12/11/13

Thesis: methodology brief

I begin alliteratively with a note about iteration during ideation. Before I started building prototypes, I kicked around various ideas for how to make a new kind of music-driven video game. These mental prototypes are difficult to user-test, so I eventually settled on the latest and strongest version and started the laborious process of giving it life outside my brain.

I mention this because, going back now and rereading my ideas, I notice how elements of each have crept into the final concept. Though at the time of writing I thought of each idea as discrete and new, it's clear now that this was very much an iterative process in miniature. I implemented and tested concepts in my head with the help of my inner player (a difficult fellow to satisfy), stripped out the weak elements while preserving the strong, and moved on to the next phase. It's quite likely that as the project continues to shift and evolve during its birth--a shockingly-accelerated Darwinian process if there ever was one--it will come more or less to resemble all the different permutations that came before. Perhaps the final version will be a genius blending of all those fruitful ingredients, a video game smoothie. One can only hope.

In any case, I won't dive into those pre-prototype ideas here, but it's important to note that the very first *idea* was a tiny Zeus and then he traveled the world and had experiences and exercised at the Olympian gym and got stronger and finally the first prototype burst from his forehead and was called Athena. I've now created four prototypes in total, though each builds on the previous one so really it's the same prototype. They're focused differently so they bear individual examination.

First prototype--the staff

Since my game turns the musical “staff” of sheet music into a playable environment, the first goal was to recreate the staff on-screen. A staff is a series of five horizontal lines the length of a page, arranged vertically and spaced apart equidistantly. “Notes” are oval-shaped marks on the page--the vertical position (on or in-between the lines of the staff) indicates pitch, while the horizontal position indicates when the note should be played. Typically a single line of sheet music comprises two staves, one for higher notes and one for lower notes. Here’s an example:

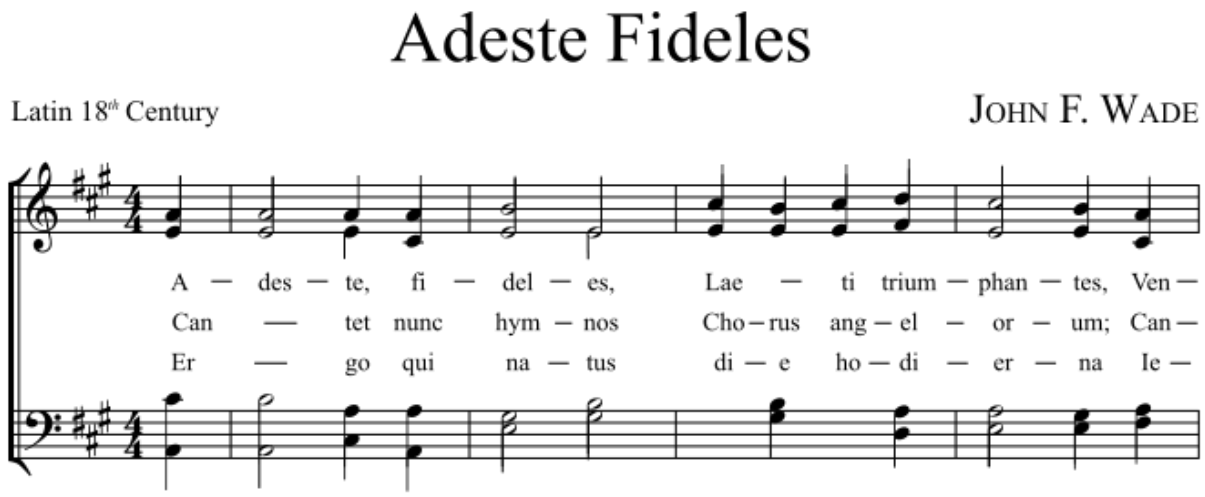

Fig 1. A single line of sheet music.[1]

The curly marks that resemble a capital “G” and the number 9 on the left sides of the staves are “clefs,” indicating whether the staff holds higher or lower notes. In between the staves lies “middle-C,” the note more or less in the middle of a piano keyboard. See the note just below the syllable “Er” at the beginning of the bottom lyric and the note just above the syllable “fi” four notes into the top lyric--they are the same note, middle-C, and the two staves meet here, though they are separated visually by the lyrics in this case.

[1] Src: http://en.wikipedia.org/wiki/File:Adeste_Fideles_sheet_music_sample.svg

For this initial prototype I decided to implement only a single staff, unmarked for the time being though it would function as the treble (higher) staff. This was as easy as drawing the five horizontal lines and spacing them out evenly. I included room above and below the staff to include additional notes, enough to cover a full two octaves, ascending from B-below-middle-C up to two additional Bs (BCDEFGABCDEFGAB).

The next step was to add notes, so I set the program to create a random note on the right every time I clicked, with the notes moving from right to left. Here's one result:

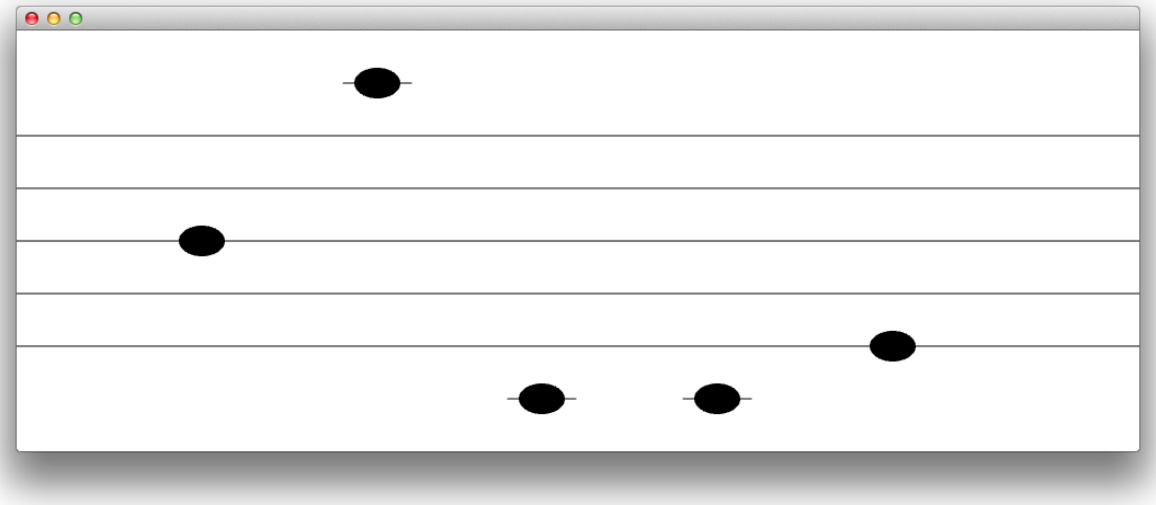

Fig 2. Some notes![2]

This was fine, and worked as intended, but not very useful for testing. It's only interactive in the loosest sense, since one can't do anything but create random notes, and they don't actually play sounds yet so they're not even really "notes."

