

Rationale to

“Enjoyable Block Game”



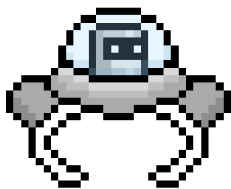
0. Introduction

My project consists of a small game prototype, hopefully titled appropriately as “Enjoyable Block Game” that seeks primarily to develop an interesting relationship between action and puzzle gameplay through an adaptive soundtrack system. More specifically, I aimed to create a soundtrack system that not only reflects the dichotomy of action and puzzle gameplay, but adapts to and is influenced by the state of the game in order to improve the player’s experience of *agón*. I developed the game using [Godot Engine](#), and composed the soundtrack using [LMMS](#). Prior to this project, I hadn’t ever implemented an audio system in a game more advanced than just looping whole tracks, so it was an interesting experience learning the ins, outs, and inevitable difficulties involved in working with game audio in a close manner.

Note about the included game demo: For ease of use, all the players are mapped to the keyboard, so the controls described below are slightly different from the ones used in the version of the project included. I’ve outlined more specifically how to use it in the included README.md file. If the game ends up being too buggy, there’s also a video showing it in action.

1. Gameplay Design Overview

This project was designed to incorporate both “puzzle” and “action” gameplay mechanics. As such, it emulates an archetypical block-placing mechanic from games in the *Tetris*-like category such as *Lumines* simultaneously alongside simple action-platformer mechanics, for the “blockmaster” and “climber” players respectively. Each player type is described as follows:



Blockmaster: The blockmaster’s job is to move and place pieces in such a way that the climber can reach the trophy in the center of the stage. In order to accomplish this, pieces can be moved left and right in one-block units, rotated, and dropped straight down -- unlike *Tetris*, the pieces do not fall on their own, and cannot be maneuvered once they begin to drop.

Keyboard	Gamepad	Action Phase	Puzzle Phase
A / D	Left Stick	Move piece left / right	Change upgrade
W	Nintendo B	Drop piece	Select upgrade
J	Nintendo Y	Rotate piece	N/A



Climber: The climber’s job is to platform up the pieces dropped by the blockmaster and win a round by touching the trophy. Alongside moving left and right, the climber can jump (initially to the height of one block) and attack, which serves as a means to break blocks (so the climber is never truly stuck) and knock back the other team’s climber.

Keyboard	Gamepad	Action	Puzzle Phase
A / D	Left Stick	Move climber left / right	Change upgrade
W	Nintendo B	Jump	Select upgrade
J	Nintendo Y	Attack	N/A

The game takes place in a round-based first-to-five match format. Each round consists primarily of the game’s action phase, the main gameplay setting in which teams compete to take the trophy.







Action Phase

Puzzle Phase

Between rounds, a puzzle phase also takes place -- here, each team strategically selects upgrades for both their climber and blockmaster players. Here, I was trying to capture the puzzling, cost-benefit decision scenarios found in many powerup-centered roguelike games like *Risk of Rain* and *The Binding of Isaac*. Some examples of upgrades and their effects are the following:

Blockmaster	Climber
Timer - Decreases the cooldown period before a new piece to place spawns.	Speed Potion - Increases the movement speed of the climber

 Stairs - Adds a new helpful piece type to the pool	 Wings - Allows the climber to perform a second midair jump
 Thrusters - Increases the speed at which pieces can be moved left/right	 Syringe - Increases the attack speed of the climber

With this, not only does each team consist of a player representing individually action and puzzle gameplay, but the game as a whole cycles between a more fast-paced action phase and a slower, more strategic puzzle phase. My hope was to create an environment that would allow for a soundtrack to frequently represent the state of interesting combinations of action and puzzle elements within the game in various ways.

2. Soundtrack Design

With this gameplay format, the relationship between the action and puzzle aspects of the game can fluctuate in interesting ways, especially when taking into account the relative activity level of each action and puzzle player (for example, is the climber moving around a lot, are pieces being maneuvered rapidly, etc). For example, there may be points where the game is in its action phase, but the puzzle player is more active -- my goal was to have the soundtrack reflect specific state like this, and especially communicate the push and pull between action and puzzle elements within the gameplay varying with the “intensity” of the game. I ended up having to consider many approaches in order to approximate this effect, but ended up with a relatively robust system that accomplished most of what I initially intended.

