1. プログラムについて（カルビン）

　上記に述べたように，本プロジェクトは2chanで流行っている“野獣先輩”というネタキャラクターをもととした人口無能であり，多様性を測るために5つの辞書を用いた人口無能である．４つの辞書の中身は普段の会話の辞書で，もう1つの辞書は入力ミスや前の４つの辞書に登録されていない返答が返ってくるとき（例外処理）のエントリーが含まれている辞書である．我々が目指している人口無能の設定とその流れをこれから述べていく．

　設定としては，ユーザがあるチャットルームで，“野獣先輩”と出会って，会話をしようと思っている．“野獣先輩”は最初に“あなたは誰？”という挨拶から会話を始めようとする．これに，ユーザが返事し，その返事がユーザ名となる．そして，事前に用意された先ほどの４つの辞書を用い，ランダムに選択され，“野獣先輩”は“ほお、＜ユーザ名＞，＜辞書の最初のエントリ＞”をユーザに返事を送信する．この返事に対し，ユーザは会話を自然に“野獣先輩”が思う通りに継続しなければならない．つまり，辞書に登録されているエントリーの流れに従うということである．もし，ユーザが自然な返事をしなかったら，“野獣先輩”が起こりはじめ，怒りを込めた返答をし始める．自然ではないレスポンスをユーザが継続していくと，徐々に“野獣先輩”の返事が悪化していき，最終的に非常に不良な言葉で返答する．怒りの言葉をし始める前の“野獣先輩”の発言にきちんと返事すれば（辞書に返事の対応が含まれている場合），会話が進んでいくようになり，“野獣先輩”の怒り度数が減っていく．“野獣先輩”が飽きたら（辞書の最後のエントリにたどり着く場合），会話が終了し，”リスタート?”というレスポンスが返り，“Yes”などのような同意する言葉を送信したら会話をもう一度やり直すことができる．会話の途中で，ユーザが“bye”などのような送別を意味する言葉を送信する場合，“野獣先輩”が“元気でな。”という送別挨拶を言い，会話を終了させる．

　この設定を実現するにあたり，我々が作成した人口無能は主に二つの部分に分割することができる．一つ目の部分はGUIの部分で，二つ目の部分は対話処理の部分である．今回のプロジェクトでは，我々は他チームと別の方法でプログラムの作成を行い，使用される言語はJavaではなく，Pythonである．Pythonにした理由はいくつかあるが，以下に示す．

* 他の言語と比べるときに，比較的に簡単にプログラムのフィーチャーや機能などを実現することができる．
* すでにあるプログラムに制限されず，開発の際，自分が実現したい機能に集中でき，自由度が高くなる．
* GUIを作成するつもりであったため，比較的に早く開発ができるPythonのGUIライブラリを用いたい．
* 自分のPythonを用いた開発能力の向上につながる．

　GUIを作成するために，本プロジェクトはKivyというPythonのGUIライブラリを用いる．このライブラリは，マルチプラトフォームで，モバイル型デバイスまで対応できるというところが特徴である．

* 1. GUIの部分（はやてええ）
  2. 対話処理（カルビン）

　Pythonで対話処理を作成するため，あらかじめ用意されたJavaプログラムを使用することができず，そのプログラムの機能を完全にいちから実現する必要がある．Javaのプログラムを分析した結果，主にいくつかの大きな処理に分けられると考えた．これらの処理をリストアップすると以下となる．

