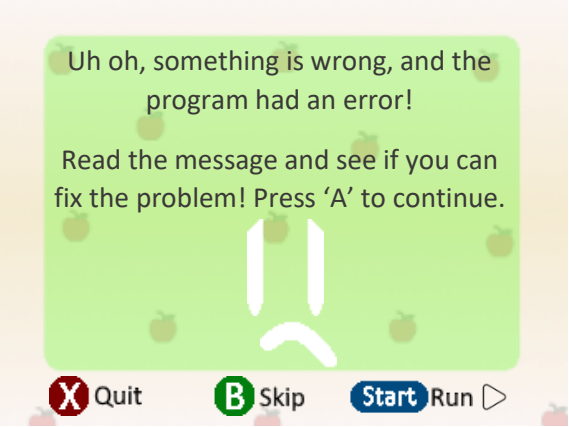
Have these sets of screens display for a brief pause, like 1.5s, then move on to PANEL 13.2.

Super stretch goal: have incorrect note asset's "shake" slightly before the "X" overwrites them.

PANEL 14.1 *ERROR, INCORRECT SONG:* TOP.5; BOTTOM.1

If the user's song does not match the pattern, these sets of screens can be used for any an all of the pattern matching scenarios throughout the game.

The error box has the text built into it as part of the image, but I have one that doesn't include the text if it shows up choppy/looks bad.



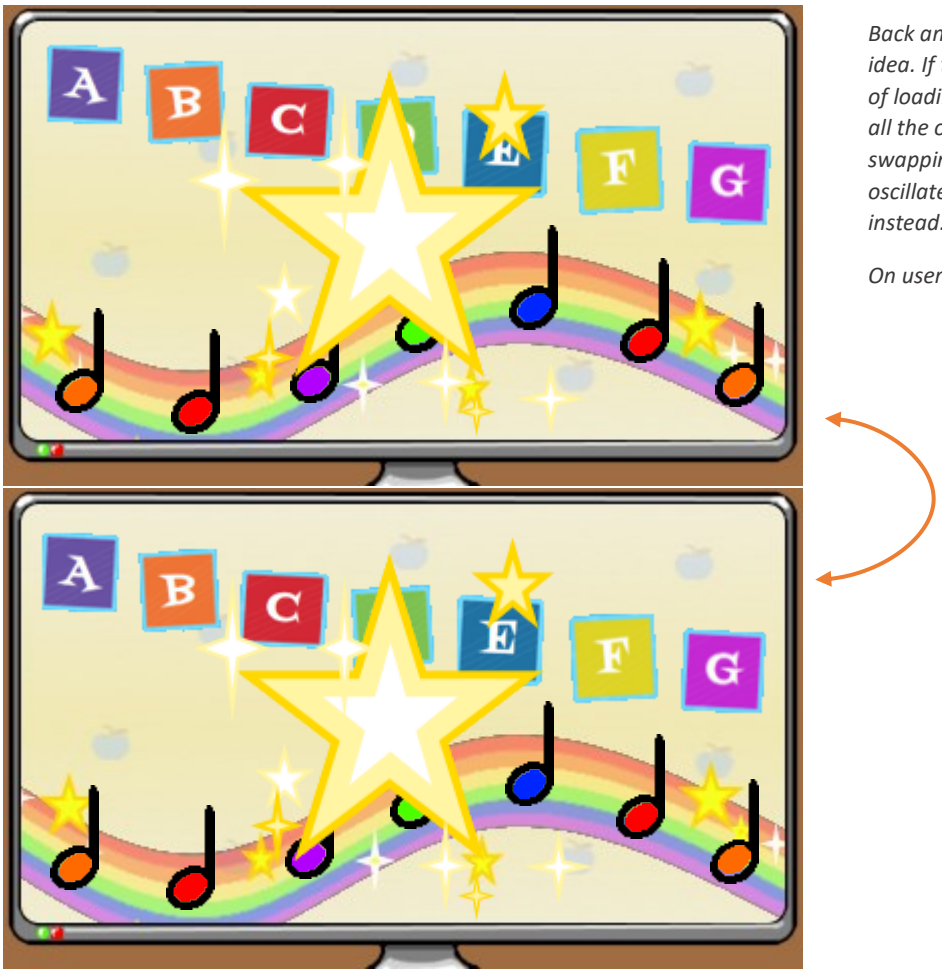
Compy's face just barely fits on here, the text box I use had to be expanded ever so slightly to prevent an extra newline.

Note: the definition of error is actually an important part of the dialogue, my preference is that it isn't removed.

The assets fit, but it seems a little more crowded than the usual fare.

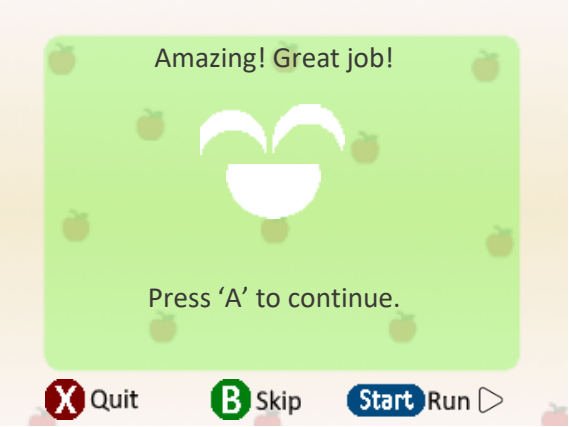
Once a user presses A, return to last known state of set PANEL 12.