Soundtrack Vector Model

The main way I conceptualized the behavior of the soundtrack was thinking of its state as existing at a position between a few axes. They were as follows:

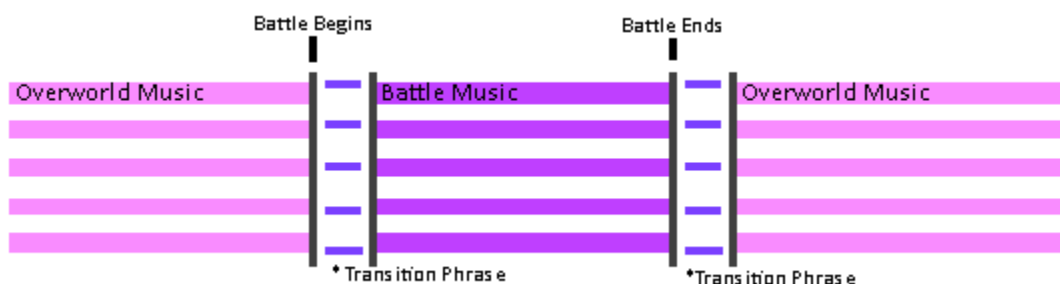
- 1. Puzzle vs. Action Phase:** Represents the phase of the game. Binary, either -1 or 1.
- 2. Puzzle vs. Action Activity:** Represents relative activity levels of action vs. puzzle mechanics. Ranges from -1 (only puzzle) to 1 (only action). Decays over time to 0.
- 3. Intensity:** Represents the intensity of the game. Ranges 0-1.

So, for example, the vector [-1, 0.4, 0.75] would indicate the game is in its action phase, the climber players are more active than the blockmasters, and the game is in a pretty intense state. From that information, the sound engine would be able to determine the soundtrack’s behavior. In this example, it would be sequencing sound modules tied to the action phase, consider reflecting an increase in climber behavior, and draw from sound modules that reflect a high level of intensity. The specific approaches through which the soundtrack ended up adapting to each aspect of the vector are explained in the following sections.

Puzzle vs. Action Phases in the Soundtrack

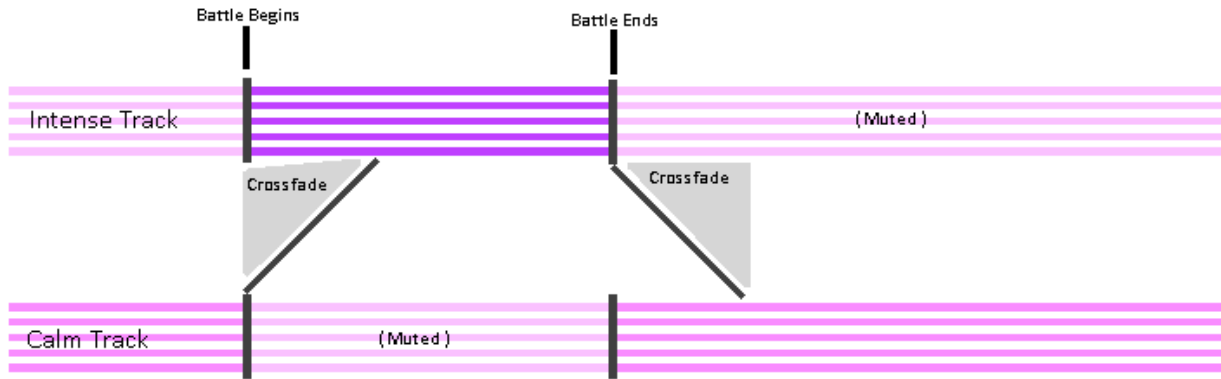
The most noticeable way in which the soundtrack in my project changes is how it adapts based on whether or not the game is in the “puzzle” or “action” phase -- Or, within the sound vector model described above, whether the first value is -1 or 1 (and what to do in moments where it notices that it changes). I looked into a variety of games with soundtracks that could be similarly conceptualized in order to identify general techniques and figure out how I wanted to design this behavior of the game audio.

