Neuromancer

Contributor Descriptions

# Andrew Trewin

I was responsible for handling information retrieval and analysis along with some high-level speech analysis. These functions were placed collectively in the Wintermute module. This module contained many classes for low level manipulation of data. This was later abstracted and blended further in **wintermute.core.wintermute** where many packages were consolidated into a single method. This high-level access and inline analysis made it possible for Neuromancer (the front end) to reply to complex queries with pre-processed data. Another thing that made these packages crucial was the management of DataSets (**wintermute.data.DataSet**) and Nodes (**wintermute.data.Node**) which compose a hierarchy of related data structures. These data structures can be referred to by name and come in a variety of flavors. VoiceActor (**neuromancer.voice.VoiceActor**), a joint effort between Liam and I as it contained code that could easily have fallen within the realm of either module, exists to allow requests to be stripped to bare elements by parts of speech (verbs, nouns, etc.) and handled accordingly. Wintermute’s packages contain many classes accessed directly from Neuromancer without much abstraction, such as any class within **wintermute.music**. These classes do not pass through **core.Wintermute** at all as they are simple and static by nature.

Wintermute’s handling of the MediaWiki API is so complex I feel it deserves its own section. The method **wintermute.core.Wintermute.formatWikitext(String input)** focuses on removing excess junk from Wikitext, MediaWiki’s markup language of choice. I was required to filter out wikitext’s bizarre link scheme, which consists of something similar to this:

Carbon is an element that composes [[Graphite|pencil lead]] and is also used to make [[graphine]]. {{File:Carbon.jpg|Some carbon}}

This needed to be distilled into human readable text, and no method for doing so has currently been devised. The method used in Wintermute consists of a primitive collection of while loops to remove content that does not match. This method is not without its downsides, as it takes up expensive processing cycles to complete the operation. Another method abstracted to by Wintermute is **addWikiByName(String name)**, which will search Google for a wiki, discern where the API lies, and add it to an **ArrayList<String,WikiObj>**. This list can later be accessed, and any method can retrieve content from a custom wiki. A Javadoc will be bundled with the source code when it is submitted on presentation day, which includes everything necessary to understand the entirety of the code.

Neuromancer

Contributor Descriptions

# Liam Reckley

I was assigned to create the packages that make up Neuromancer, the front end of the collective entity composed of Wintermute and Neuromancer. These packages dabbled in speech recognition, processing, and synthesis. Google’s speech recognition API was utilized to allow for quick and easy retrieval of raw speech. These chunks were nowhere near ready to be machine-readable. The recognition process consists of **4** steps:

1. Remove junk words like “for”, “the”, and “a”
2. Iterate through every word in a given input and look for a word that is classified as a verb
3. Match this verb against a list of known verbs
4. Apply action as necessary

All of these steps are contained within **neuromancer.voice.VoiceActor.act(String input)**. This method acts in a similar way to that of Wintermute’s own methods in that it methodically scans each chunk for value, yet mimics Neuromancer’s style as it deals with voice and the use of heavy abstraction.

Speech synthesis was a problem that gave me quite a bit of trouble for about half of the project. Native Java libraries such as FreeTTS proved too complex or poor quality. I finally settled on eSpeak’s command line app and attached MBROLA voices. This allowed for higher quality synthesis and no multithreading requirement (although this was entirely possible).

Neuromancer contains the ability to interact with much of the data storage engine Wintermute possesses. This allows DataSets to be modified and read by voice command alone. Pattern analysis is orchestrated (yet not preformed directly) by these same controlling forces. The resulting data is cut up and reassembled around a unique yet similar template sentence which is then read to the user.

Neuromancer presides over all graphical functions, including animations, buttons, and image display. The **neuromancer.core.Neuromancer** class extends Applet and as such contains all graphics and mouse events. Any image retrieved from MediaWiki is rendered here, and any text that may be relevant is also retrieved and displayed.