PANEL 15 *WIN STATE:* TOP.5 <-->TOP.6 + Success_stars; BOTTOM.1

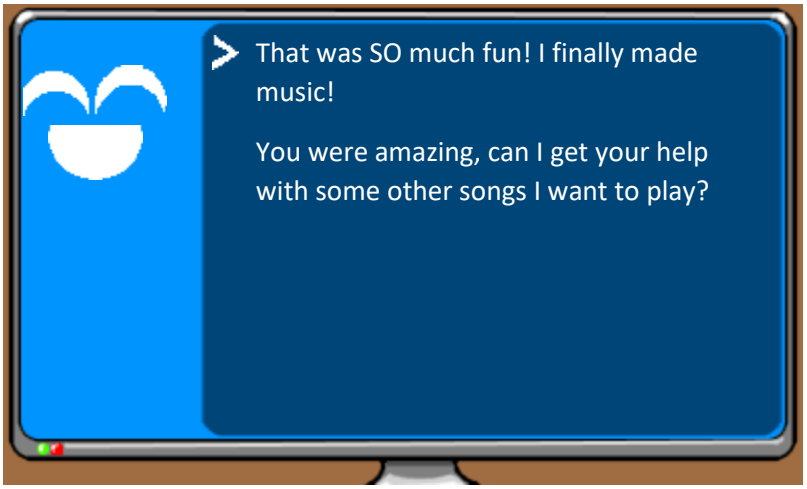


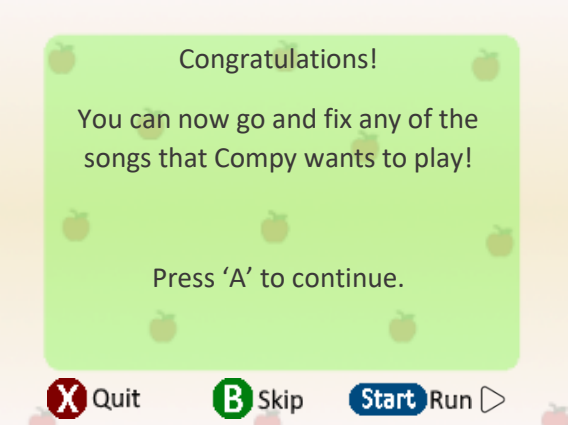
Back and forth, back and forth.. you get the idea. If there is significant overhead because of loading in TOP.6 (i.e., have to re-draw in all the other note stuff), leave the bg swapping out altogether, and simply oscillate the overlaying success_stars asset instead.

On user input, this game cycle is complete.



PANEL 16: TOP.4; BOTTOM.1





This is Compy’s last exposition for the first half of the game. After this, all games will end and return to PANEL 17 instead.

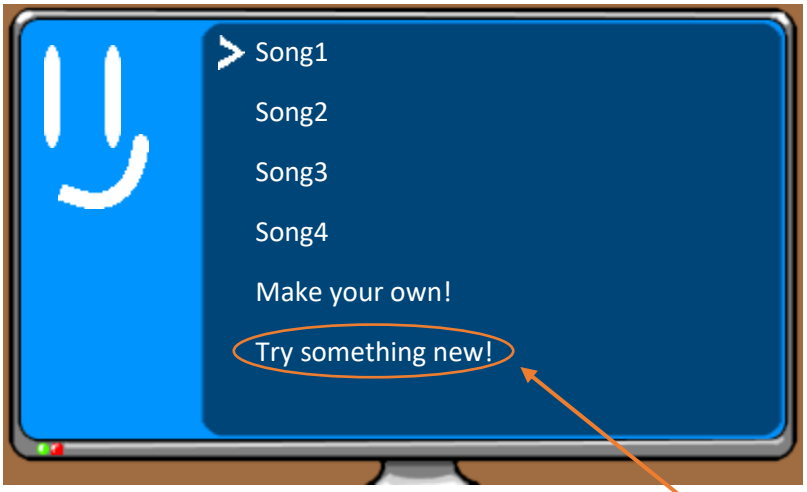
Compy will talk once more after a user has unlocked the multivariable experience half of the game.

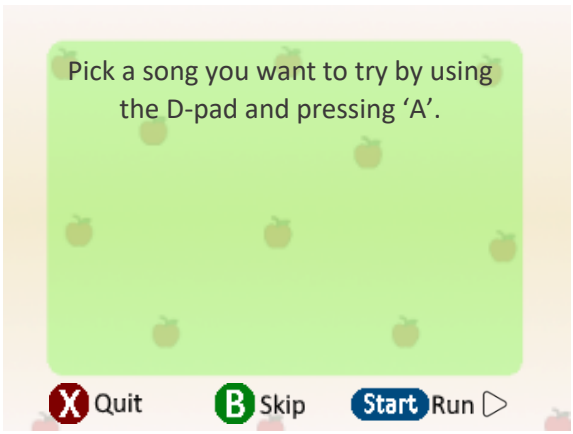
For below, on the next panel:

Note: need to make a decision – add the “try something new!” now or after 1+ songs are completed?