1. **Two-Piece Hard Transition Method:** While not usually considered “dynamic,” many RPG games like *Pokemon* and *Zelda II: The Adventure of Link* reflect two game states -- in these examples, overworld and battle states -- in the player’s aural experience by playing music drawn from two disjoint collections. In *Zelda II*, for example, upon encountering an enemy in the overworld, the following sequence of events occurs: the overworld track hard stops, a transition melody is played, and then a battle track plays from the beginning. Likewise, when a battle area is left, the music stops again and shifts back to playing the overworld theme from the start.

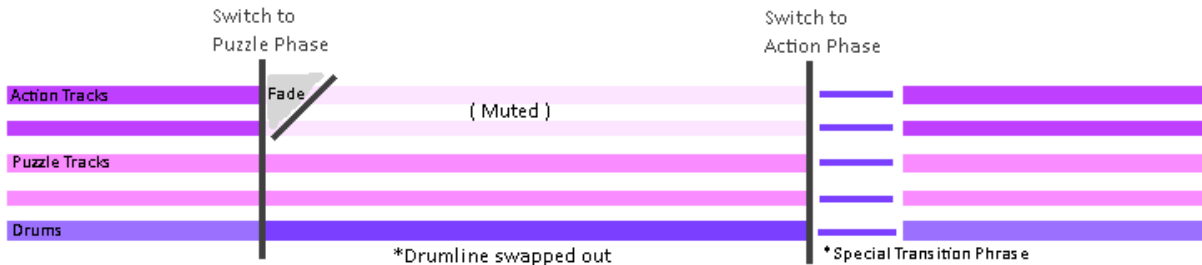


2. **Two-Track Simultaneous Method:** This is probably the most common way a game soundtrack dynamically reflects the oscillation between two states, as featured in games like *Fire Emblem: Three Houses*. As a specific example, newer Fire Emblem games have background music that switches between two tracks during the bulk of the gameplay: one “calm” track for when the player is panning over the game’s chess-like map, and one “intense” track for when two units clash in a battle (referred to as “Rain” and “Thunder” variations).¹ These tracks are usually completely distinct in instrumentation, but contain the same melody and are synced up alongside each other -- so, switching between the two is as easy as crossfading between them. I’ve attempted to summarize the behavior of a dynamic soundtrack that uses this system below:

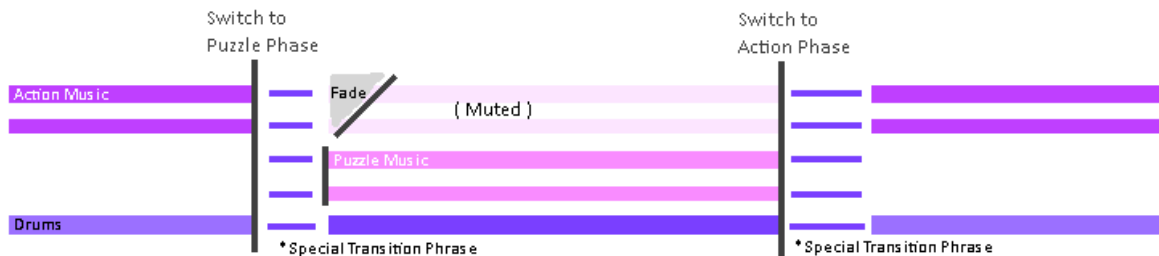
¹ “List of Music in Fire Emblem: Three Houses.” Fire Emblem Wiki. Accessed May 23, 2021. https://fireemblem.fandom.com/wiki/List_of_Music_in_Fire_Emblem:_Three_Houses.