* 辞書を読み込む処理
* 辞書を参考にし，ユーザの返事に対応する処理
* ユーザに対する返答を出力し，音声を流す処理

　最初の2つの処理はPythonの標準ライブラリで実現できるが，音声を流す部分を実現するために，Pythonの標準ライブラリの上に以下のライブラリを導入した．

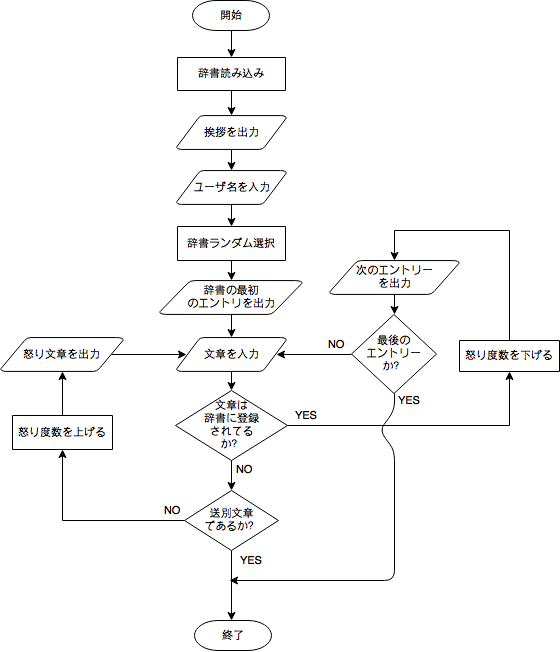
1. pykakasi

このライブラリとは，KAKASIという言語フィルタをPythonで使用できるようにしたライブラリのことである．KAKASI（Kanji Kana Simple Inverter）は，漢字，ひらがな，カタカナのいずれかをひらがな，カタカナ，ローマ字のいずれかに変換する言語フィルタのことである．このライブラリを使用すると，辞書の中に書かれているエントリーをローマ字（訓令式ローマ字）に変換することができるため，辞書は漢字，ひらがな，カタカナで書いても音声ローマ字で書かれている音声ファイルにも対応ができるようになる．

1. pygame

pygameとは，従来ゲームの開発を支援するライブラリのことである．しかし，そのパッケージの中で，音声ファイルを非同期に流す処理が含まれており，この処理を使用すると，遅れなしで音声ファイルを流すようになる．

　本プログラムのフローチャートを以下の図に表す．



図？？プログラムのフローチャート

　最初に，用意された辞書を読み込み，各エントリをリストに入れる．辞書の一つのエントリでは，現在のID，次のID，出力文章，入力文章の４つの要素が入っており，その間にタブで分ける．辞書を読み込んだ後，挨拶を出力し，名前を問う．ユーザは名前を入力し，その名前を今回のセッションのユーザ名となる．

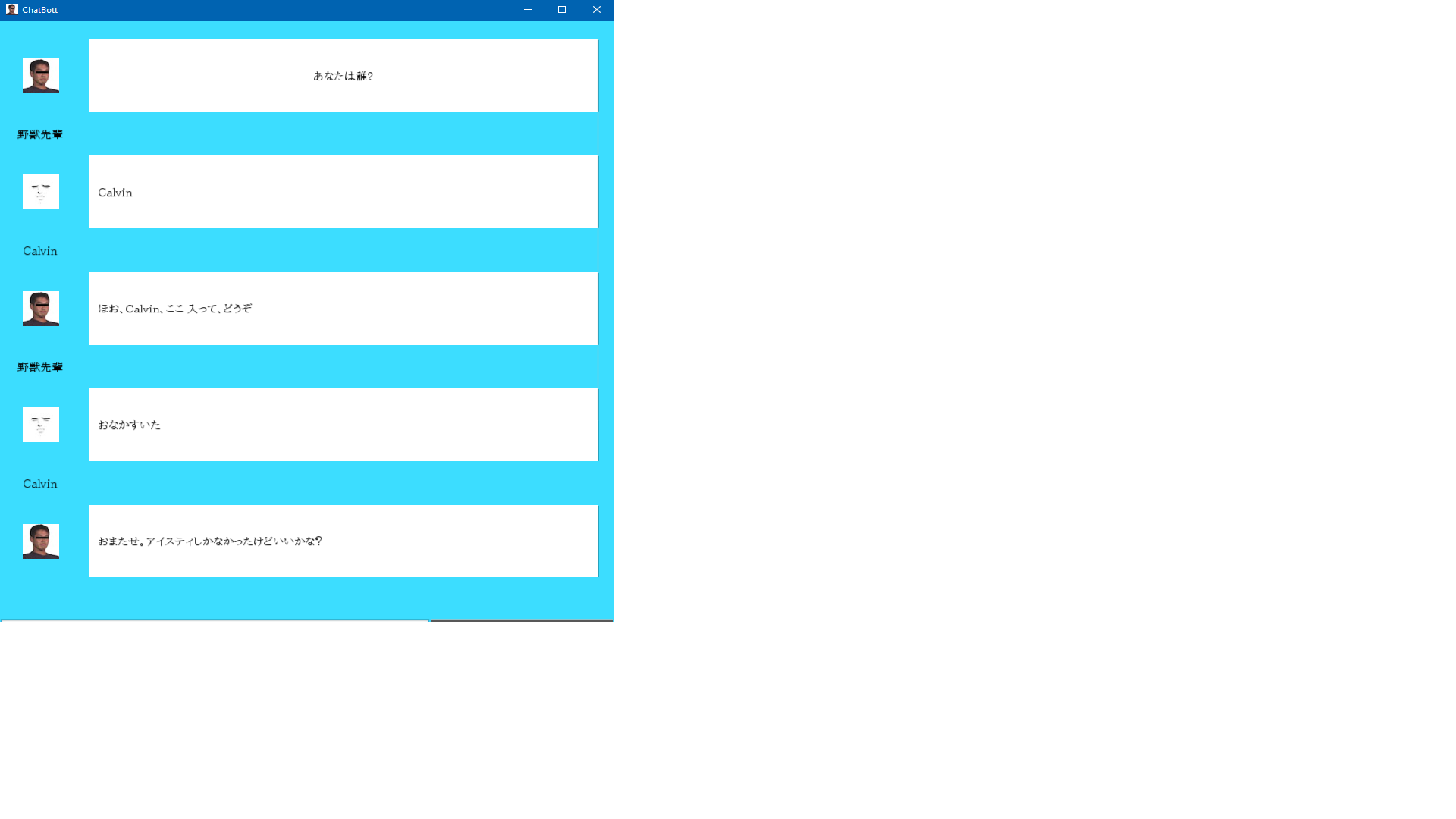
　次に，リストからランダムに一つの辞書を選択し，今回の会話の筋にする．そして，その辞書の最初の出力文章を出力し，現在の連鎖番号をそのIDとする．ユーザがこの出力文章に対し文章を入力し，現在の連鎖番号の中の入力文章の可能性の中に含まれていたら，怒り度数を下げながら，次のエントリーのID番号に変換し，その番号の出力文章を出力する．会話が継続できるように，この処理を繰り返す．もし，入力文章はその可能性に含まれていなかったら，怒り度数を上げ，怒り度数に合う文章を出力する．現在の連鎖番号の中にある可能性のある入力文章が答えられたまで，この処理を繰り返す．

