1. Year and Semester: 2013 SPRING

2. Course Number: CS-492

3. Course Title: Kinect Programming

4. Work Number: PR-06

5. Work Name: Kinect Project Report

6. Work Version: Version 1

7. Long Date: Thursday, 16, May, 2013

8. Author(s) Name(s): Jake Waffle

Kinect Project Progress Report

License Statement:

This file is part of the "Theme Song Generation," Kinect Project for CS 312. The "Theme Song Generation," Kinect Project is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The "Theme Song Generation," Kinect Project is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Project Name: "Theme Song Generation"

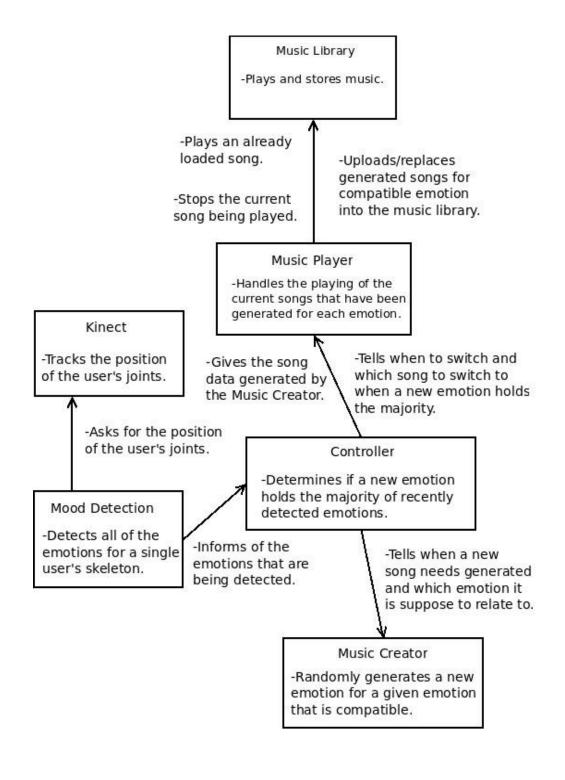
Purpose: To entertain an audience with music that expresses the audience's overall observed emotion.

Requirements: This application needed to be able to detect the emotions being signaled by a user.

Then it was suppose to play generated music that suited the observed emotion. And the emotion

detection was to be done by checking the user's relative joint positions.

Architecture:



Detailed Design:

Mood Detection

PoseRule

-This defines a relative relation between two joints and is able to check if the relation is being observed in a specific user.

-The special thing about this is that it can define a bunch of different relations between two joints. For instance, it can tell i a joint is above/below/left of/in front/within a distance of another joint. There is even a rule for checking to see if the user is leaning forward.

 -Asks if a PoseRule object is observed in a specific user.

LimbPose

-Manages a number of PoseRule objects that are associated with the emotion that this LimbPose object would be for (each LimbPose object defines the pose of a limb with PoseRule objects for a specific emotion.)

 -Can determine if all of the PoseRules inside of it are being observed for the specific user that this is for.

 -Can return the emotion that a LimbPose object would be associated with.

-Asks which emotion a LimbPose object represents.

-Asks if the LimbPose is being observed in a user.

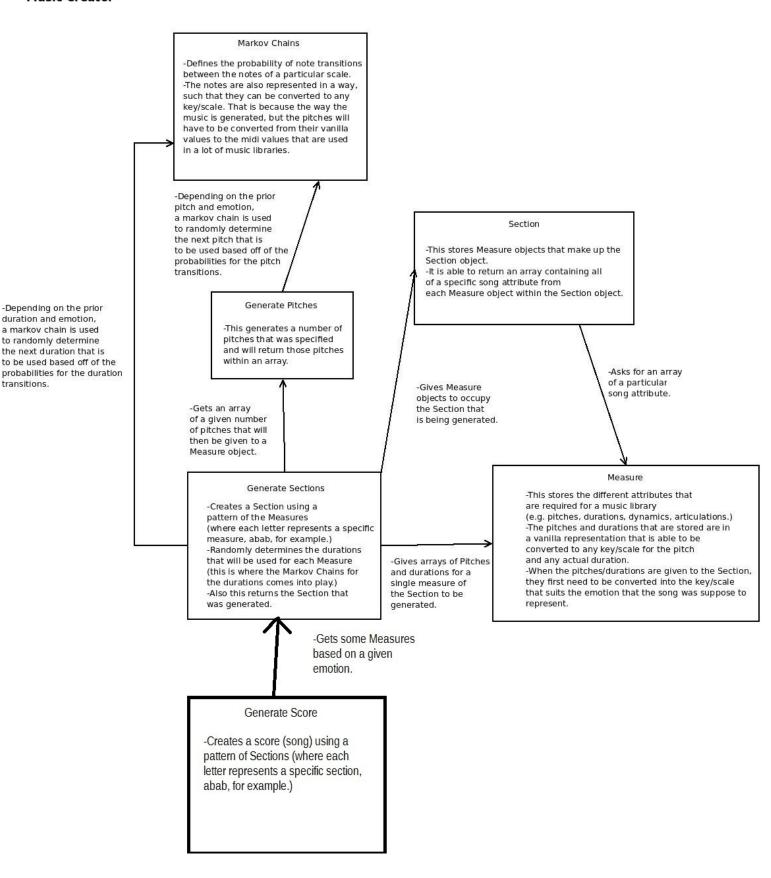
-Gives a PoseRule object to a LimbPose object. That PoseRule helps represent LimbPose, but there could be many of them representing the LimbPose object.

