Karen Li
karenli

Project Proposal

Project Description:
Rhythm Keys is a rhythm game inspired by Guitar Hero. The game allows the user to upload a MIDI audio file and generates a chart of notes separated into some columns based on the information in the MIDI file, the difficulty level the player selects, and the instrument or part the player chooses to play. The player will press the keys to the notes in time with the music. There are also tokens and obstacles that can be added to the chart. A local multiplayer option is available and allows two players to compete using the same song and interfere with each other's note charts.

Competitive Analysis:
Guitar Hero Composed is a similar project. My project will be similar in that there is beat detection that determines when the keys will be pressed based on the audio file. It will also have different difficulty levels which is similar. It is different in that my project will allow the user to upload any MIDI file, given that the song only has one tempo. My project also allows the user to choose which instrument or part from the song the user would like to play.
My project is also similar to Go Girl, since they both have a multiplayer mode. It is different from Go Girl because Go Girl does not include music.

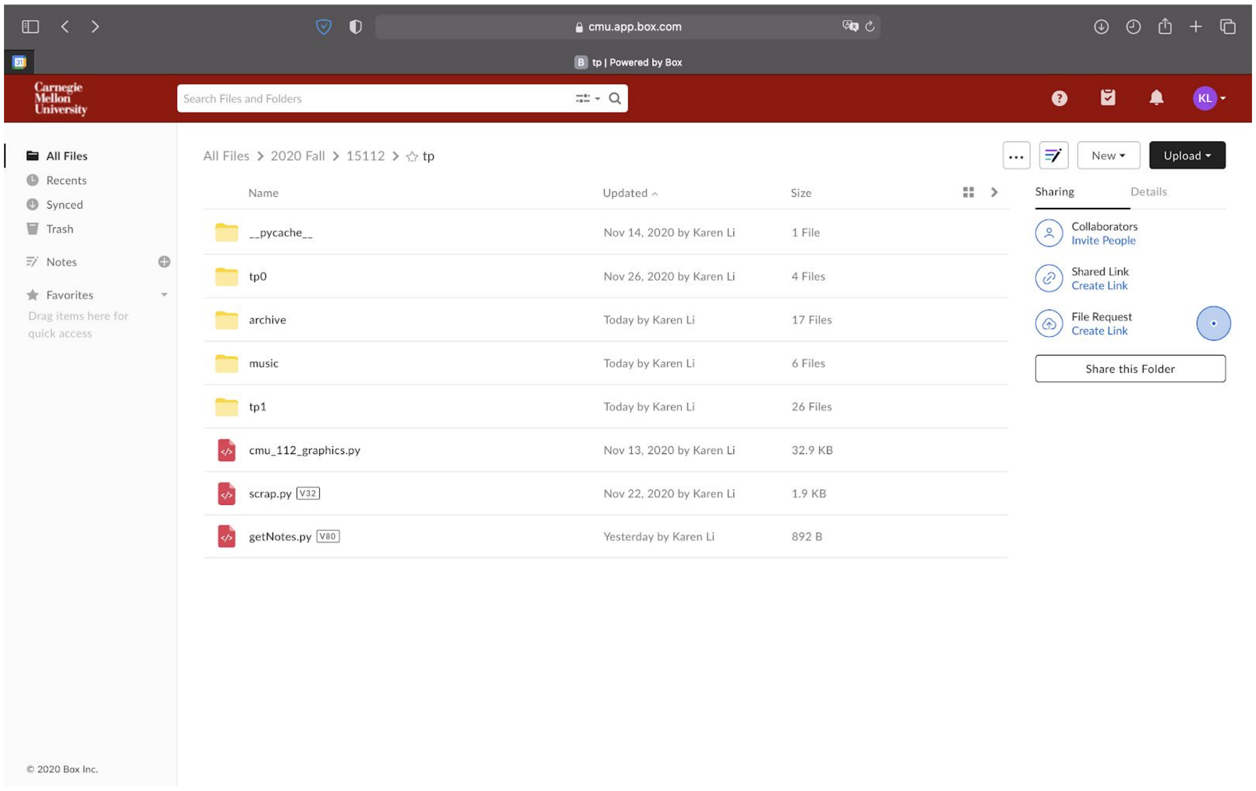
Structural Plan:
Each game mode, such as single player mode and multiplayer mode, will be its own class and each of these classes will be in its own file.
There will be 3 types of objects to be used in the note chart: Targets, Tokens, and Obstacles. Targets are the notes that are to be pressed and will earn the player points. Tokens are coins to be collected that will earn the player points. Obstacles are to be avoided and will decrease the player's points or end the game. Each of these objects has attributes such as its column number, where in that column it is located, and whether or not the object has been pressed before.
These objects are stored in 3 separate dictionaries, one for each type of object. Each dictionary will have a key for each column. Each key will map to a set of the particular objects.
I can also create an object for a note chart, since there will be multiple note charts in the multiplayer mode.

