A Contextual Design Approach to Optimizing TunePad's Dual Mission in Music and Coding Education

By Lev Rosenberg

Section I: Pre-Interview Phase

Contextual Design

My UX research process was heavily influenced by Contextual Design (CD). CD is a structured user-centered design process created by Karen Holtzblatt and Hugh Beyer. It encompasses techniques to analyze and present user data, drive ideation from data, design specific product solutions, and iterate those solutions with customers/users. Holtzplatt's PHD in applied psychology brings a cognitive perspective to user-centered design, which I believe should be integral to any UX research.

Target Users:

In order to begin the interview process, I had to identify my target users. This Fall quarter, TunePad was taught to NU students in COMP_SCI 110. These students hoped to come away from the course having learned basic Python and CS skills, and weren't expected to have any prior musical experience. Thus, I decided that my target users were student programming learners without any musical background.

However, as TunePad is as much a coding literacy educational platform as it is a music-making tool, I wanted to examine both my target users—students without music experience—and students who happened to be musicians, in order to better understand the differences between the two and how they interact with TunePad.

*At the end of this report I will talk about this dichotomy between TunePad as a music-making tool vs. coding literacy educational platform.

Initial Hypothesis:

TunePad students in COMP_SCI 110 without a musical background would struggle to learn programming and music concepts simultaneously, decreasing proficiency in TunePad and coding literacy more generally.

Initial Analysis:

Before the UX interview process, I conducted a poll of the COMP_SCI 110 students regarding their comfortability level with TunePad and prior experience with music. With solely 60 respondents, nevertheless response analysis backed my initial hypothesis. Students were asked to rank both their musical expertise and comfortability with the TunePad platform on a scale of one to ten. Student's were

polled 4 weeks after first being introduced to TunePad. Those students that self-reported as music experts (7+ out of 10) reported a tunepad comfortability 22% higher than those who self-reported as TunePad beginners (1-3 out of 10). Specifically music experts reported an average of 5.6/10 TunePad comfortability whereas music beginners reported an average of 4.6/10.

Section II: Interview Method

Sampled Contexts:

Contextual Design is named as such because it asks designers to observe/interview users *in context*. Some contexts explicitly pertain to your target research area, while others are perhaps only tangentially related or even unrelated. Nevertheless, all contexts impact work practices, and thus it is important to determine which work contexts to sample explicitly and which to sample just to be sure you get some variation.

In my project, I used prior music education as my explicit context and graduation year as my non-explicit context. Why music experience? My project was looking into the effect of music experience on TunePad comfortability and programming learning. Why graduation year? I hypothesized that students with differing experience in college level courses might come in with different learning processes.

	No Music Experience	Some Music Experience	Music Expert	Total
Freshman	3		1	4
Sophomore	2		1	3
Junior		1		1
Total	5	1	2	8

You'll notice that I did not sample an even distribution across my two contexts, as I ended up with a higher concentration of lower-classmen without music experience. This was for two reasons:

1. The makeup of COMP_SCI 110 is largely underclassman. I wanted to sample an accurate distribution of graduation years to the course.
2. My target user was the inexperienced musician CS learner, I decided that having a larger number of students without music experience was important to my research.

The Field Interview:

The challenge in UX work is getting to that level of detail about work that is unconscious and tacit. So, I used the contextual inquiry technique outlined in Contextual Design. Using in-depth observation of a small sample of users and their work practices and behaviors, contextual inquiry allows designers to go out into the field and talk with people about their work while observing them. The general concept being: knowing that users are often unaware of their work processes, if designers watch users while they work, the users do not have to articulate their own work practice. In essence, field interview data-gathering overcomes the difficulties of discovering tacit information.

Each interview followed the same general structure: Introduction → Observation → Wrap up.

