Group Members: Stephen Pupa, George Whitfield, Shreya Ramesh

Hit It!

A rhythm game

Scope

Due to the current global pandemic limiting the opportunities for in-person and communal interaction, certain genres of games which require additional hardware have become limited in their availability. For example, two famous rhythm games, those being DDR and Beat Saber, require potentially expensive and hard to acquire hardware setups in order to be enjoyed in a home environment. Games lacking hardware often involve minimal movement, often just involving the tapping of a few fingers. As a solution to both issues, our game "Hit It!" aims to be a middle-ground between these two popular mindsets, as its minimalized hardware setup will hopefully undercut the exclusivity of more expensive game setups while still enabling audiences to be more actively engaged through movement.

To be more specific, "Hit It!" is a rhythm game where the player drums on a set of drum pads to the beat of a user-specified song. Our system will consist of a tabletop device capable of connecting directly into a computer via a USB connection. From this point, there will be a software application that interprets the user input through this connection and displays the relevant information to the user through a GUI depicted on the computer's monitor.

Requirements

Part of our game's functionality will be the ability to provide any song to the game, at which point it identifies the beats and/or melody of the song and maps that to a format playable within the game. The primary challenge of this will be the accurate tracking of those beats within the user-inputted song, which we are hoping to achieve an accuracy of at least 80% through our script. The game will also require an engaging GUI to display for both the song selection and gameplay portions of the project. One concern regarding the integration between the digital signal processing, the game design, and the hardware components is latency (delays). For optimal performance and user enjoyment we wish the maximum latency to be no more than 80 ms between the user providing their hardware input to the point at which the GUI displays the outcome of said input.

Regarding the hardware setup, our current expectation is to have four drum pads that the player hits with drumsticks to play the game. For the limitations on this hardware, the above specified latency is probably the largest concern, however, other factors such as accuracy in recognizing hits to the drums pads is also of extreme importance. We currently are not aware of how much force is represented in a "normal" hit to a drum pad, but we would hope that with some reasonable application of force we will be able to recognize at least 99% of hits to the drums while also maintaining little to no structural damage to the hardware system. Additionally, the size of this system will be a factor influencing the availability to audiences, so keeping the system within a reasonable space, likely small enough to fit on a desktop, will be necessary. We are not currently aware of what dimensions would be optimal for this, but for now we have the goal of a 3 ft by 1.5 ft rectangle that also is no more than 4 inches in height. Since each drum pad will likely have a modularized design, I believe a good value to aim for would be a 6 inch x 6 inch footprint for each of the 4 drum pads on the board. This hardware setup should be less expensive than a modern VR headset and smaller in size than a DDR platform, while also allowing for a more physically engaging experience than a keyboard controlled rhythm game.

Implementation

The above description of our system has an information flow as follows. Firstly, user input is provided via the drum pads. The drum pads will utilize piezoelectric-based sensors or pressure sensors that will feed into a PCB mounted microcontroller so that we can turn that information into an electrically readable form for the software. These will allow both for the user to navigate menus and select songs, as well as to actually play the rhythm-based portion of the game. Also, these pads are expected to take a beating through normal play, so they likely will need to be made of a dense material to ensure they do not lose their shape or deform over extended periods of use. From this point, the microcontroller will pass the information over a USB connection to the computer, at which point our specially designed software will enable the user to launch and play the game. This engagement with the game will be primarily represented through our GUI, which as previously mentioned will be programmed within C++.

Additionally, another feature of this project is the ability to take a user provided song and create a beat mapping of that song so that it can be played within our game. For this digital signal processing, we currently plan on using python as the platform to generate timestamps and lengths of the beats used for this portion of the project. Other components, such as beat mapping of melodies, are being considered for higher levels of difficulty.

Testing Plan

Above, we have listed a total of four different quantitative margins for evaluating our project.

For starters, we set the goal of 80% accuracy within our python script for transforming user provided songs into playable tracks within our system. Since a “correct” track is somewhat hard to evaluate, we will likely transcribe the beats of a sample set of songs by hand, and then evaluate how well the python script’s output matched our handmade solution. This accuracy value will be determined by the total proportion of time in which the handmade solution was in agreement with the python output, with any areas of disagreement being marked as inaccuracies.

For the latency concern, this can be assessed via observing the delays in electrical signals within the hardware. For our start point, we could use the time at which the voltage first appears from the piezoelectric or pressure sensor, and our endpoint could be the time at which the USB line from the microcontroller to the computer stops sending information. If for any reason this delay is marginal compared to the overall delay in the system, we could also try using a high-speed camera in the worst case while using the action of the drum pad being depressed and the GUI displaying that change as our start and endpoints respectively.

For the accuracy of drum hits, we first will need to determine what we think to be a reasonable force to expect from a user hitting the drum pads. However, if that method proves too difficult or inconsistent, we could also very well use a certain degree of drum depression (such as a 1 cm depression of the pad) to categorize a “valid” drum hit. Once determining our method of providing “valid” hits, we will simply need to perform those hits repeatedly on our system and track any misrecognitions, both false-positives and failed reads. This value should hopefully be minimal if present at all, as an inconsistent control scheme is a sure fire way to disappoint users.

Lastly, for the system size, this is very easy to assess, as all we will need to do is measure the dimensions of our system. This testing is more likely to be a concern during the design of the modularized drum components, but even still, this limitation is more along the lines of a specification than a characteristic that can be failed.

MVP

For our MVP, we hope to produce a controller with 4 drum pads that is able to send signals corresponding to drum hits with the speed and accuracy specified above to an C++ script. From that point we desire our script to be capable of displaying these inputs, particularly how they align with expected inputs corresponding with a song. These expected inputs will likely be displayed as four columns on the screen, which progress downward as the time to expected hit approaches. Lastly, we also wish to develop a complementary python script that is capable of transforming a user provided song into a playable track within the aforementioned system. This python script should meet the expected accuracy specifications outlined previously. As a mitigation strategy, we would have ten pre-determined songs and their manually created beat maps to still make the game playable in case of issues with the automatic beat-mapping.