Second prototype--sounds and interactivity

It might seem like the first task in the second stage would be to add the sounds, but there's an immediate problem, which is how and when to *play* them. When should the notes

[2] Unless otherwise indicated, all prototype screenshots come from running an openFrameworks application.

“sound”? When they first appear on the right edge? When they cross the left edge? What if they don’t move? There is no musician performing the music as yet; rather, the music performs itself. But how to conduct it?

Audio software often uses a “playback” bar--a vertical bar that moves horizontally across the screen and plays a note by passing over it. Here’s an example from “MusicBox (Synth)”:

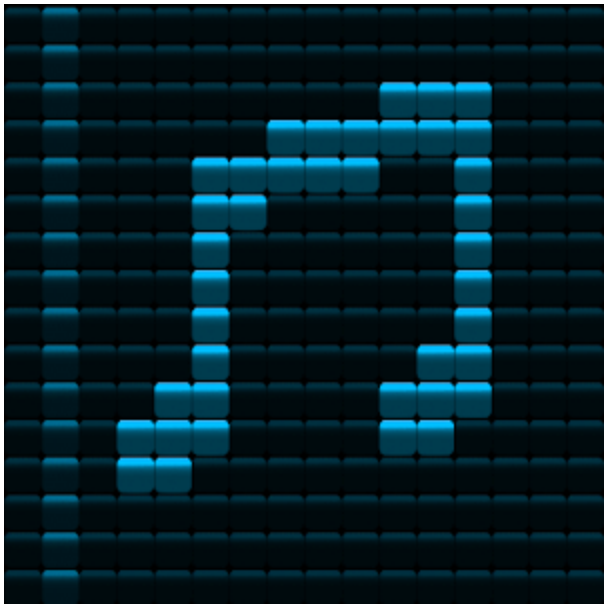

Fig. 3. Audio synthesizer with vertical playback bar running up the left side.[3]

The rectangles making up the light-blue image are notes, which the dark blue bar plays upon passing over. Perhaps in my project I could use the player’s avatar as a playback bar. After all, this mash-up of sheet music and *Super Mario Bros.* requires a “Mario” to run and jump through the world of notes. Why not play the notes when the avatar gets close?

So, I added an avatar and laid out some notes, and this time I gave the notes their actual sound, using audio files I had created for a previous music-based project last semester (the progenitor of my Thesis, really). I set the sounds to adjust their volume based on proximity to the

[3] Src: https://play.google.com/store/apps/details?id=com.appspot.kimurastudio.musicbox2

avatar, so they would get louder when it was close. I can't show the audio playing, obviously, but here's what the result looks like:

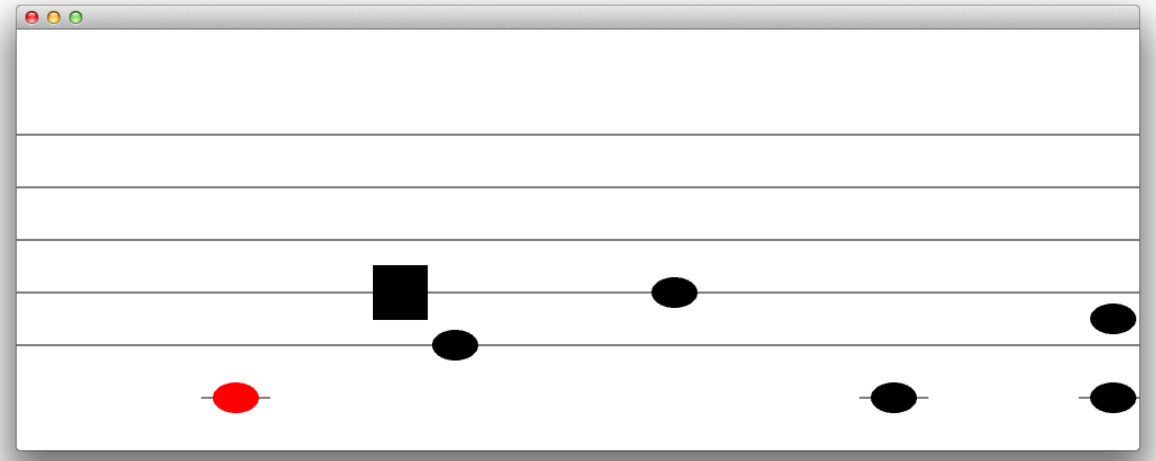

Fig. 4. Second prototype: playable avatar (square) with proximity-based audio from the notes.

The large black square is the avatar, which the player can move in all directions using the arrow keys. The notes are static but arranged and spaced to form an actual musical riff, i.e. a real musical idea. Right now, the only note sounding is the second, to which the avatar is close. Moving to the left or right decreases the volume of that note while increasing the volume of the note in the direction of movement. Moving close to the two notes stacked atop one another on the right side (a "chord") allows the player to hear both at once, thus playing the chord.

It's embarrassing to admit but it took me a while to realize that I attached the wrong audio to each position. That is, the first note visually is a C but aurally it's a D. I fixed it in the next iteration, but there it is. Inconsequential in this case except to my pride. The audio is also the wrong octave (lower-frequency versions of the same notes), but it works for a quick prototype,.