Introduction:

In the Introduction I began by introducing myself and my project focus. Then, I asked interviewees basic information about themselves—name, grade, major—in addition to an overview of their past experience with music and programming. I then introduced the main event of the interview: User's were complete a simple assignment in TunePad. Here was my general script to introduce the assignment:

Observed Assignment:

- "Your goal for this session is to create an original, short musical composition in TunePad. By short I mean something ~16 beats, but you could make something shorter or longer as long as you are putting in effort.
- Your final composition should include at minimum 2+ instrument cells, which could contain anything from chords, rhythm, bass, melody—whatever inspires you to be creative.
- I would encourage you to use those coding techniques we learned in class—loops or functions—to help you compose, but I don't want you to focus on artificially adding those into your code in order to fulfill an assignment requirement. I want this to be an organic TunePad composition.
- Please do any of the tasks you normally do when working with TunePad. You can play music for inspiration, check canvas, look at a tutorial, campus-wire, or the internet, anything you might do normally when completing a TunePad HW assignment you can do here.
- I'll be observing you and I may stop you when I see something interesting and ask questions. But I'll try not to interrupt too much so that you have the time to make a project you enjoy and are proud of"

Observation:

While user's completed this assignment, I observed and took notes on their process. My general goal was to "look beyond the task" as CD would call it. I wanted to identify tasks that users' completed, but more importantly note how each task fit into a larger story—focusing both on individual tasks and larger overall processes.

I took special note of:

- Alternate ways of performing tasks—instances where the user deviates from the norm
- Artifacts that the user uses or refers to
- Breakdowns in the user's work—what doesn't work for the user
- What works and doesn't work in the tools of TunePad that the user uses.
- Which functions of TunePad aren't used
- UI level details inspired by the user's work
- Questions I had about the user's work

Wrap Up:

After each observation session, I would discuss with user's what they thought of their process in completing the assignment. First, I would ask them for high level interpretations—were you satisfied with what you created. Then I would ask them for more specifics—what went well, what didn't go well. I also wanted to collect retrospective accounts of work processes. I would ask user's to compare their experience in the interview to past times they worked with TunePad, mainly HW3. Finally, I asked the user to answer those questions I wrote down in the observation. After the third interview, I began also asking users about their experience with the Programming Reference section of the platform.

Section III: Post-Interview Phase/Analysis:

User Personas and Affinity Notes:

After each interview, I would spend about 1-2 hours compiling my notes into user personas and affinity notes, as outlined by the CD process.

Each write-up of user affinity notes is color coded by category.

- Coding Process Notes
- Music Process Notes
- Design Ideas
- User quotes
- Use of artifacts
- Problems in the work
- Lev's Interpretations of work
- Cross-User Habits

For these in depth interpretations of the individual TunePad users sampled in this study, refer to the following google folder.

Cross-User Habits:

All observed users:

- Changing instruments:

    - Students spent significant portions of their interview playing cells many many times with different instruments to find their preferred sound.
- Clicking on the keyboard interface:
    - Students would use this interface all the time to create their own melodies and try out different pitches and drum sounds.
- Referring to other TunePad compositions.
    - The most common artifact referred to by students was their HW3 TunePad project. Every user I interviewed looked at their HW3. The second most common artifact was the TunePad community website.

Music Beginners:

- Confusion regarding meter and 4/4 time. *See design suggestion below*
    - Music beginners would routinely write code with unexpected beat lengths:
        - Ex. beats = 0.42 or beats = 0.85.
- Confusion regarding how cells repeat.
    - Music beginners did not understand that a cell 4 beats long will repeat itself to match another cell 16 beats long
- Heavy reliance on outside artifacts to inform melody and harmony (chords). *See design suggestion below*
    - In addition to looking at HW3, music beginners would look up TunePad tutorials, online chord videos, etc. for harmony help.
- Excessive time:
    - On average, the amount of time spent making melody and chords sound "good" was longer than it was for music experts.
        - Outside artifacts (Tunetorials) helped remedy this problem.
    - On average, the amount of time spent making rhythms line up was longer than it was for music experts.