Note: Can add “make your own song!” now, but have to somehow restrict being able to build chords until “try something new!” sequence has been finished.

PANEL 17: TOP.4; BOTTOM.1





Eventually these songs will have proper names, I'm also interested in having a way to separate the single-value variable and multi-variable options.

Use the carrot asset to indicate which song is being selected.

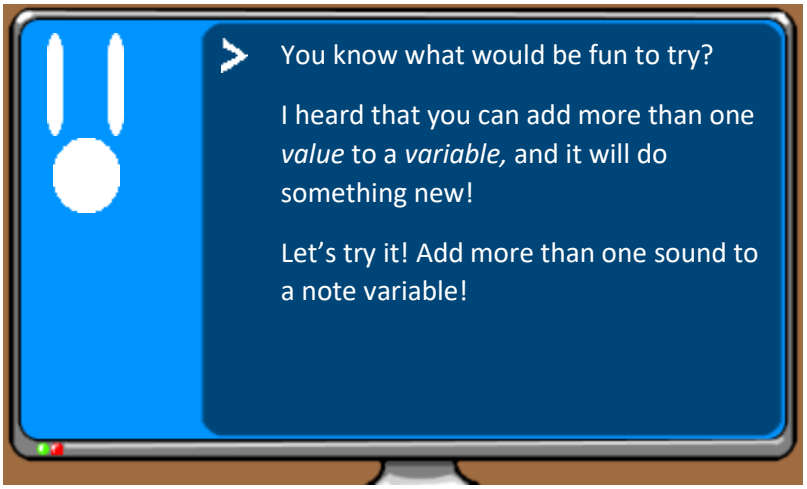
*Once a song is selected, load in PANEL 9.**2** and the correct song sequence. Run forward from there.*

Stretch goal: have Compy "animated" while sitting on this screen – maybe like in the beginning, where he blinks.

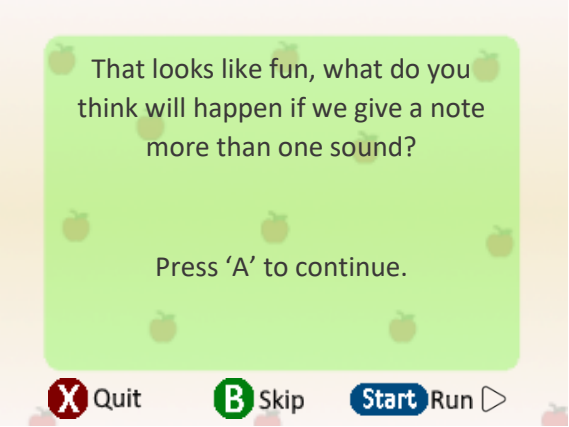
Super stretch goal: Interrupt animation to have him react with "=0" on highlighting a new selection, then returning to his standard sequence after a second.

This might not exist yet depending on what design choice is settled on.

PANEL 18: *MUTILVAR TUTORIAL START:* TOP.4; BOTTOM.1

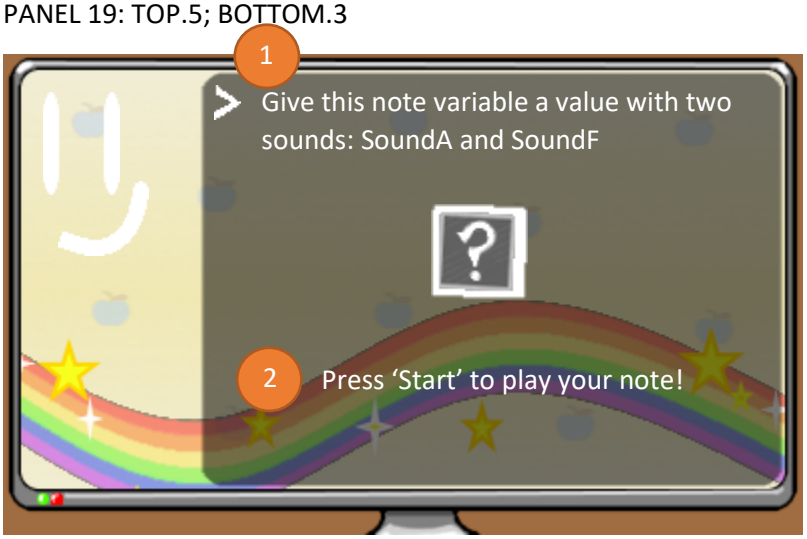


Selection of "Try something new!"