I mentioned that the notes are static, but I added a keyboard control to set them in motion, and I set them to reappear on the right (after they go off the left edge) with enough delay

so as to be rhythmic. That means that if the player doesn't move the avatar and just lets the notes move, he or she hears the riff, repeated with the correct timing to match the tempo determined by the spacing of the individual notes. Here's a visual example:

Fig. 5. The end of the riff disappears on the left while the beginning reappears on the right.

The original idea for this project was to have the player battle "boss" enemies, who would each generate and repeat a specific musical riff, and the player would have to interact with it in some way to defeat the boss. In the case of the above, the player would need to "record" each note of the riff and then "play" the whole riff back at the boss to defeat it. As such, with this prototype I also took the first crack at implementing the recording and playback mechanic:

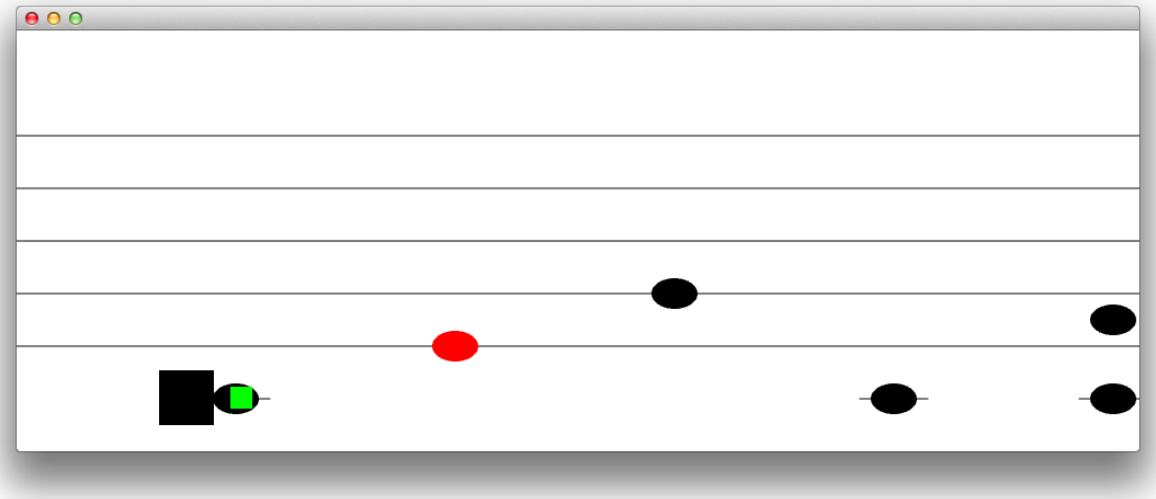

Fig. 6. The avatar's recording box overlaps the first note, "recording" it.

The player can press and hold a button to summon a recorder, represented by a green box, which makes a copy of a note by colliding with it. Note that in the image above, the red highlight has moved from the first to the second note, to indicate which one to record next in the riff.

Having recorded a note, the player can press another button to "play it back." In practical terms this means that the program creates a copy of the recorded note at the player's horizontal position and sends it moving to the right. This gives the effect of chucking the note "upstream," as it were. Here's the first note, played back:

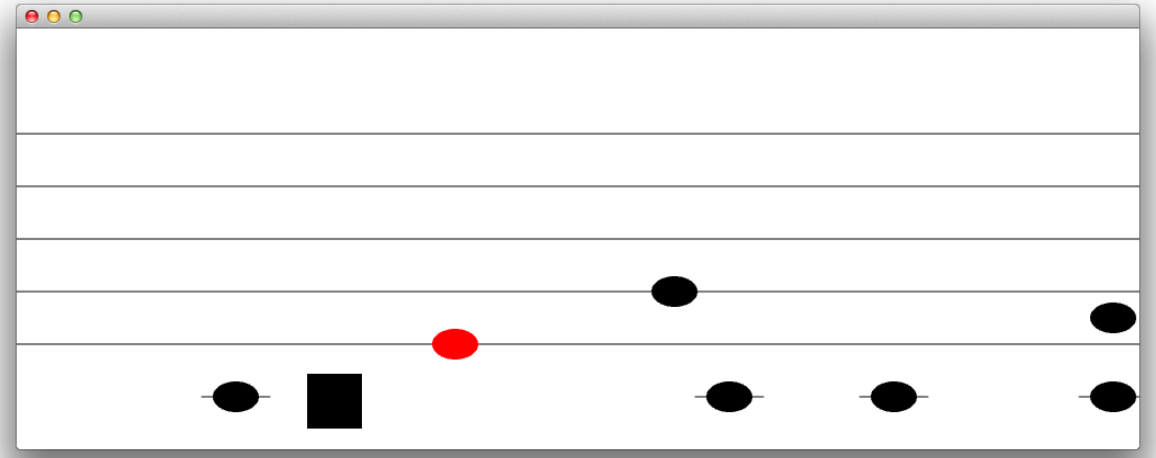

Fig. 7. "Playback." The note just to the right of the third note is a copy of the first note.

I also implemented a riff-playback system that would cause the avatar to freeze in place until all recorded notes had been replayed, and would space those notes according to their original spacing. Hence, a true copy of however much of the riff the player recorded before replaying.