3. **Multi-Track Splicing Method:** While I was previously familiar with the soundtrack to *Paper Mario: The Origami King*, I hadn't known specifically about its dynamic properties before it was brought up in class. The game's soundtrack, much like its battle system, segues between action and puzzle phases in very clever ways depending on even the beat of the measure each transition occurs.² Unlike the previous examples where the music playing just completely changes, here only specific instruments fade in or out, the drumline changes, and special care is taken to make these transitions natural -- including the addition of special transition phrases from the puzzle phase to the action phase depending on the beat of the measure the transition is triggered. In this game, the dynamic soundtrack thus takes the following more elaborate form:



For this project, I decided to combine different aspects of these approaches in order to find the best fit for the gameplay. I ended up with a system akin to the following diagram:



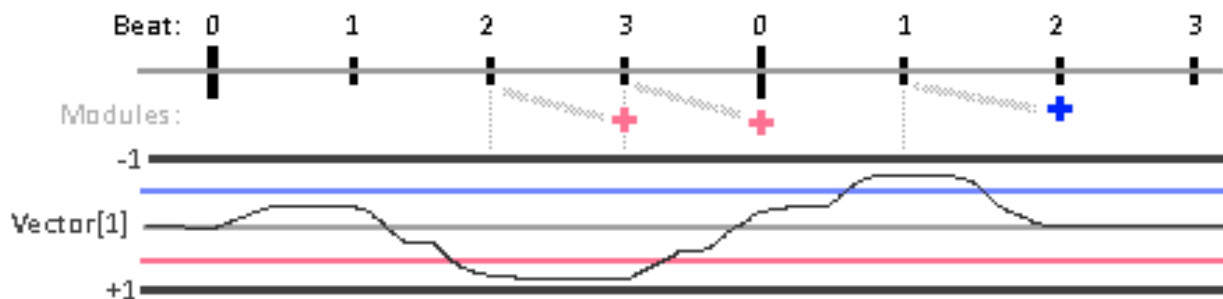
By separating the puzzle music from the action music, I was more easily able to allow the action music to reflect aspects of the action's phase's gameplay (as described in the following sections) within the time constraints I had to complete this project. In order to blend these more disparate tracks together, I was

² "Paper Mario: The Origami King Doesn't Cut Corners." 8-bit Music Theory, September 30, 2020. YouTube video, 17:57. <https://www.youtube.com/watch?v=i13K77BAL50>.

able to emulate the beat-based transition phrases of *Origami King* upon each phase transition. By keeping track of beats and measures during gameplay, upon a transition trigger, the sound engine is able to queue up and play an appropriate transition phrase for the next beat, and then switch between the action and puzzle tracks as soon as the current measure is completed.

Puzzle vs. Action Activity Sound Modules

The way the relative levels of puzzle and action activity, or relative activity levels of the blockmaster and climber players respectively, are communicated through the soundtrack is primarily through a procedurally generated sequence of accompanying sound modules to the soundtrack. The goal here was to emulate a pattern of “modular combinations” as described by Medina-Gray.³ Similar to the behavior of Medli’s harp in *The Legend of Zelda: Wind Waker* when it overlays the Outset Island background music, a randomly-generated sequence of sound modules overlays the soundtrack to this project with the intent of conveying which of the two player types is more active.



With this mechanism, when the value of the puzzle vs. action activity component of the sound vector is past a threshold in either direction at the start of a beat, a corresponding sound module (either puzzle or action) is queued to be played on the next beat (or, for two-beat modules, the next even beat, etc.).

Something I initially struggled with was giving these modules “meaning” -- akin to how the module system in *Wind Waker* expresses Medli’s studious nature and developing skill, a theme central to the game’s story and thus an important element of the player’s experience. I ended up trying to express meaning by tethering the module to each player’s respective speed and direction. The sound modules associated with the climber have a generally quicker, ascending melodic pattern, and the instrumentation was designed to have a more plucky timbre. The blockmaster modules contrast by containing descending melodies, lower-pitched instruments. In this way, I’ve attempted to inspire a reflection of the ascending nature of the climber’s action-platforming and the steady, descending motion of the blocks dropped by the blockmaster in the soundtrack with these modules. I was only able to implement this aspect of the adaptive soundtrack during the game’s action phase.

³ Donnelly, K. J., William Gibbons, Neil William Lerner, and Elizabeth Medina-Gray. “Meaningful Modular Combinations.” Essay. In *Music in Video Games: Studying Play*. New York, NY: Routledge, 2014.