Changes: dialogue changes as of wk10

PANEL 19: TOP.5; BOTTOM.3

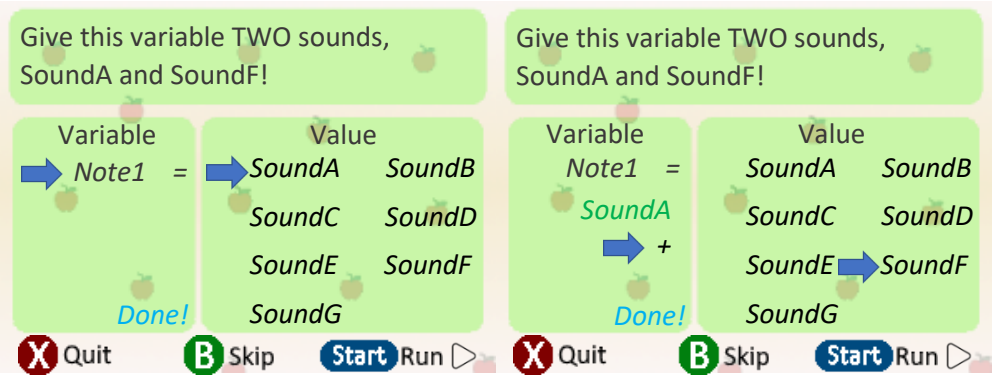


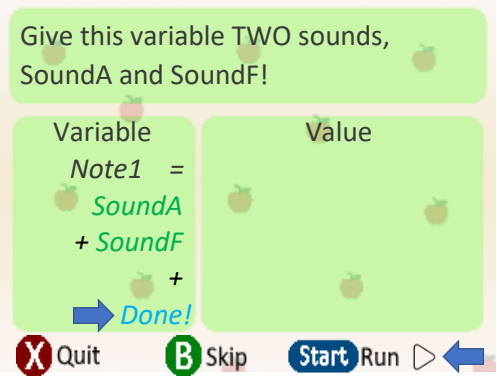
The bottom screen design was very tight and a little janky.

If it mostly follows this, we'll be able to construct chords that have a max of 3 values assigned to it.

Here, we only want them to assign 2, but I've still included the extra "+" since that's probably going to be present in the actual gameplay.

The already-assigned values should be selectable and replaceable, and the list of sounds associated with that Note should populate as data in that first list.





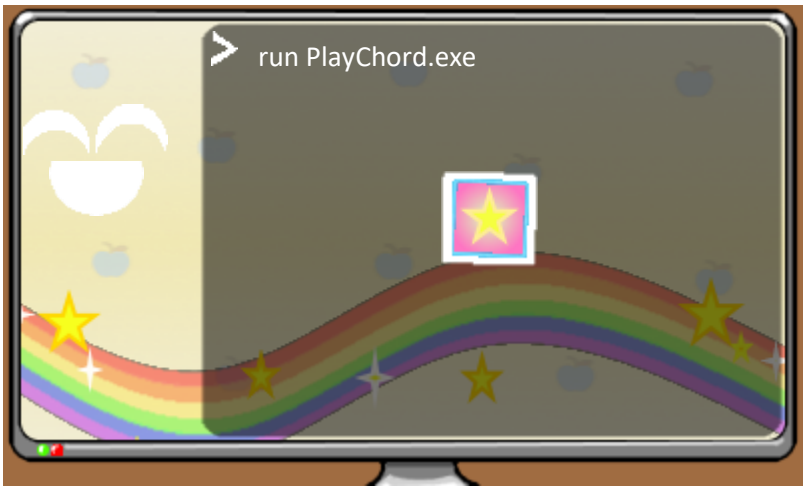
This is technically the tutorial, so if possible, I'd like to highlight the "start"/"run" button as soon as (any) TWO values are assigned.

The problem here is: if they accidentally assigned the wrong value, and need to change it before hitting run, we need to still allow the highlighting cursor to sit on the next element in the list (here, it's the second "+").

The top text should populate the (2) (check top screen above) text element as soon as two values are selected.

UPDATE: added "Done!" option to then allow d-pad control to swap off in active gameplay. Users still need to press "run" to complete tutorial.

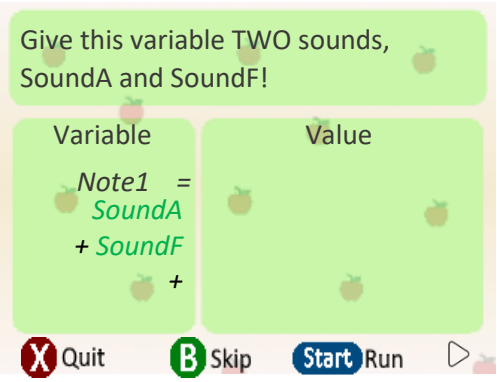
PANEL 20.0: TOP.5; BOTTOM.3



Pretty much a replica of P8.0, except the note asset is this generic block that I'm currently using to signal a variable is composite.