Algorithmic Plan:
Most algorithmically complex is creating the note chart and the difficulty levels.
Creating the note chart:
Using the music21 module, I will be able to extract information about the music, such as the tempo, a note's pitch, and a note's location or offset. I can loop through each element in the music "stream" to create the note chart for the game. The column number can be determined by

the note's pitch. The note must not overlap other notes in its column, so a set stores the coordinates of the previously added notes. This is the same for the tokens and obstacles.
Difficulty levels:
One indication of difficulty can be the number of notes that can be played at the same time. For the lowest difficulty, there can only be one note across all columns at any given time. The number of notes added to the chart can be restricted by checking if a note's y-coordinates would intersect with another note's. For the highest difficulty, there can be one note in each column at any given time. Extra notes apart from the notes from the MIDI file itself can be added to increase difficulty if there are not so many notes in the MIDI file.
Another indication of difficulty is the rate at which the notes need to be played. This can be lowered by adding only the notes with (offset % [some constant] == 0).
Difficulty can also be changed based on the number and size of obstacles added to the chart.
Tokens, obstacles, and targets:
In order to determine if these are pressed at the correct time according to the music (right when they reach the line at the bottom of the screen), I will check if the line intersects with the coordinates of any of the objects in the particular column that has been pressed.
If nothing intersects, then this is a missed hit and the score will decrease.

Timeline Plan:
Dec 3: difficulty algorithm, improved note extraction
Dec 5: local multiplayer mode
Dec 6: improved UI, potential "create" mode for player to create their own chart

Version Control Plan:
Code autosaves and backs up to Box.
For each major feature changed or added, a copy of the folder containing the project files is created and edited.



All Files > 2020 Fall > 15112 > tp

Name	Updated	Size
__pycache__	Nov 14, 2020 by Karen Li	1 File
tp0	Nov 26, 2020 by Karen Li	4 Files
archive	Today by Karen Li	17 Files
music	Today by Karen Li	6 Files
tp1	Today by Karen Li	26 Files
cmu_112_graphics.py	Nov 13, 2020 by Karen Li	32.9 KB
scrap.py V32	Nov 22, 2020 by Karen Li	1.9 KB
getNotes.py V80	Yesterday by Karen Li	892 B

Module List:
Music21: http://web.mit.edu/music21/
Pygame for audio

TP2 Update
I added a new object called “attacks” that are like the other game pieces but are only used for multiplayer mode. When hit, the other player’s keys are disabled for 5 seconds.

TP3 Update
I added a create mode which allows the player to compose their own music and create a gameboard at the same time. They are able to add notes and also tokens, obstacles, or attacks to a grid. They can play their song in create mode to see if they would like to edit the song further. This grid is then converted to a midi file and saved to the music folder. The grid’s data is also saved as a text file in the gameboards folder. The player can then play using the specific gameboard and song they created. This is different from generating a gameboard from just a MIDI file, since there is some variation in each gameboard generated depending on the difficulty level and random generation of obstacles, tokens, and attacks.
I also made some changes to graphics: targets now have colors based on their pitches, tokens are yellow circles, and obstacles are skulls.