　会話を終了させる方法が二つ存在する．一つ目は，ユーザが辞書の最後のエントリーまでたどり着くときである．二つ目は，ユーザが送別文章を入力するときである．この時， “元気でな。”という文章を出力する．

　本プログラムは，文章を出力すると同時に，文章に合わせた音声ファイルを流す．しかし，音声ファイルが存在しない場合（この場合では怒りの文章），そのエントリーの文章を上記に述べたpykakasiというライブラリを用い，ローマ字に変換する．変換した後，そのローマ字の文章をあらかじめ録音した訓令ローマ字の音素ごとに合わせ，単純な音声合成のようにその音素の音声ファイルを連続的に流していく．上記にも述べたように，音声ファイルはpygameのオーディオ処理機能で流す．

1. 会話の例（カルビン）

　ここでは，いくつかの動作例を挙げる．最初に，普通に会話が流れるときのスクリーンショットを以下の図？？と図？？に示す．



図？？普段の会話（前半）



図？？普段の会話（後半）

　これらの図に示す会話の流れは，１番目辞書（Dictionary 1）が選択され，辞書に登録されている返事をした場合である．会話が終了するときに，通常に“リスタート？”が返答されることがわかった．これに“yes”と返事すると，再開される．この状態を以下の図？？に示す．



図？？会話を再開する

　そして，会話の途中で辞書に登録されていない返答をする場合，怒り度数を増加させ，それに合わせたレスポンスを返す例を挙げる．この動作例を以下の図？？に表す．また，デバッグ用スクリーンを示す図？？の“angry level”と“angry percentage”に注目すると，怒り度数が通常に上がっていることが分かった．