Fail: P20.1

Success: P20.2

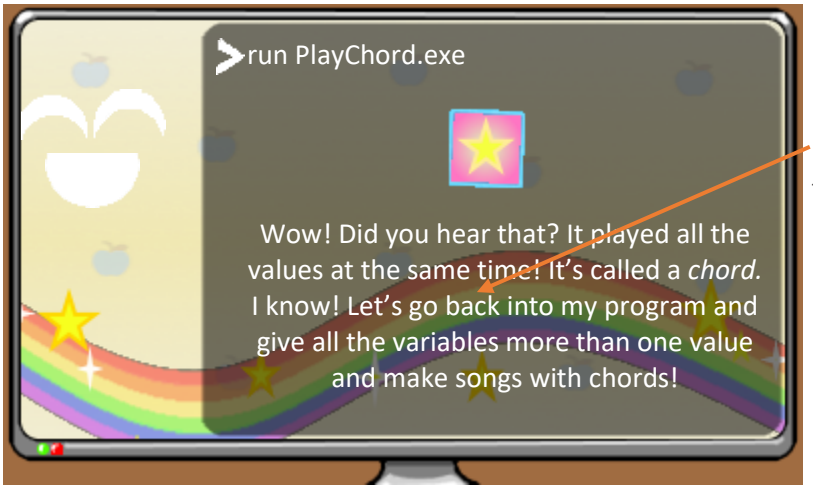


PANEL 20.1 *ERROR WRONG NOTES:* TOP.5; BOTTOM.3 (same as above)

The bottom screen can remain the same, but pressing 'A' here should take the player back to PANEL19. The bottom screen can either reset (nascent load of P19) or just update the top screen – whichever is easier.

Since the chord asset is generic, I couldn't really represent the "wrong" note, so the "X" asset makes a debut.

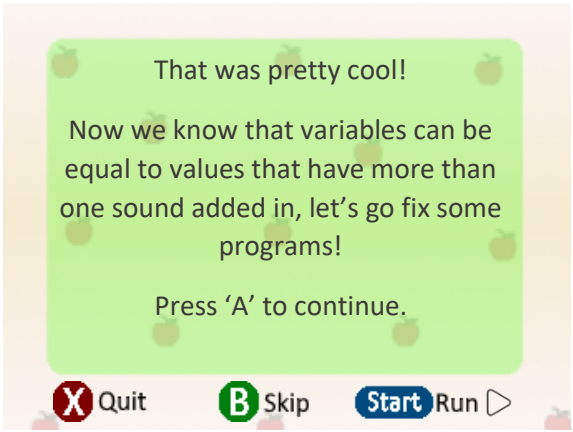
PANEL 20.2 *WIN STATE*: TOP.5; BOTTOM.1



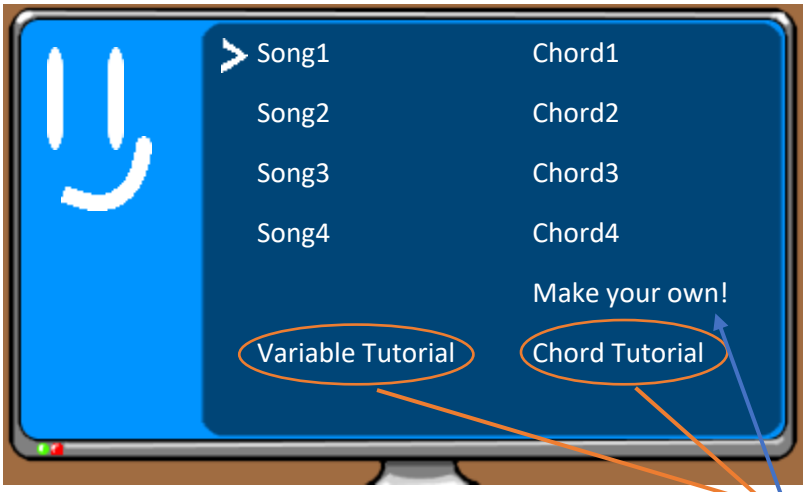
The teaching dialogue is a little wordy, but as long as it fits in it'll be okay.

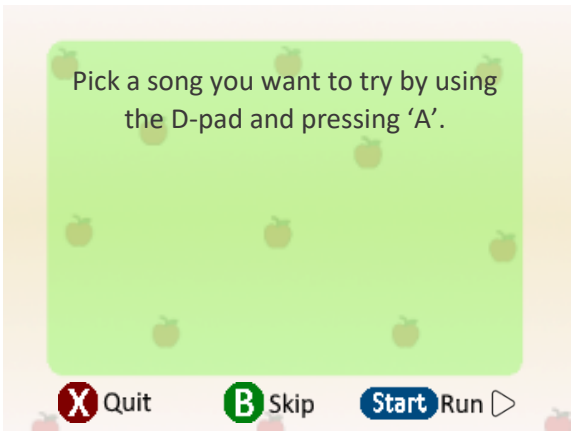
There is a line break between "chord" and "I know!". If it formats better on the DS so that there's a small amount of space there, it'll lessen how in-your-face this text wall is.

On 'A' (or start, or B, really anything), take the user back to the now fully fitted Menu: P17. Details below.*



PANEL 17* *UPDATED MENU:* TOP.4; BOTTOM.1





Okay, design #1: having the full menu be a double-paned table, where the indicator can now surf to the right or left to pick a choice.

Design choice #2: Creating a new "link" on the original (P17) that updates the text and options with the Chord song names/list.

If Design #2 ends up be the choice there, we need to add a button or option on the right side that a user can D-pad over to that will toggle to the next set of options available.