This system worked quite well, as long as the original notes remained static, but as soon as the notes were in motion (an integral part of the planned combat), the spacing calculations got screwed up, occasionally resulting in a mess like this:

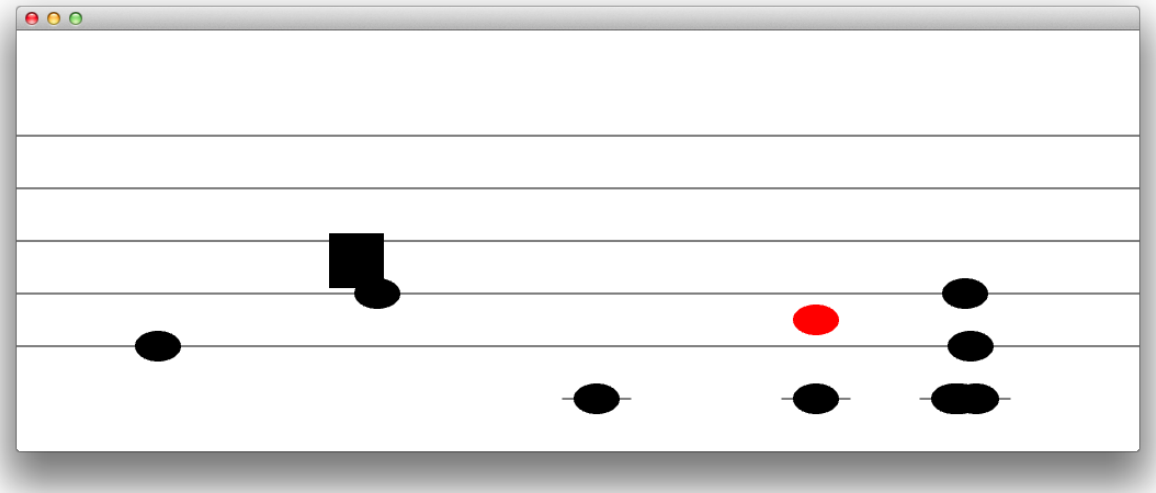

Fig. 8. Playback of recorded moving notes. That mess on the right is all the notes at once.

At this point, I had my first chance to playtest and get feedback, when I sat down for an individual meeting with John[4] and showed him what I had. He pointed out quite accurately that there isn't really any gameplay yet, and when he heard that I was planning to implement jumping (rather than the free motion possible in this prototype), make the notes physical objects for the avatar to jump on and dodge, and focus on the moving notes rather than the static ones, he urged me to pursue those things immediately.

Third prototype--jumping, physicality, playback, and feedback

This next iteration was a substantial upgrade. The first things I did were to add a force of gravity to pull the avatar to the ground, and a jump command--no more free vertical motion.[5] The next thing was to make the avatar collide with the notes, meaning something would happen when they touched. I had this already in the form of the recorder, so I built it out further, making the notes solid objects through which the avatar could no longer pass. In order to facilitate this, I changed the shape of the notes from ovals to rectangles, since it is easier to calculate the overlap of rectangular shapes. This also made more sense conceptually, since the notes were acting as platforms on which the avatar would stand (and presumably not slide off):

[4] John Sharp, my Thesis instructor this semester.
[5] This, as well as just about everything else related to technical implementation, involved a lot of challenges and tweaking to get right, which for the most part I won't go into.

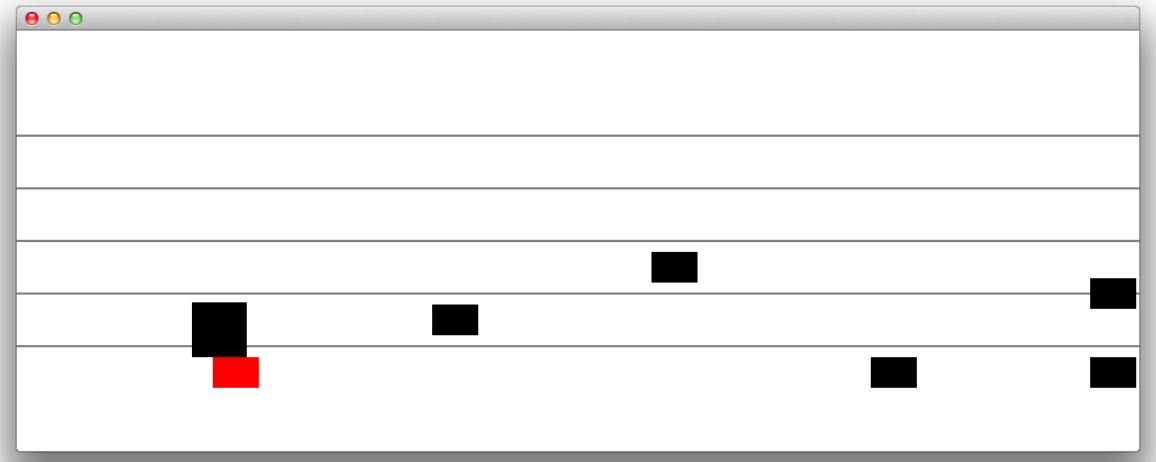

Fig. 9. The avatar rests atop a note (now rectangular), which prevents it from falling.

Furthermore, moving notes now imparted their motion to the avatar, so it could be carried along (or pushed out of the way) without any input from the player.

I also worked extensively on the recording and playback mechanics and feedback. For instance, the initial version of the recorder was just a static green box appearing in front of the player, which was visually dull and also made it hard to record since it forced the avatar to be just to the left of the note. I changed this to a green circle that rapidly orbits from the upper-left of the avatar clockwise 360 degrees, sort of like swinging a net over one's head to catch something. This had the advantage of being more visually dynamic while also enabling the player to record a note from any angle. Finally, I set the notes to turn green while being recorded, so the player could tell something was happening.

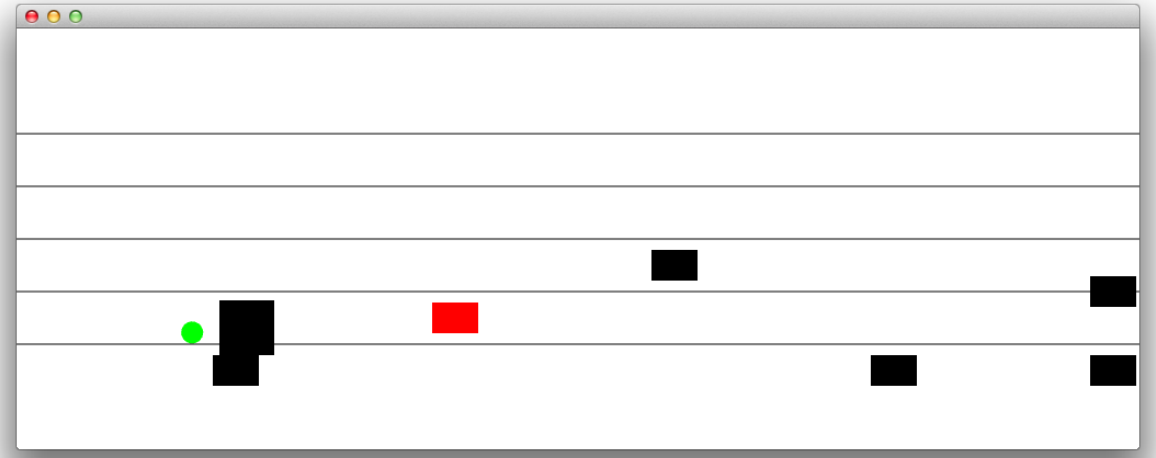

Fig. 10. The recorder is now a circle that orbits a full 360 degrees around the player at speed.

