**Term Project Proposal: Bagpipe Hero**

Project Description:

Bagpipe Hero is a game similar to Guitar Hero in its basic concepts and mechanics. The general concept is as follows: while music plays in the background, the player hits notes on some sort of controller device (in this case, the keyboard), attempting to hit notes on the screen that correspond to those in the song. However, the catch in Bagpipe Hero is that the game is bagpipe- and Scottish-themed. That is to say, not only does it have an aesthetic layout that is Celtic in nature, but it also features bagpipe music as its available default songs and incorporates components of playing bagpipes specifically into the game, such as having nine notes to play (as this is the entire range of the Highland Bagpipe) and having to “blow” into the bag (by pressing a button) to refill the air to keep the drones and chanter playing, among other differences. Bagpipe Hero also incorporates note analysis of music audio files such that music files can be uploaded and analyzed, allowing for the generation of game levels.

Competitive Analysis:

Bagpipe Hero contains many similarities to other Guitar Hero spin-offs, including past 15-112 projects. The following examples were used to complete an analysis of Bagpipe Hero’s competition:

* Lori Kipp’s “Keytar Hero” (found at <https://www.youtube.com/watch?v=u1LsuUSl2iE>)
* Alex Yu’s “Guitar Hero + Scroller Game” (found at <https://www.youtube.com/watch?v=-8alb6eMW-Q&t=84s>)
* Krishna Donepudi’s “Keytar Hero” (found at <https://www.youtube.com/watch?v=JyQ0HPu_dFc>)

Lori Kipp’s “Keytar Hero” features a unique-yet-traditional 3D experience for the user that very strongly resembles the graphic layout of the real Guitar Hero games. It utilizes 3D graphics to present notes that travel directly towards the user from inside of the screen, rather than traveling across the screen from one of the sides. The player presses keys on the keyboard as notes approach them. Additionally, the game offers the player options to customize the graphic aesthetic layout of the game. Alex Yu’s Guitar Hero game mixes traditional Guitar Hero Elements with the elements of a side scroller action arcade game. To “hit” notes during gameplay, the player moves around a character which can run into notes by moving from one part of the music staff to the next. The game also includes power-ups for the character to collect, as well as enemies that can reduce the player’s final score. Finally, Krishna Donepudi’s “Keytar Hero” uses beat detection to place notes on the screen using simplified arrow key symbols that correspond to the arrows on the computer keyboard. While it is not the most graphically complex, it does seem to be quite to-the point and user-friendly. Some features that all of the competitor projects contain include: four note options, custom song uploads, a list of high scores, and difficulty levels. Some key highlights for the competitor’s projects are outlined in the graph below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project | Graphics Layout | Enemies and Characters | Graphical Note Design | Customizable Background and Graphics | Difficulty Levels |
| Lori Kipp’s “Keytar Hero” | Complex, 3D | No | Traditional Guitar Hero (3D, no extended notes) | Yes | 3 |
| Alex Yu’s “Guitar Hero” | Moderate complexity, 2D with many elements | Yes | Traditional Guitar Hero (2D, extended notes present) | No | 4 (with an additional challenge feature) |
| Krishna Donepudi’s “Keytar Hero” | Relatively simple 2D layout | No | Arrow Keys (no extended notes) | No | 3 |

Based off of the data gathered from competitors, Bagpipe Hero will need to contain several features similar to these projects in order to be competitive. This includes an option to upload custom songs to the game, multiple difficulty levels, a high score board, and a graphical interface that is appealing to the user. Bagpipe Hero will establish its uniqueness in doubling as an educational experience for the player, teaching the user (who is likely to be unfamiliar with the Highland Bagpipe) through gameplay how the bagpipe works and what some key features of the instrument are. Bagpipe Hero will increase the number of note options available to the player to nine (the total number of notes that can be played on the bagpipe chanter) and include a feature where the player must “blow” into the bag by pressing a button (as this maintains air presence in the bag that allows it to keep playing). It will also display the finger positions for notes on an image of a bagpipe chanter as the notes are played and include a section for the player to read about how bagpipes are played.