**Note: any stretch goal updates we've implemented will also be available here: the only change, truly is adding a list of options.*

Once the Second tutorial is unlocked, this is the menu until saved data is cleared.

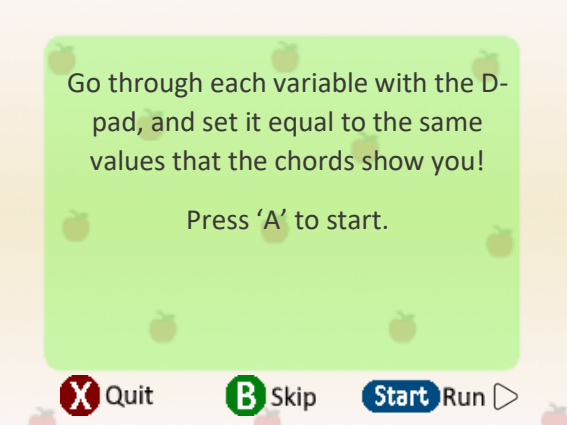
*On startup: Start: P1-P2, then load P17**
Variable Tutorial: restarts from P4
Chord Tutorial: restarts from P18
"Make your own!" Consolidated now that chords are unlocked.

***CONCERNS**: If saved data is not implemented, then every time a user restarts the game, they'd have to go through the tutorial experience again. In this case, have all SKIPS immediately available:*

- *(B) on tutorial #1 leads to P17 menu*
- *P3 has skip option implemented*
- *(B) on tutorial #2 leads to P17* menu*
- *No gating, 'try something new' immediately available*
- *On review of the game, lets try and find points where a user is "trapped" in a sequence and fix it.*

PANEL 21.0 TOP.4; BOTTOM.1

Intro upon loading up a new chord game. After users press 'A' off the bottom screen, update to actual gameplay starting with P21.1.



PANEL 22.1 *ACTIVE GAMEPLAY; DESIGN #1:* TOP.4; BOTTOM.3

Note: missing highlight behind the "?" mark assets here.

A note: The reason I went with this somewhat complex design choice is to reflect some tactile feedback to the user. Having the stack shrink back down into a star chord asset keeps focus clean, as the columns would be far, far too busy if all were loaded with stacks right off the bat, or were kept on screen afterward.

Design quandary: how to represent a chord visually using the current assets so that a young player can actually get the right answer?

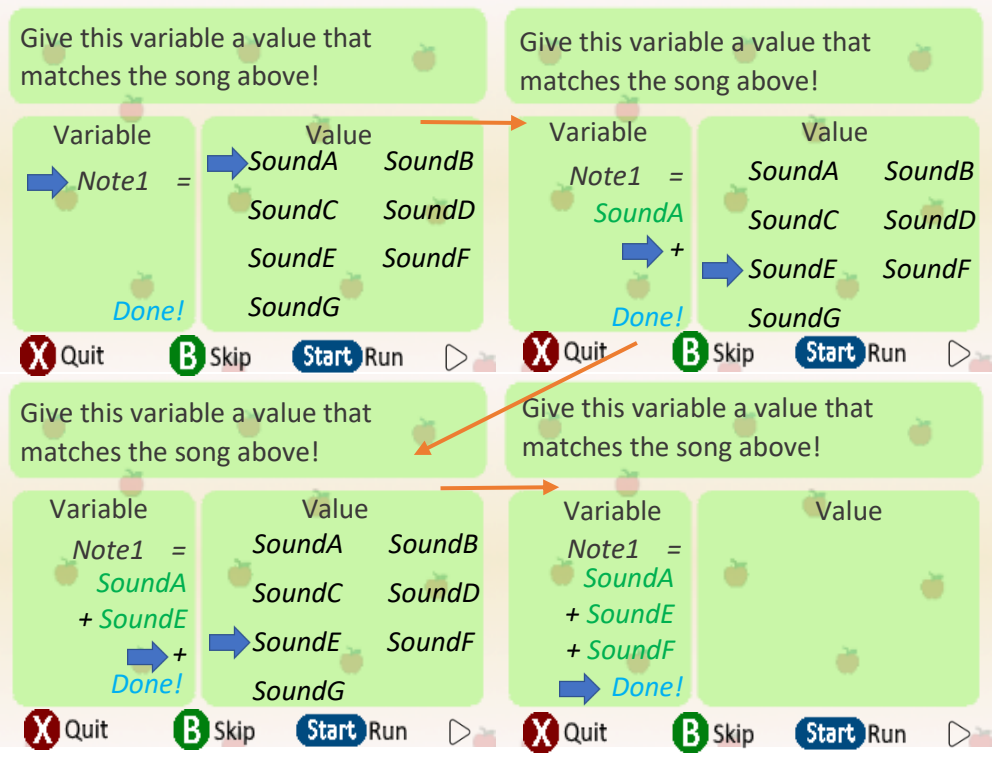
*Design #1: Create a stack inside the prompt cell (7 each, equal dimensions, refer to APPENDIX) that associates specific y-values and x-offsets per **resized** note assets. Shown here is a stack of all 7 – yes, there's overlap, that's okay here, it's a part of the style.*