Music Experts:

- Music theory:
    - The bulk of the composition is informed by previous music theory knowledge.
- More code:
    - Music experts write more code per cell, they utilize more cells, and the code is often more complex.
    - This is likely due to the fact that making music "sound good" doesn't take as much time for experts. So they can spend more time working on other areas.

Notable User Quotes:

- On music theory informing composition:
    - "You know the circle of 5ths will sound right…ii→V→I will sound better than ii→V→VI"
- On rhythm:

    - “[the hardest part of TunePad for me is] getting the beats correct. You can mess around with notes, but understanding beat lengths is more difficult”
- On TunePad as a whole:
    - “You can hear how you program. It's better if it sounds better”
        - This was said in a positive context, it is what they loved about the platform. But that's not a good outlook if TunePad is used as a coding literacy educational tool. Good code could sound awful and bad code could sound great.
- On the simplicity of music:
    - “My HW3 was hard because I tried to make it too complicated. It didn’t really occur to me that simple beats were actually easy. All we had seen in class was the professor recreating drake. In class I saw what other kids did in the showcase and realized that making code sound good can be simple.”
- On the Programming Reference tab: *See design suggestion below*
    - “The help button is a misnomer. Clicking help isn’t usually useful on a page”
    - “Play a note with a pitch greater than 0, I don’t know what that means”
    - (in response to my asking a user where they think the help button will lead them to) “I’m not sure. Maybe an email to the creators of TunePad?”
    - “I wish that the helper guide was more like the textbook. The textbook is like English, whereas the help guide is more advanced/concise. It’s helped me on some things, but not a lot”.

Section IV: Design Suggestions

Chords

Most beginners struggle with harmony. While online tutorials proved helpful in getting chords in cells, using external artifacts is unnecessary when TunePad could provide help itself.

Knowing that all users loved clicking on the keyboard interface. I thought TunePad could replicate that interface with diatonic chord buttons

Here is an example of what a new chords feature might look like.

To note:

- Chord buttons correspond to the seven diatonic chords in a major scale, represented by roman numerals.
- Clicking on a chord button plays that chord, similar to clicking on an individual note.
- Hovering over a chord button will produce example code for playing that chord.
- The feature can be toggled on and off so that it doesn’t need to take unnecessary vertical space when not used.
- The key of the entire project would be the key that the chords correspond to.
- Additions to the feature I didn’t include:

- Options to include chords in different modes/scales than major/Ionian.
- Applying the feature to guitar and bass as well.

TunePad Programming Reference Section

Many, many users noted that the TunePad programming reference section is confusingly placed, written, and/or organized. Here is an example of what a restructured reference section might look like.

From this video, you will notice that The reference section would be split up into various sub-sections. I believe that this organization will make the reference section more manageable for beginners, providing the necessary information.

- Notes & Rests
- Time & Rhythm
- Python Documentation
- Audio Effects
- Modules
  - Chords
  - Constants

Many users commented that TunePad documentation was confusing. I also noted in tutorials and office hours as a PM for 110 that students struggled a LOT with the concept of parameters. So, I think TunePad function definitions should be split up into:

a) An overall description of the function
b) Specific descriptions of individual parameters

Below is a sample of a redefined playNote().

playNote(note, beats = 1.0, velocity = 100, sustain = 0)

playNote is a fundamental building block in TunePad. It is a function that produces a pitched tone corresponding to some parameter input **note.**

Keyword Parameter/s:

note: sets the pitch of the note played. velocity can be any number from 0 (lowest pitch) to 127 (highest pitch). You can also call playNote with a list of notes to play multiple notes at once.

Optional Parameter/s:

beats: sets how long the note will last. Larger values make the note last longer and smaller values shorter. Without a beats input, beats will default to a beat-length of 1.

velocity: sets how hard/loud the note sounds. velocity can be any number from

0 (quietest) to 127 (loudest). Without an input, velocity will default to a volume of 100.

sustain: allows a note to ring out longer than the value given by the beats parameter. The value of the parameter is the number of beats the note should sustain. Without an input, sustain will default to a beat-length of 0.