Reflecting Intensity Within the Soundtrack

In this project, the “intensity” of the game is recognized by the sound engine in two ways: the height of the highest player during the action phase, and the number of rounds each team has left before winning. Basically, if one of the climbers is above $\frac{2}{3}$ of the way to the trophy’s height level, the intensity component of the sound vector begins to ramp up and increase -- if neither is, it decays. Alongside that, if it’s match point for at least one of the teams, the intensity value of the sound vector exists at a higher baseline resting state. Based on this intensity value, the sound engine tweaks the soundtrack in a few ways. The primary method in which intensity is reflected is in the drum line -- at any given time, both a “mellow” and “intense” drum track are playing simultaneously, their relative volumes automatically adjusted and inversely related to the intensity value. So, at higher intensities, more complex and fast-paced percussion is highlighted in the soundtrack. Furthermore, the intensity level plays a role in determining the climber and blockmaster sound module pools as described in the previous section when the intensity is above a certain threshold. These modules are only subtly different, but layered with more sound effects.

Agónic Infusion in the Composition

As I often forgot while working on this project, I also had to actually compose music for the game alongside designing the audio system. To gel with the pixelated graphical style, I used the NES sound plugin *NEScaline* and GameBoy plugin *FreeBoy* that come built into LMMS to create most of the instruments with a retro style, but backed it up with some various synthesizers in order to add some reverb and depth to the sound profile. Since the game is a two-versus-two competition, and especially since I was capturing an almost competitive relationship between the game’s equal action and puzzle elements, I wanted to inspire Caillois’ idea of the *agón* aspect of play⁴ in the musical composition for the action phase. In order to do this, I tried my best to emulate the energetic drum style found in *agón*-esque games like *Super Hexagon* (good example here is the track [“Focus”](#)), and emphasized arpeggiated melodies in order to approximate a frenzied feeling, such as is found in tracks like [“Guardian Menace”](#) from *Hyrule Warriors: Age of Calamity*. For the puzzle music, I drew inspiration from the *Tetris* A and B themes, specifically the NES version, but also tried to include some of the more retro-futuristic synth background elements found in newer puzzle games like *Lumines* (for example, as is present in the track [“Fly Into the Sky”](#)).

3. Reflections & Conclusion

Difficulties and Setbacks

I learned a lot about the implementation of game audio during this project. I knew beforehand I was going to underestimate the complexity of designing such an audio engine, but was still surprised with many of the difficulties I encountered that I hadn’t previously considered.

⁴ Caillois, Roger. *Man, Play and Games*. trans. Meyer Barash, Urbana, IL: University of Illinois Press. 2001.

One such difficulty lied with accurately timing playback of procedurally generated sequences of sound modules. Within the game engine I was using, I used the in-game time along with the BPM to determine measure and beat timings -- but, since this updates after small but stochastic intervals of time, it wasn't a truly precise estimate. Ninety-nine percent of the time, this approximation was smooth enough that things flowed well. But, if the game happened to lag, a thing like a "ghost rest" would occur when certain modules were played a fraction of a second too late and were cut off a fraction of a second too early on the next interval. I ended up having to cut some corners to combat this effect. For example, while I originally wanted the percussion to be sequenced in real-time, small disturbances in its timing were extremely noticeable -- so, I ended up distancing that part of the soundtrack from the module-sequencing system and relegated it to looping audio tracks in the background. I wasn't able to find a perfect solution to this roadblock, but due to its rarity, it didn't pose too much of an issue on the whole.

I also learned, yet again, that it can be resource-intensive to operate with too many sound files at one time. Due to the combination of the game engine I used and the relatively middle-spec laptop I developed on, having too many things either loaded playing at once could cause lag spikes in the game. Not only did this feed into the small module offset effect, but it interfered with the fluidity of the gameplay. Ironically, in trying to achieve my initial goal of the gameplay determining the form of the soundtrack, the soundtrack's technical overhead ended up influencing the pace of the gameplay in an adverse way. I had to rework a lot of code and some design decisions on how the sound engine would adapt in order to ensure the game operated as intended.