*Implementation: Streamline how the backend actually "highlights" a given choice – the screen, split into 7 cells (APPENDIX images) associates a cell number with the variable, and the actual images can be wherever, technically. Upon a column highlight, the top asset will transform into the correct "stack", allowing a user to view the sequence they need to input. Once (any) **1 value** is assigned, the "?" turns into the composite star chord asset, regardless if it's correct or not. After deselection (pressing D pad in either direction) the stack will be replaced back with a star chord asset.*

Order of tasking:

- *7 col even split; each col has an associated x coordinate value (would be used in first half of game too)*
- *New arrays containing x-coord offsets and y-coords*
- *"selection" of a column calls an update on associated prompt asset*
    - *Draw in asset A, E, F with coords:*
      `(col_loc+xcoords[0], ycoords[0]),`
      `(col_loc+xcoords[4], ycoords[4]),`
      `(col_loc+xcoords[6], ycoords[7])`

  Where, of course, the magic numbers are pulled from the win-state sequence array.
- *Upon userinput, draw in chord asset*
- *Upon deselection, replace prompt with chord asset again.*



Sequence of bottom screens to show how the text develops as users make selections.

***Oh boy, we missed a step. We need a "confirm" so that the program knows what the d-pad is actually paneling in between.*

Update: added "Done!" at bottom in left-hand panel.

Update: added the "Done!" to earlier frames.

Implementation: Pressing 'A' on "Done!" will release the bottom screen, and return user control to D-pad select from the options there instead.

PANEL 21.2 *ACTIVE GAMEPLAY; DESIGN #2 (WIP):* TOP.4; BOTTOM.3 (same as Design#1)

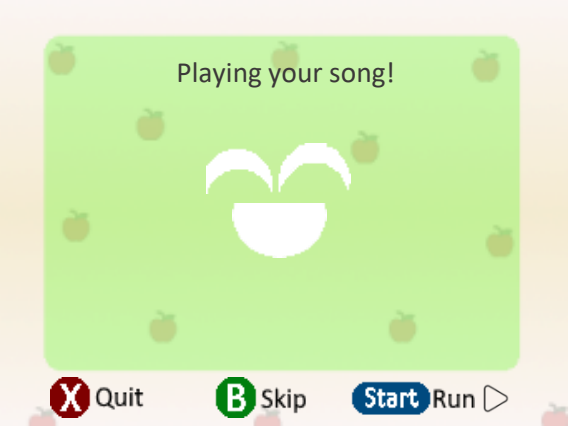
PANEL 22: *PLAYING SONG;* TOP.6; BOTTOM.1

Reuse everything from P12-16 with only ONE exception:

The sequence that is displayed on the top row of course will not have individual note assets. For now, instead, have just a straight row of star-chord general assets (as shown).

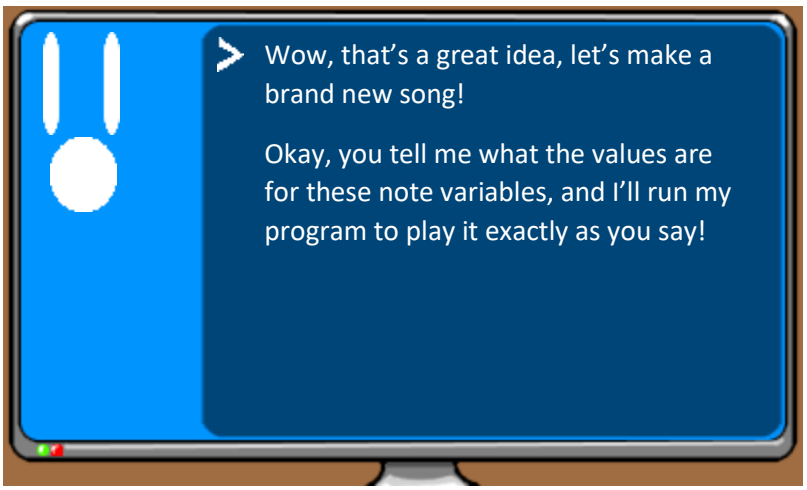
STRETCH GOAL: Per chord/note being played, have the star-chord asset be replaced by the stack of relevant notes that was ***selected by the user.***

- *This might have a ton of overhead, hence the stretch goal.*



*Again, reuse **everything** from P12-16 only excepting that one asset change. This bottom screen is just an example, refer to P12-16 sequence for actual goals.*

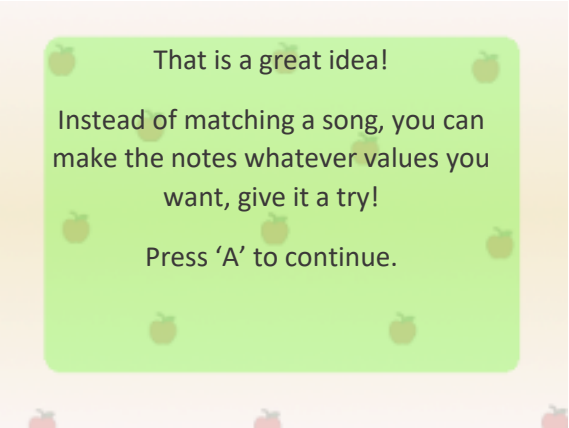
PANEL 22.0 *MAKE OWN INTRO_1:* TOP.4; BOTTOM.1