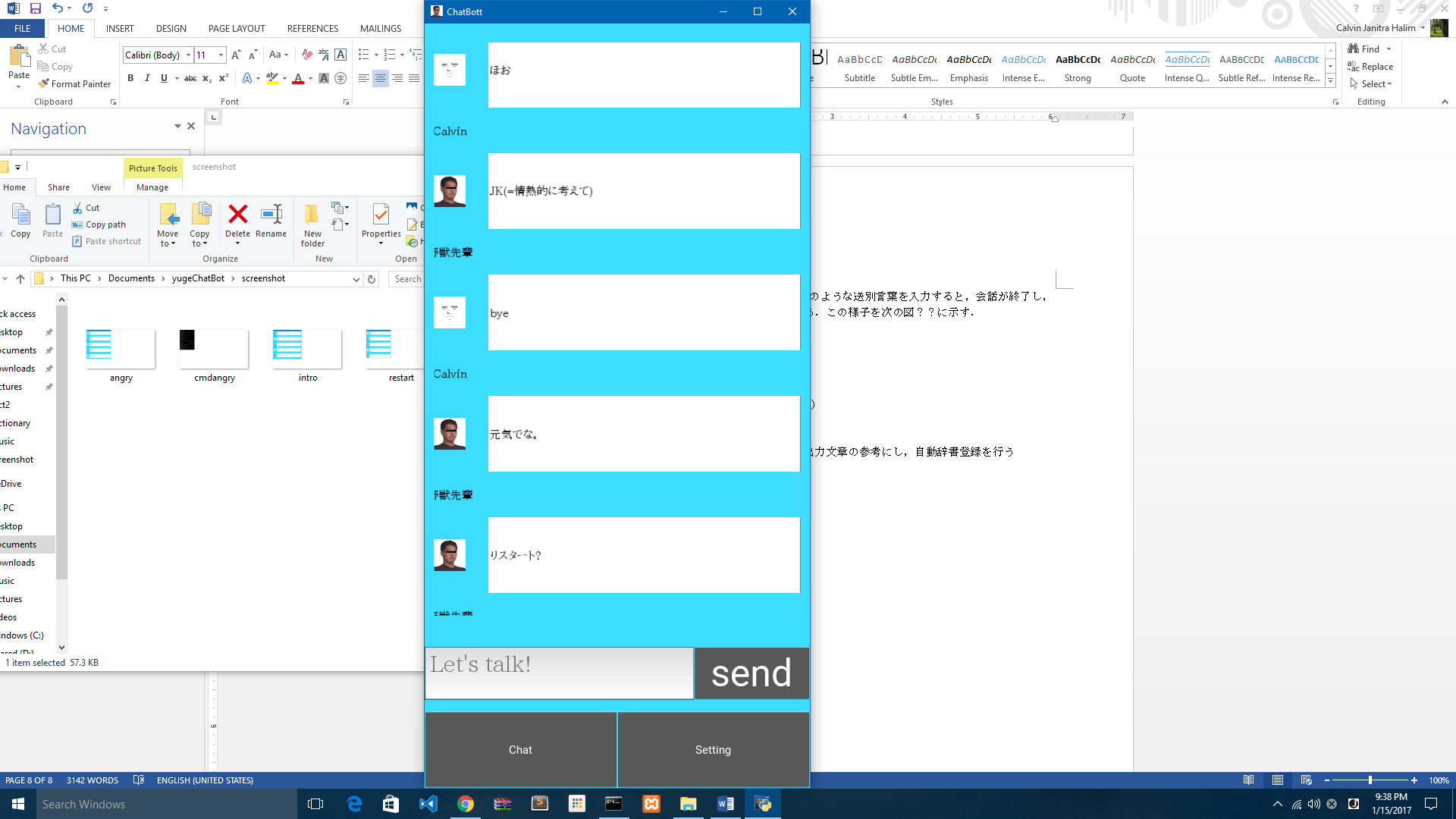


図？？怒っている様子



図？？怒っている様子（デバッグ）

　最後に，会話の途中で，“bye”などのような送別言葉を入力すると，会話が終了し，“元気でな。”という言葉が返ってくる．この様子を次の図？？に示す．



図？？会話を強制的に終了させる場合

1. 今後の課題（カルビン，みんな？？）

　本プロジェクトでは，マルコフ連鎖を利用したかったが，時間の都合上や辞書のスケールがあまり大きくないせいで応用できなかった．今後の課題としては，形態素解析を用い，ユーザが入力した文章をいくつかの部分に分け，それらの部分からユーザが言いたいことを分析する．分析した文章から，辞書から意味を推測し，マルコフ連鎖で確率的に返事を返すという機能を実現していきたい．

　また，ユーザの入力文章が辞書に登録されていなかったら，ダイナミックに対応できるように，辞書の自動登録を行いたい．登録されていない文章が入力されたら，人口無能はそれに対し質問をし，意味をユーザに問う．これを行うことにより，意味と一緒にキーワードを辞書に登録でき，さらに効率のいい学習システムになると予想される．

　現在では，音声ファイルが存在しない場合は音声合成を用い自動的にスピーチ音声を発生させるが，音素を連続的に流すときに，まだ自然的ではないと感じる．さらに自然的にスピーチを発生できるように，様々なアクセントやイントネーションで発言した音素を録音し，文章に合わせたアクセントでスピーチを発生させたい．これと同時に，漢字を変換するときに文章に合わせていないため，不思議な変換し方が発生する可能性がある（例：入って→いつつて）ため，今後の課題とする．そして，現在日本語にしか対応していないため，今後ローマ字スペリングに対応していきたいと思っている．

　GUIの改善はやてえ

　みんなの感想でも

1. 感想

　初めてこのような人工知能技術を用いたプログラムを作成し，前知らなかった様々な知識が知るようになり，非常にたくさんのことを勉強できたと思う．最初に開発を始めたときに，自分の中でいろいろなアイディアを実現したかったが，一から開発するということで困難であることが分かった．また，スケジュール管理があまりできていないことも一つの原因といえる．そのため，今回のプロジェクトを参考にし次のプロジェクトのスケジュール管理をもっと効率的にやっていきたいと思っている．しかし，よかったと思ったことが一つで，自分が持っているPythonの知識を実現できたので，よかったと思う．最後に，このプロジェクトにわたって，ノウホウやみんなとのチームワークの大切を学ぶことができたと思うので，非常に良かったと思う．（カルビン）