Finally, it was surprisingly hard to "test" the soundtrack, or ensure the largely non-deterministic soundtrack sounded "good" at a baseline level to most of the playtesters I demoed the game with. While the "goodness" of music is really subjective, I didn't want it to stand out as too dissonant or cacophonous and instead have the player's focus be drawn to the sequence of modules. I primarily tweaked some of the aspects of the music by asking for feedback on how *agon*-ic (after providing a brief definition) the audio experience felt. I spent a lot of time reworking the percussion, sound modules, chord progressions, and puzzle phase music with regards to this -- initial drafts of the soundtrack ended up sounding more like the muses of an *Electroplankton* species rather than a backing track to a competitive game.

For the Future

Having spent a month working on this project, there are plenty of areas I would want to improve given more time, and things I wish I could have done slightly differently from the start. A few of the major ones are as follows:

1. **Reflect another form of *agón* by tying sound modules to each team:** For example, assigning the blue team modules in a major key and the red team the minor key, and let the dueling keys reflect the dueling teams in the gameplay.
2. **Expand sound modules into more meaningful melodies:** With the current implementation of the audio engine, each sound module can only be a measure long -- enough to make an impact, but a hard limit on the system's flexibility. Allowing for variable lengths of modules would be a straightforward way to increase the degree to which meaningful melodic combinations can be created, and allow for more formal melodic phrases (which may be more easily able to inspire feelings of *agón* in players) to be implemented.

3. **Find more unique and significant ways to represent the “action” vs. “puzzle” dichotomy:** Currently, the sound engine keeps track of the skew towards greater puzzle or action-related activity by receiving signals when players perform actions -- which, having now implemented it, I recognize is a really loose estimate of this phenomenon. For example, the climber can spam punches in the corner much faster than the blockmaster can spam blocks, which makes it too easy for the sound engine to be dominated by action-related activity. Furthermore, this system doesn't capture the “significance” of each action -- so even if the blockmaster is carefully creating the perfect path upwards, they can be overwhelmed in soundtrack representation.
4. **Experiment with more iMUSE-esque instrumentation swapping:** One idea that I didn't have time to implement was let the presence of certain upgrades change the instrumentation of the music -- similar to how the melodies of tracks stay. For example, the climber possessing the “sledgehammer” upgrade could swap out the kick drum for a more metallic hammer clunk. I never prioritized this because it would be a lot of work (I definitely got a little bit carried away with the number of upgrades in the game) and doesn't really contribute to representing the specific balance between the action and puzzle gameplay, but would be a cool touch and contribution to the theme of having the soundtrack closely adapt to the gameplay state.

Conclusion

Overall, I found this project as interesting as I expected, with satisfying highs to accompany the sometimes frustrating low points. It was certainly time-consuming, but I'm proud I was able to cut myself off and deliver a functional, playable game. I'm also usually pretty self-conscious about showing off music I make, so through getting playtesting feedback I was able to break out of my comfort zone a bit as well. I definitely plan on developing this game out a bit more over the summer, so I'm looking forward to taking this project even further and addressing some of the shortcomings identified above!

Bibliography

Readings:

1. 8-bit Music Theory, “Paper Mario: The Origami King Doesn't Cut Corners,” September 30, 2020. YouTube video, 17:57. <https://www.youtube.com/watch?v=i13K77BAL50>.
2. Caillois, Roger. *Man, Play and Games*. trans. Meyer Barash, Urbana, IL: University of Illinois Press. 2001.
3. Donnelly, K. J., William Gibbons, Neil William Lerner, and Elizabeth Medina-Gray. “Meaningful Modular Combinations.” Essay. In *Music in Video Games: Studying Play*. New York, NY: Routledge, 2014.
4. “List of Music in Fire Emblem: Three Houses.” Fire Emblem Wiki. Accessed May 23, 2021. https://fireemblem.fandom.com/wiki/List_of_Music_in_Fire_Emblem:_Three_Houses.

Soundtracks:

1. Chipzel. "Focus", Retrieved May 23rd, 2021 from <https://www.youtube.com/watch?v=wD3j1O1XHQY>
2. Nakamura, Takayuki. "Fly Into the Sky," Retrieved May 23rd, 2021 from <https://www.youtube.com/watch?v=p4hUORohJb8>
3. Kumi Tanioka, Reo Uratani, Ryotaro Yagi, and Haruki Yamada. "Guardian Menace," Retrieved May 23rd, 2021 from <https://www.youtube.com/watch?v=VlXooRDZdt4>