I added similar feedback for playback, this time with a blue circle that orbited counter-clockwise 360 degrees from the avatar's lower-left, like a quick underhanded toss. I also added a question mark to appear above the avatar when trying to replay with no notes.

Though I liked the idea of replaying a full riff with the original spacing, I didn't like the way it took control away from the player, and I worried it wasn't compatible with actions such as recording notes out of order, plus there was the spacing issues I ran into with the moving notes in the previous iteration. Eventually I decided it wasn't worth the problems it caused and I removed the "playback mode," meaning the player now played back notes one at a time regardless of how many were currently stored, and could simply keep playing notes back (or not) at whatever spacing and timing was desired until there were no more. This greatly improved play flow, although it raised a new problem of how to replay chords.[6]

I also fixed a conceptual consistency issue with the replaying from the last prototype. Previously, recorded-and-replayed notes would move, even if the source note was static.

[6] A chord is multiple notes played simultaneously; the new system only allowed individual note playback.

Instead, I made it so that replaying recorded static notes produced static notes, while replaying recorded moving notes produced moving notes (though moving in the opposite direction).

Finally, I implemented various technical systems, such as one to prevent the player from recording many copies of the same note with a single action, and I gave the replayed notes a limited lifespan, after which they would fade away and disappear. And, in a nod to both a diverse audience and the need for better presentation, I created a title screen with control settings:

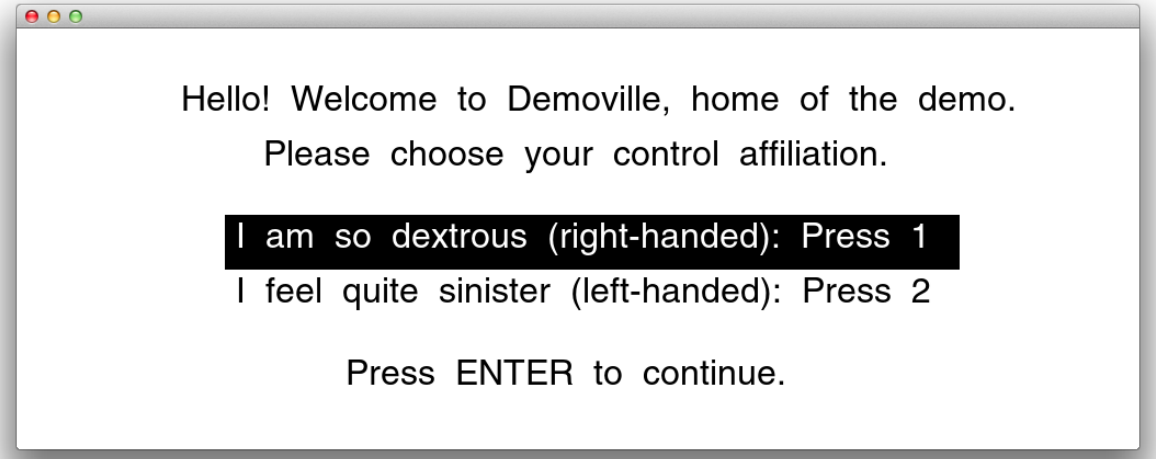

Fig. 11. The title screen lets the player choose controls.

Fourth prototype--scrolling and tutorial

I now had the systems in place to start building actual gameplay, with goals and challenge. I decided to make a tutorial level, in which the player would move the avatar from left to right through a horizontally-scrolling environment, with notes as well as non-note obstacles, and the tutorial would teach the player how to play by challenging him or her to learn and use the mechanics to get past the non-note obstacles.

The first challenge was how to implement the scrolling. Scrolling is analogous to viewing the world through a camera's lens, and panning to follow something, with surrounding objects

sliding into and out of the camera's view. I credit the esteemed Michael Kahane for helping me use a pre-existing, very simple camera system, which I attached to the movement of the avatar.

Then I had to actually make a level. I divided up the controls and mechanics and laid out situations that would require the player to use them. For instance, the tutorial opens with this:

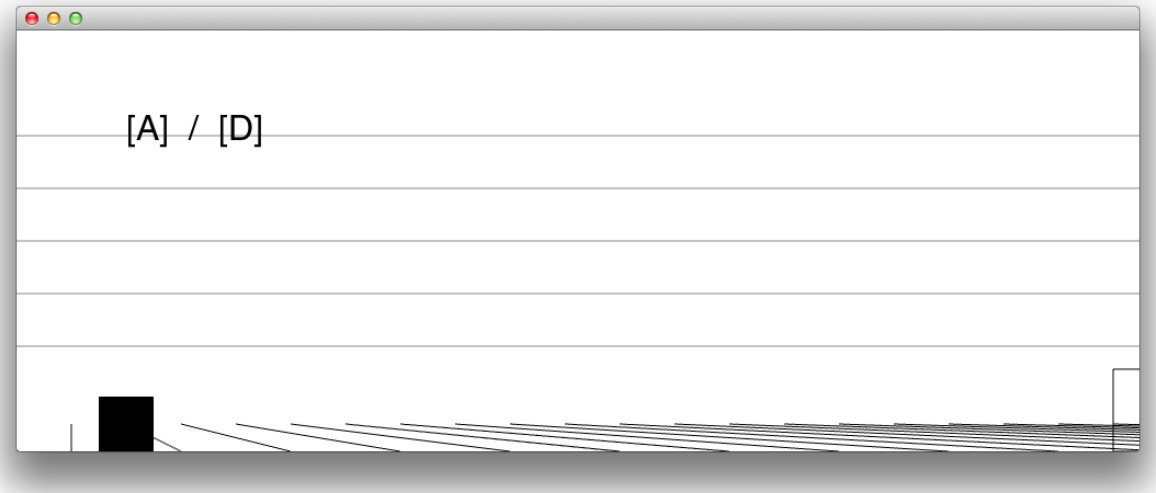

Fig. 12. The first screen of the tutorial.