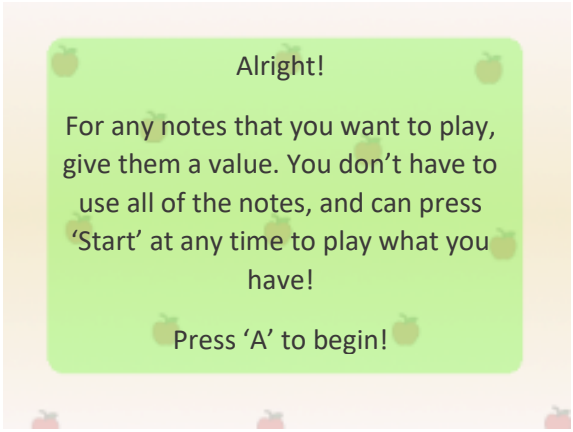
Upon selection of "make your own"!

Sequence A: Chords not unlocked

Sequence B: Fully unlocked version.



PANEL 22.1 *MAKE OWN INTRO_2*: TOP.6; BOTTOM.1



PANEL 23.0 *MAKE OWN;* ***SEQUENCE A***: TOP.6; BOTTOM.3

PANEL 23.1 *PLAYING USER SONG:* TOP.6; BOTTOM.1

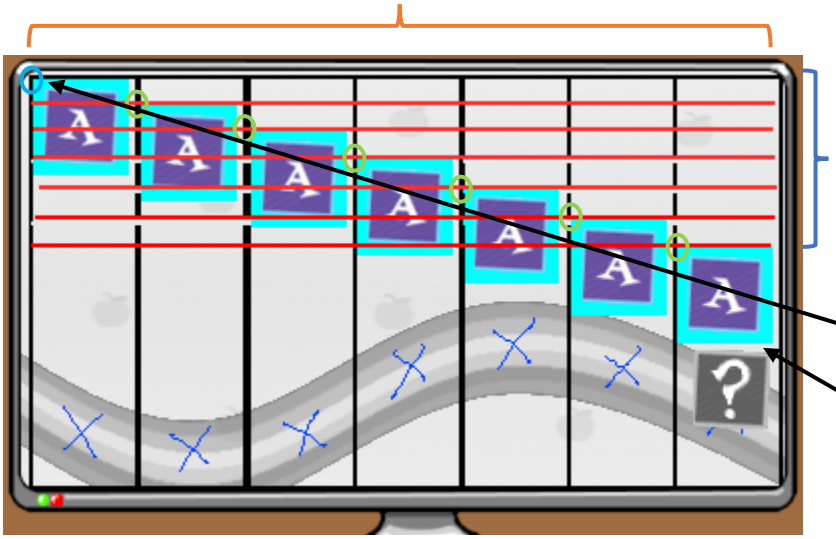
PANEL 24.1 *MAKE OWN;* ***SEQUNCE B:*** TOP.6; BOTTOM.3

PANEL 24.1 *PLAYING USER SONG:* TOP.6; BOTTOM.1

End game loop.

APPENDIX FRAMES

> Columns evenly spaced. Note, they are NOT splitting the entire DS screen, but rather specifically "Compy's" screen.



6 y-coord locations that design where the top left pixel of the note asset should start. The transparent edges of assets have been colored in for visual clarity.

This coordinate is the (0,0) of Compy's screen.

This is the top-most y-value that the user prompt line can start.

NOTES:

- Highlighting decisions should be determined very early in so as to not bog down coding
  - On 2nd or 3rd pass, consider ramifications of updating the "quit_skip_run" asset and ability to highlight only the available choices
  - Unavailable option: opacity 50%; Tutorial highlight: Aura/halo/keep the arrow
  - For ALL ITALICS, change to a different color, such as light blue or green
  - In current version of variable tutorial (game1, coded by Ernesto), the Variable/Value menu has returned to using a table. It is likely that there will NOT be enough real estate on the screen to keep this functionality. Reconsider using that code or find measures to reduce cell size enough to be able to fit 1+4 rows (table header + 4 suboptions).
- Considering drawing a diagram that integrates the use case + available pathways (task@Eryn)

USE TABLE:

Section	Panel range	Quit Action	Skip Action
Start	P1-P2	Seedlings Menu	N/A
Intro	P2-P3	Seedlings Menu	N/A
Single Var Tutorial	P4-P16	P1	P16
Menu	*$P17_1$	P1	N/A
In-game: User input	P9.2-P11 /	P17	N/A
In-game: Playing song	P12-P16 /	P17	P13/P14 (lose/win)
Menu	P17	P1	N/A
Multivariable Tutorial	P18-20	P1	P16
Full Menu	*$P17_2$	P1	N/A

Make your own: A	P	P17	N/A
Playing song	P	P17	P
Make your own: B	P	P17	N/A
Playing song	P	P17	P

*$P17_1$: Incomplete multivariable tutorial, *partial options available*

*$P17_2$: Full menu, *all options → includes second panel of multivar options.*