Structural Plan:

Bagpipe Hero will be mostly (if not entirely) object-based and designed using a wide variety of classes and methods. The project will be broken up like so:

* Reference music audio files (for training the game and for playing default songs) will be stored in a directory entitled “Music”
* Players with high scores and their scores will be contained within a comma-separated text file, “highScores.txt”

The following will be contained within the main game Python file, “bagpipeHero.py”:

* The game itself will be played using PyGame, and will be stored as an object in the main Python file that runs the entire app. It will contain attributes for the screen width and height, the FPS for the game, the title of the game, the high scores of the game, the player’s current score, the current screen/mode being displayed and other objects if necessary. It will contain an \_\_init\_\_ method, a method for running the high-level game, and a method for running each screen in the game, including the start screen, the high score screen, the “More Info on Bagpiping” screen, and the gameplay screen.

The following will be contained within the “gameData.py” file:

* The PyGameNote class will store the sprites for notes in the game that appear on the screen (Rectangles corresponding to note length). It will contain attributes for the length of the note and the pitch of the note (by letter, that is), as well as the radius, x/y coordinates, Rect object, color, and image for the sprite. It will also include the update method which will change the note’s position.
* The BagPipeSprite class will contain the sprite for the bagpipe image that players will have to click in order to refill the bag with air while the song plays. It will include attributes for the sprite’s size, radius, x/y coordinates, Rect object, image, and fullness (that is, how much of the bagpipe sprite should be visible at a given time). It will have an update method to change its fullness attribute.
* The Scoreboard class will contain an attribute for the players with the high scores in the game, as well as one for the players’ scores.

The following will be contained within the “songData.py” file:

* The Song class will contain the song to be analyzed and played in the game. It will have attributes for the notes in the song, the song file, and it will have a method for analyzing the notes in a song.
* The AnalyzedNote class will contain information about the analyzed notes in the Song class. It will include attributes for the note’s length and pitch.

Algorithmic Plan:

The trickiest parts of the program will be analyzing and storing the notes, converting note data into usable note rectangles, switching between game screens in PyGame, and saving high scores.

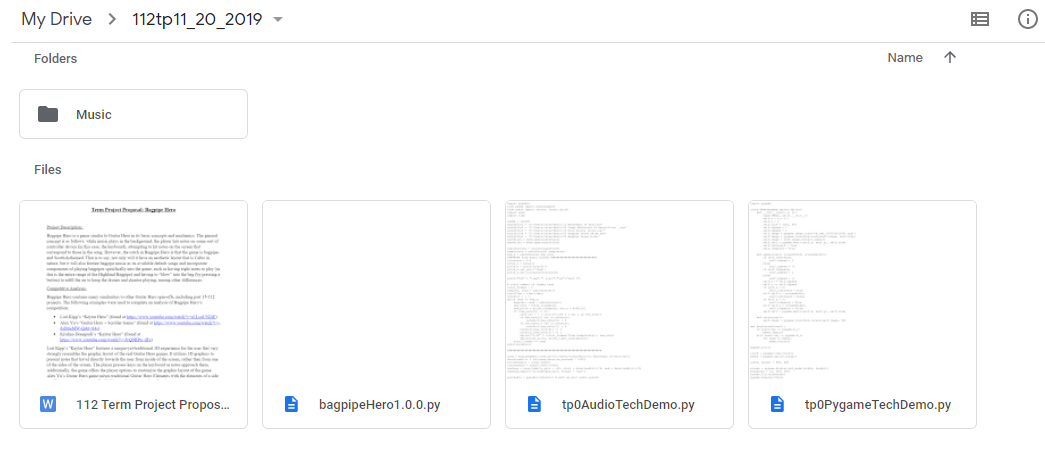
* Music will be analyzed using Aubio. Irrelevant sounds (i.e. bagpipe drones and snare drums) will be memorized by the program using a dictionary and reference files containing only bagpipe drone sounds and snare drums. Using the aubio.notes class, note values for a song will be stored in a list, and their counts for the song will be stored in a dictionary. The eight largest note counts that do not fall under irrelevant sounds are the 8 bagpipe notes. Using these note values, the note list can be cleaned up by replacing inconsistent or inaccurate note values with nearby note values that fall under the bagpipe range. Using this cleaned list, note values and their lengths can be used to generate another list of note objects for the song, which can then be converted into sprites as detailed below.
* Note rectangles will be generated using the note objects created during music analysis. The vertical position of the note will be determined by the note value. The length of the song will be divided by the length of the note in order to determine the relative length of the bar in pyGame (the notes move across the screen at a constant rate, so the length of the song determines how long notes are moving across the screen).
* Switching between game screens will be done by having a game mode attribute set to different string values, each one unique to a game mode/screen. The run method will switch between game screen methods based on these strings, which can be changed as needed in the game screen methods themselves.
* High scores will be saved in a csv file. The csv file can be overwritten with user score and name input if they earn a new high score.