The hatch marks along the bottom are there to give a ground-like texture. The screen is otherwise empty except for a mysterious shape just appearing at the right edge of the screen, and some letters floating above the avatar's head.[7] By pressing the keys for those letters, the player discovers how to move left and right. Going to the right reveals this:

[7] Note that if the player had chosen the left-handed configuration, the prompts would be different.

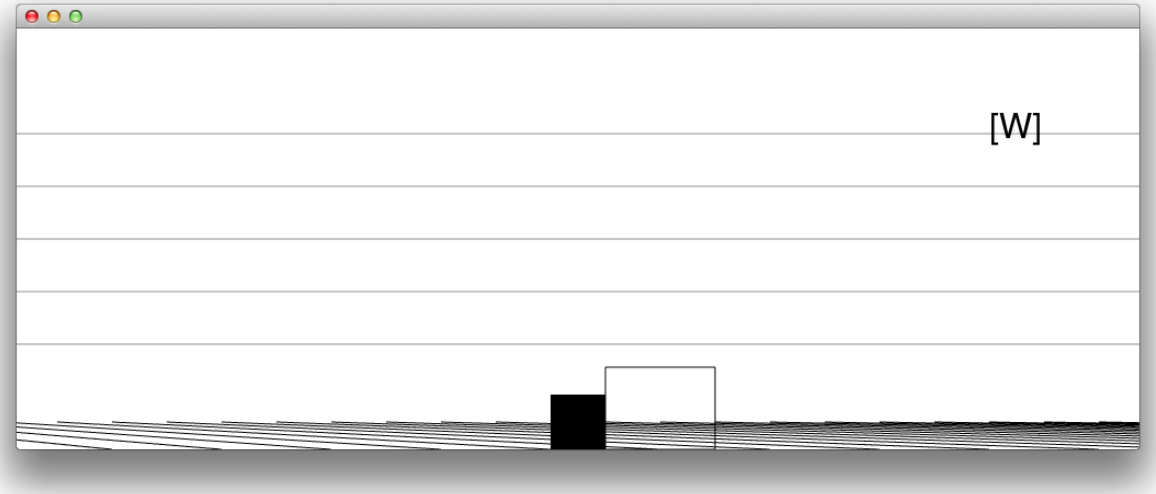

Fig. 13. The first challenge of the tutorial.

The avatar cannot push through the hollow rectangle. However, another prompt floats in the upper-right. By pressing this button, the player can make the avatar jump over the obstacle.

Continuing a little ways to the right, the player encounters this:

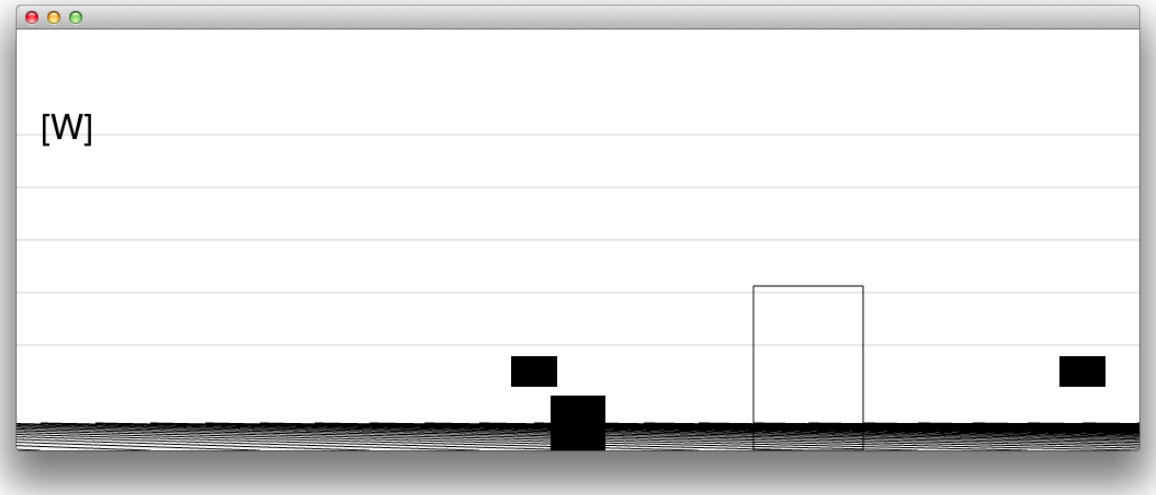

Fig. 14. The first appearance of notes, and the second challenge.

Assuming the player has the sound on (I later added a prompt on the title screen to encourage this), he or she discovers audio plays in proximity of this solid object. Furthermore, it is

necessary to jump on the object to get over the next obstacle. The player has learned that solid objects play sounds and they are tangible.

Continuing onward the player is prompted by the button for recording and later for playback. These are presented without explanation and floating by themselves away from any context. The idea is that the player presses the buttons and receives the visual feedback, but isn't sure what the actions are for. However, the way onward is blocked by an obstacle too tall to jump over, and there's nothing in the vicinity to use as a step. The player figures the new actions are probably key, but how? Eventually he or she returns to the earlier notes and discovers that it is possible to record a note and then play it back somewhere else. Using this skill enables the avatar to get over the next obstacle.

There are a few more puzzles to teach the player about using lower notes to reach higher notes, that notes have limited lifespan, and that it's possible to record more than one at once. At the end, the player gets a prompt to press a button and enter into the boss battle from the previous iterations. The battle is (still) not yet complete so there is a humorous note to that effect, although the riff is there and in motion and the player may interact with it:

Fig. 15. The conclusion of the tutorial, with boss battle still under construction.