Rhythm:

I noticed that many beginner musicians did not understand beats. They would often provide beat-lengths such as 0.86 or 0.1, as opposed to 0.25 or 0.75. These unexpected beat lengths would sound “wrong” to users. But they didn’t know how to fix it, or even that the beat-lengths were the culprit. Consequently, they would spend far too long attempting to figure out rhythm instead of writing more complex code. I thought of two design possibilities to remedy this, autocorrect and tick marks.

Autocorrect possibilities:

1. **Predictive rhythm:**

With predictive rhythm, TunePad would provide 1-3 predictions for what the beats parameter might be for a given playNote, playSound, rest, or other function that takes a beats input.

Such a feature would take into account the meter of the project, and the current location of the playhead in a measure. Ex. In a project in 4/4 time, if the playhead is on beat 3.5, it might suggest 0.25, 0.5 and 1.5.

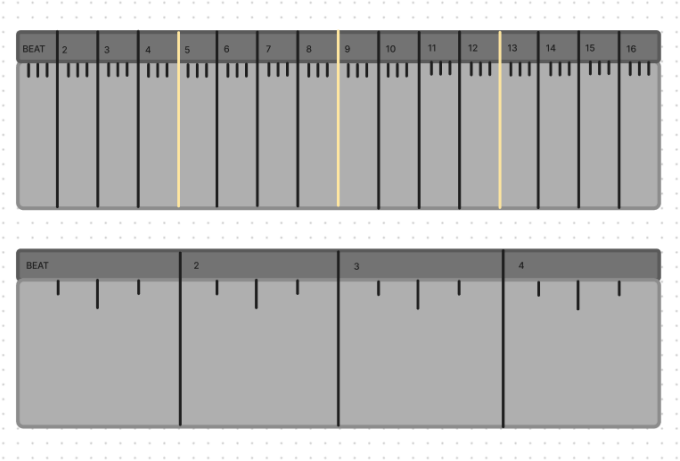
I imagine these predictions would be located between the instrument-toolbar and editor-container divs (underneath the editor).

2. **Corrective rhythm**

With corrective rhythm, TunePad would underline unexpected rhythmic elements in red, and provide specific corrective choices to fix it.

Should a user attempt to type in a beat length that is not a multiple of 0.25 or 0.33, the integer would be underlined in red. Then, TunePad might suggest alternatives that *are* multiples of 0.25 or 0.33 that take into account meter and playhead location. Should a user intentionally use an unexpected beat-length, they could simply click an ignore button (similar to corrective text in google docs).

Tick Marks:



Tick marks would provide a visual representation of how beats *should* line up depending on the meter. Visual elements include:

- Highlighted downbeats for cells 16 beats or longer.
- Varying tick-lengths for 8th and 16th notes in shorter cells (4-8)

Section V: TunePad mission clarification:

TunePad is new. It is in a state of constant flux and will likely continue to be for some time. As a result, no one seems to have a solidified concept of the purpose of TunePad. As it exists now, there seem to be two options for TunePads mission:

1. TunePad hopes to teach coding literacy to beginner programmers
    a. Coding literacy is the main goal of the platform, with music-making as an incentive.
2. TunePad hopes to provide a music making platform for musicians with some programming knowledge.
    a. Music-making is the main goal of the platform.

I believe that it is possible for TunePad to exist in both realms—it could be a platform meant for musician programmers and CS learners. However, if this is the case TunePad must have features to support both users.

Music beginners need more scaffolding for TunePad to work as a coding literacy educational tool. This report serves as the reasoning behind, and possible designs for what that scaffolding might look like.

On the other end, I want TunePad to exist as a platform for music making. I think the nexus between coding and art is important and fascinating. There's a reason the acronym STEM is being replaced with STEAM in some schools. However, as a musician myself, I am unsure *why* I would use TunePad over

another music-making platform. I think further research needs to be done to see how coding provides a unique outlook on music-making that other platforms cannot provide.