Timeline Plan:

The Bagpipe Hero project timeline is designed to reach an MVP by the TP2 deadline. As such, all dates listed below will be before TP2, although 100 percent of these features may not be completed to their fullest extent by the dates listed below. Any work beyond that would be working out bugs in the project and adding additional post-MVP features that are not detailed in this project proposal. The timeline for the project is as such:

* Audio analysis algorithm complete by 11/20/19
* Song class and AnalyzedNote class complete by 11/21/19
* PyGameNote, BagpipeSprite classes and gameplay mode complete by 11/23/19
* Song choice screen complete by 11/24/19
* High score screen complete by 11/25/19
* Start screen and “More info on Bagpiping” screen complete by 11/26/19

Version Control Plan: I will regularly back up my term project files by uploading them as a folder labeled by date into my CMU Google Drive, as the following image depicts:



Module List: Aubio (an audio analysis tool), Wave (for loading wav audio files into python), PyGame (for the actual game GUI), and PyDub (for a small bit of audio file analysis, though the module is usually used for audio modification outside of this project)

TP2 Update: For the most part, the algorithmic plan for Bagpipe Hero has remained the same. However, a couple of structural changes have been made since TP1. The “PyGameNote” class was renamed the “NoteSprite” class. Additionally, the line where the player presses keys to hit notes was made into a sprite (to easily detect collisions) rather than a plain line drawn on the screen, named under the “HitNoteLine” class. Finally, in musicAnalysis.py, the Song class was given two subclasses: TrainingSong and PlayableSong. Songs that are analyzed for the sake of cutting out noise in the analysis of played songs (i.e. bagpipe drones and snare drums) are put into the TrainingSong class, which does a lot less processing of audio data compared to the PlayableSong class (which includes the songs that are thoroughly analyzed and converted into Bagpipe Hero levels).

TP3 Update:

* Suggestions from TP User Study-a-Thon:
  + Change up game screen graphics
    - Partially accepted: Did not change NoteSprite rectangles but did change color scheme from black-on-white to white-on-blue
  + Change the font
    - Accepted: Changed main font to a custom Celtic font found online and changed the small font to Georgia font
  + On the help screen, add an image to show players optimal finger positioning on keyboard for playing game
    - Accepted: image/diagram added to help screen
  + Add a mechanic for holding down keys for longer notes rather than just plain key presses
    - Rejected for time’s sake

The main structural change was the addition of a new game mode, the enterPlayerName mode. This mode is activated at the end of a song in which a player gets on the high score board. It displays a congratulations message, prompting the player to enter their name, and once the name is entered, the high score is saved to the csv, and the score screen is displayed once again like normal. Also, the Scoreboard class was never created, as the players and their high scores were easily stored on the high score csv file by itself. Other new features not specified in the original proposal include features for added functionality, mainly the ability to pause and quit the game directly from the core gameplay screen.

Everything else in the project stayed pretty similar to what the original proposal plan was.