I was pretty pleased with the state of this prototype. It had the basic systems in place, a functional tutorial, and a display of both kinds of gameplay, puzzle-solving and combat (sort of). However, up to this point I hadn't done any playtesting since John took a look two versions ago. I recruited my friend Mauricio Sanchez-Duque to playtest the prototype for me.

The results were...eye-opening. Mauricio could not get past the first real puzzle, requiring the use of recording and playback. He simply kept pressing the recording and playback buttons over and over again while pushing fruitlessly against the obstacle. It never occurred to him to go back and try to interact with previous objects. It was painful to think of all the content I'd created afterward that he wasn't seeing, but I'd sworn not to say anything unless he discovered a bug, as per good playtesting protocol, so I stopped the playtest there. He told me he didn't have sufficient context to interpret the prompts he was being given. For example, he interpreted the prompts for recording and playback as directions on which way to go,[8] *not* as button prompts for action. It was clear that things were not clear, at all, that the guidance was simply too vague to be useful as clues. It was extremely useful information for me. I thanked him and asked him to try it again the next day.

I should clarify that I made these design choices deliberately. I didn't want to do a very explanatory tutorial that holds the player's hand, e.g. "Press this button to do this thing! Do that thing over there now!" I liked the idea of the tutorial itself being a puzzle, or multiple puzzles. Let the player experiment and figure it out! But I had obviously gone too far.

I went home and made improvements. I changed the formatting of the floating prompts somewhat in the hopes they'd be more obvious as *button* prompts. The settings on the title screen hadn't been intuitive either, so I changed them:

[8] In the right-handed configuration, record and playback use the LEFT and RIGHT arrow keys.

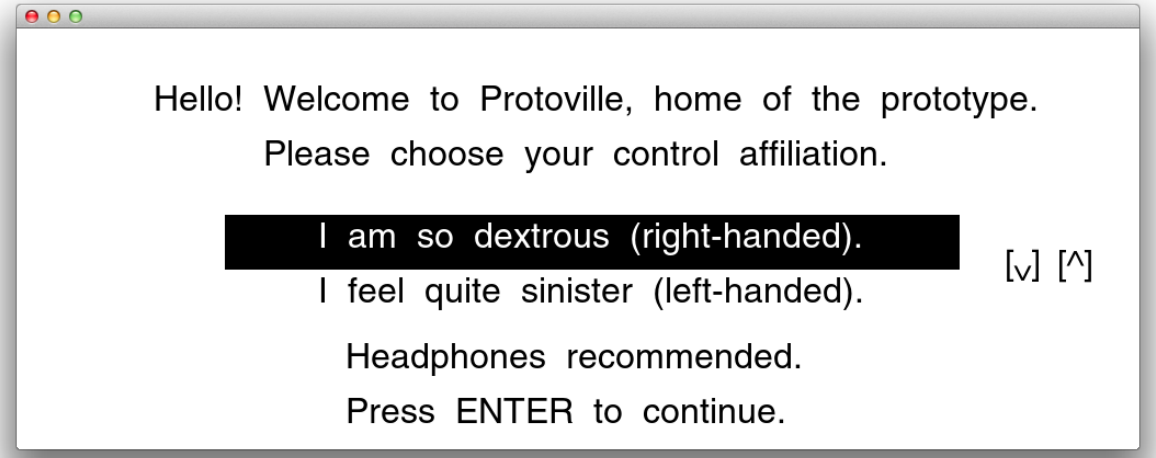

Fig. 16. Now the arrow keys select the controls, and there's a prompt about audio.

I also added a floating note in the tutorial below the prompt to record. It looks like this:

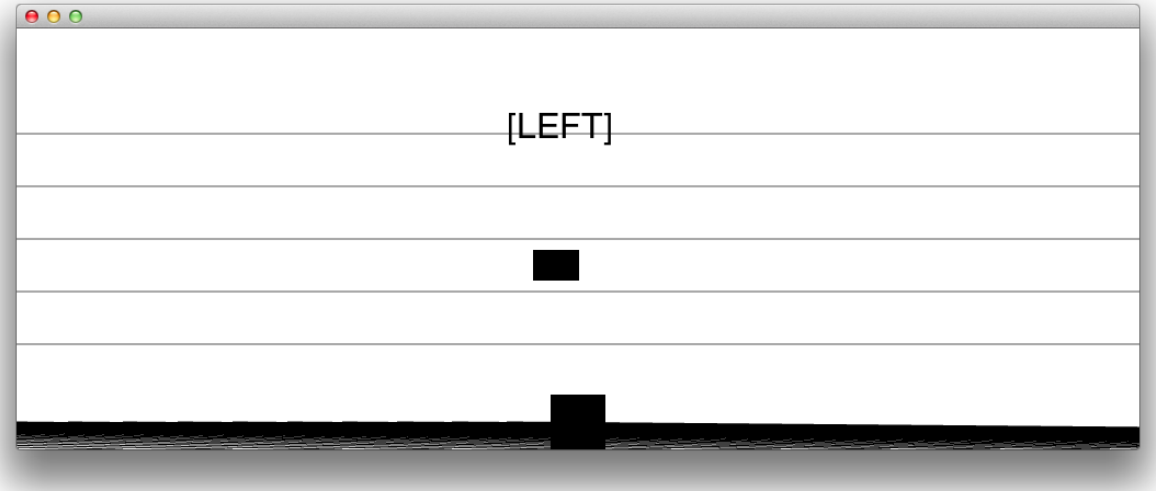

Fig. 17. That floating note wasn't there originally, but the prompt was too vague without context.

I presented this to Mauricio the next day for playtesting, round 2. The improved title screen was a success, and so was the new recording prompt! Mauricio quickly figured out he could interact with the note, and then when he reached the prompt to replay and saw a note appear at his press, that action made more sense too.