Mood Skeleton

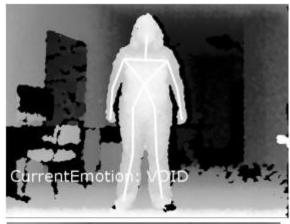
- -Draws a user's skeleton (a Mood Skeleton object only works for a single user.)
- -Determines the mostProminentEmotion.
- -Manages the rules that are placed on individual limbs of a user.
- -Can get an ArraylList containing the emotions that are currently being detected.
- -Can determine how many LimbPose objects are being followed by the user for a particular emotion and limb.

Music Creator

transitions.



Compatible Emotions:











Code Connections to the Architecture:

Mood Detection:

- -This component is basically just made up of three classes: MoodSkeleton, LimbPose and PoseRule.

 Those classes all exist within the MoodSkeleton.pde file. Those classes are connected to the main part of the program through the setup(), draw() and drawMoodSkeleton() functions.
- -Within the setup(), the rules that define the poses for each emotion are loaded into a single MoodSkeleton object.
- -Then that object is used within the drawMoodSkeleton() to draw the skeleton of a specified user (while also getting the most prominent emotion detected in the user and putting that within the detectedEmotionsQueue.)
- -And the drawMoodSkeleton() function is called within the draw() function for each user that is calibrated.

Music Creator:

-This component is made up of two classes within the Measure.pde (Measure and Section) and a few functions that exist within the main file (GenerateScore(), GenerateSection(), GeneratePitches().)

-The Music Creator is only used when it needs to be and calling GenerateScore() will trigger all of the creation to happen. The GenerateScore() function is used within the setup() to generate a song for each emotion that is compatible. And it is also used when a new emotion is detected (for the old emotion of course,) because the song for the previous emotion shouldn't be played the next time that emotion is observed.

Controller:

- -This component is made up of some logic at the end of the draw() function, a global variable (detectedEmotionsQueue) and a line of code within the drawMoodSkeleton() function.
- -The logic at the end of the draw() function essentially will check to see if there is a new emotion that is taking up the majority of the space within the detectedEmotionsQueue. And if there is a new emotion taking up the majority, the song that was already generated for that emotion will be switched to (and the old emotion will have a new song generated.) But before doing that, the logic will check to see if it has been a certain amount of time since the last song was played (when a song is stopped before it even plays, jm-etude (a music library) will throw an error.)
- -The detectedEmotionsQueue is an array that holds strings that represent the compatible emotions. It may be an array, but it should be a class object instead. That is because when the detected emotions are added to it, it must shift all of the elements over and remove the emotion at the opposite side (the behaviors are like a queue anyway.)
- -The line of code within the drawMoodSkeleton() function will essentially get the most prominent emotion observed in a user and put it into the detectedEmotionsQueue (before that, the elements within the array are shifted over though.)

Music Player:

- -This component is made up of a couple function and a music library object: playScore() and addScoreToEtude().
- -The playScore() function essentially will stop the previous song that was being played by jm-etude (a music library) and then it will play the song associated with the given emotion. It plays the song by telling the music library object to play a song that has the title of the given emotion.
- -The music library object is from the jm-etude music library (it's called Etude.) It will hold the data for the songs that are currently generated and is able to play those songs when told to do so.
- -The addScoreToEtude() function will essentially grab the data out of the data structures that the generated music was put into (Section and Measure, defined in the program, the Measures.pde specifically) and will give it to the Etude object (the music library object that plays songs.) This function was basically meant to mediate between the music library I was using and the way I wanted to store my generated music.

References

jm-Etude. 15 April 2013. http://jmetude.dihardja.de/index.html>.