Crispy Potatoes APCS1 pd 4
Marcus Ng, Shaikat Islam, and Victor Teoh 2017-01-10

Final Project Proposal

Rap Battle RPG

Our project will be a rap battle RPG where the user raps against other rappers to gain money. When the game starts, the user will input a name and hometown. The user will be able to fight other rappers by spitting bars in a rap battle. The more fire the bars are, the more damage will be inflicted upon the opposing rapper. By winning rap battles, the user gains money to purchase upgrades to increase their health, defense, and attack stats. As you win more rap battles, the enemy rappers deal more damage. In order to win the game, the user must win a certain amount of battles to get sponsored by a record label.

Character.java will be an abstract super class of Player.java and Enemy.java because the player and enemy have the same variables (name, hometown, hp, defense, attack) and methods (isAlive(), getName(), getHP(), attack(), etc.).

Shop.java will have the buyItem() method, which will allow the user to buy an item if they can afford it. Item.java will have the variables: itemName, price, and desc. Item.java will consist of multiple getters so the shop is able to access information about each item. The shop will store available items in an array for the player to purchase items from.

Rap Battle String Analysis:

a) Rhyme, alliteration, consonance, assonance, word length, and other factors:
Our program takes into account phonetic pronunciations of different words in a .txt file, provided by Carnegie Mellon University. We parse through this data using the Scanner class and make a rhyming dictionary. When the user ‘spits’, the sentence is parsed and our program compares the phonetic transcription of the different words, as well as factors such as alliteration, the use of ‘multis’ (which are multisyllabic rhymes at the end of a rap line), and assonance. Most of the methods made for this purpose will be defined in a separate file from the actual RPG. Our github repo has some of this done, albeit at a rudimentary level.

Our RapAnalyzer class contains the different methods that analyze the strings, which would be sentences in this case. Using our ‘rhymecheck’ method, which parses the sentence into individual words and entries in an arraylist, each word and its phonetic transcription (taken from the dictionary arraylist) is compared to each other word in the arraylist. For multiple sentences/lyrics, our rhymecheck method compares each of the words at the ends of each lyric with those at the end of the second sentence, and so on.

The Scanner class will be used heavily in building our rhyme/alliteration/multisyllabic rhyme checking methods. The multisyllabic rhyme checker is a beefed up version of our original checker in that it analyzes two sentences for their phonetic transcriptions.

The confidence level for a rhyme is based on the following formula:
(# phonemes that match) divided by **(average # of phonemes in both words).**

If the confidence level of those two words match that of a *preset* confidence level (usually 40%, in our case), it counts as a rhyme. As the player progresses through the game, the confidence level will increase forcing the player to rap better to deal more damage.

Furthermore, these methods, which serve as the heart of our game, will require a great deal of string manipulation, as learned nearing the beginning of the term. Our method also uses ArrayLists to parse sentences and check for consonance (alliteration), which is easy to do, in our case.

The rest of our RPG uses a system of classes and subclasses, using inheritance and polymorphism to create different levels, characters, hometowns, etc--our work draws on the RPG that we created for the class.

User input will be managed through the cs1.Keyboard package
CMU pronunciation: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

b) The rap lyrics generated by the NPCs/Bosses can be provided by two different mediums: either we can generate our own rap lyrics, increasing the difficulty as the user progresses, or we can have a .java file containing an array of rap lyrics (from the real world). This can be accomplished through the use of the Genius API, or just a lot of copying and pasting from their website. We plan to store rap lyrics in an array of strings and use a random number generator to select the lyric that the enemy would 'spit.' The enemy rapper is assigned a lyric randomly, but depending on the level at which the user is at, the lyrics will become harder. In order to make the enemies harder as you progress through the game, we would store weaker lyrics in a different array than stronger lyrics and choose the lyric based how strong the enemy rapper is.

c) There will have to be a way to constitute how one 'wins' against a NPC generated rap lyric. Using the methods determined above, there will be a 'fire' factor that takes into account rhyme, alliteration, consonance, word length, etc. If the 'fire' factor of the enemy rapper is lower than that of the user, the user will inflict more damage.