However, something else interesting happened. I assumed that once Mauricio realized he could record notes, he would realize that the note below the recording prompt was too high for him to jump on and thus useless to help him get over the obstacle, so he'd go look for a lower note and find the one earlier in the level and bring it back to the obstacle. He did eventually go back to the earlier note, but he didn't make the connection to carrying it back with him. Instead, he discovered that he could record the lower note, replay it directly adjacent, then record the replayed note and replay *that* directly adjacent, and repeat, thus moving across the screen horizontally without touching the ground. Like this:

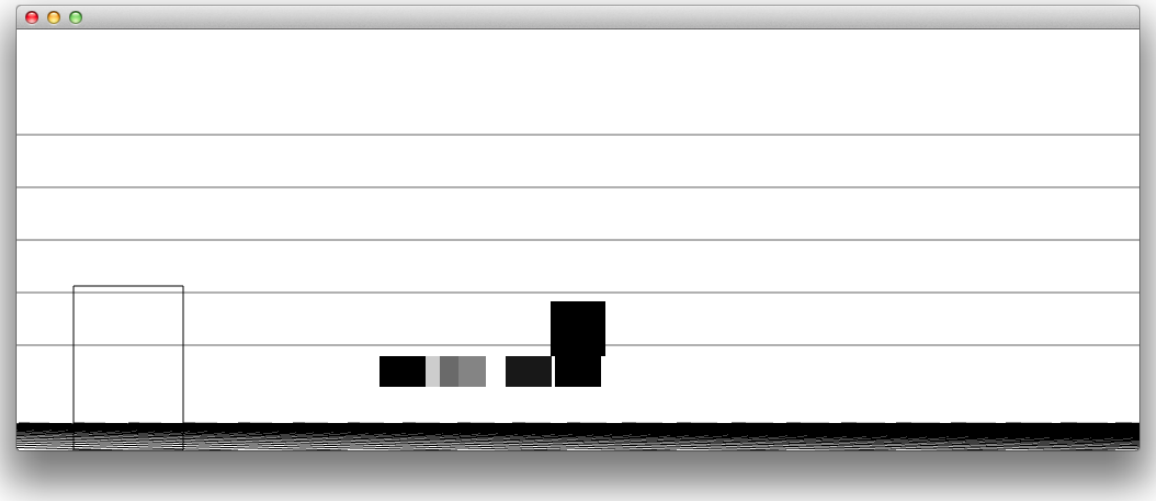

Fig. 18. Who needs the ground when you can keep recording and replaying notes?

This is a technique I had anticipated and planned on requiring the player to use--but as an *advanced* technique, to cross gaps or unsafe ground, *not* to move through regular parts of the level. However, when Mauricio reached the prompt to record (with the higher note), he jumped onto that, then repeated the same process, and so made it over the obstacle. In fact, using this technique, he passed across many of the puzzles without solving them in the way I had intended. It was fascinating and frustrating and funny.

This is all well and good, but also problematic because as you might imagine, it was very tedious for Mauricio to navigate in this way. Afterward I broke my code of playtesting conduct and explained that there was a much easier way to use the same mechanic of recording and replaying to get past the obstacles, and showed it to him. It was clear that, while I had solved the initial problem of context for recording, I hadn't solved the next problem of context for replaying.

Indeed, feedback continued to be a problem. Despite his new successes, Mauricio still felt like there wasn't sufficient clarity about the results of his actions. For instance, he recorded many notes without realizing what he was doing, then was confused when he expected to replay one note and a different note appeared. He didn't understand that he had a stored set of notes and was replaying them in order. Similarly, I later asked another friend, Alec Dawson, to playtest. Mauricio was there too and helped him. Alec figured out the recording and the playback much more quickly than Mauricio had and solved the puzzles the way I planned, but he too was perplexed by the lack of feedback, especially about stored notes and replay order.

After my final presentation, I did more work on feedback. A hollow rectangle follows the avatar and indicates the position of the next note to replay. Recording two or more notes generates dots on the avatar, one for each note in reserve. I haven't had a chance to playtest this yet but hope this will resolve many issues related to recording and replaying feedback:

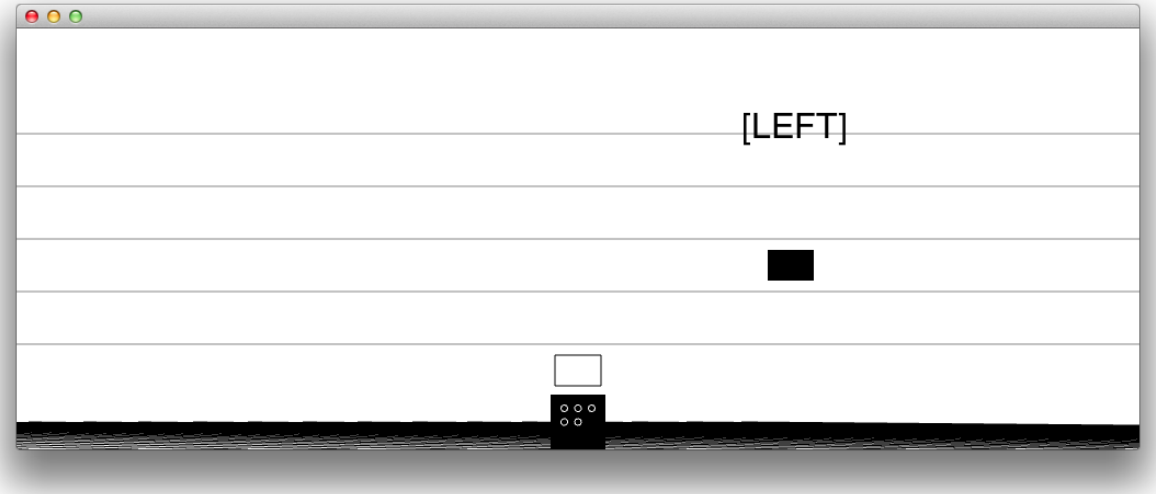

Fig. 19. The next note is a D above middle-C. There are five notes currently recorded.

At my final presentation I received extremely helpful feedback. The critics were more interested in the dynamism and musicality of the boss battle than they were in the puzzle-solving with its disjointed tones (although they didn't actually try the prototype). They encouraged me to push the musical side of the project more aggressively. I agree with this approach and believe I can use these existing systems to achieve it. On to the fifth prototype!