Course Project Documentation

CS101 Project

Akinator

TEAM CODE : 206

Vidit Gupta 140110078

Dharamveer Kumar 140010023(Team Leader)

Nitya Singh 140010028

Ila Toppo 140010043

Table Of Contents

1. Introduction

Our project Akinator is a PC-based game, which attempts to determine which character the player has in mind by asking him or her a series of questions, to which the player can answer in “Yes”, “No” or “Don’t Know”.

We have tried to make a small version of the web based game “Akinator- The Web Genius”. We have reduced the size of database. The main game works on the basis of probabilistic approach and decision tree. We have removed the probabilistic feature which will reduce errors but the number of questions to guess a personality will increase. Also our game works on a predefined and limited database of characters which can be later extended.

Definitions, Acronyms and Abbreviations

Database: It is the tabular information about the celebrities.
Question set: It is a string of questions associated with different properties of the characters.

Domain: It is the fields of character spanned by the questions.

Quality: Quality is a particular field for example type of bowler, party to which player belongs, etc

Attributes: There are the variations in qualities. For example, leg break bowler, seam bowler.

2. Problem Statement

- We have attempted to build a game like Akinator- The Web Genius, which attempts to determine which character the player has in mind by asking him or her a series of questions, to which the player can answer in “Yes”, “No” or “Don’t Know”.

- To create a database of selected personalities and frame the questions to be asked.

- To design the decision tree.

3. Requirements

Software Requirements

1. SimpleCpp Code Blocks(http://www.cse.iitb.ac.in/~ranade/simplecpp/): The IDE used in this project is SimpleCpp code blocks, developed by IIT Bombay. It includes SimpleCpp graphics in built, the graphics package which has been used in this project.
2. 'graphics.h' package(http://winbgim.codecutter.org/): Used for developing graphics. It was included to enable the resizing of text in the canvas.

Hardware Requirements

1. A minimum of 256MB RAM is required for smooth functioning this game.

Instructions

1. Setup SimpleCodeBlocks

a) Download the setup file from http://www.cse.iitb.ac.in/~ranade/simplecpp/
b) Run the setup file.
c) Follow the instructions onscreen

2. Setup Borland BGI Graphics

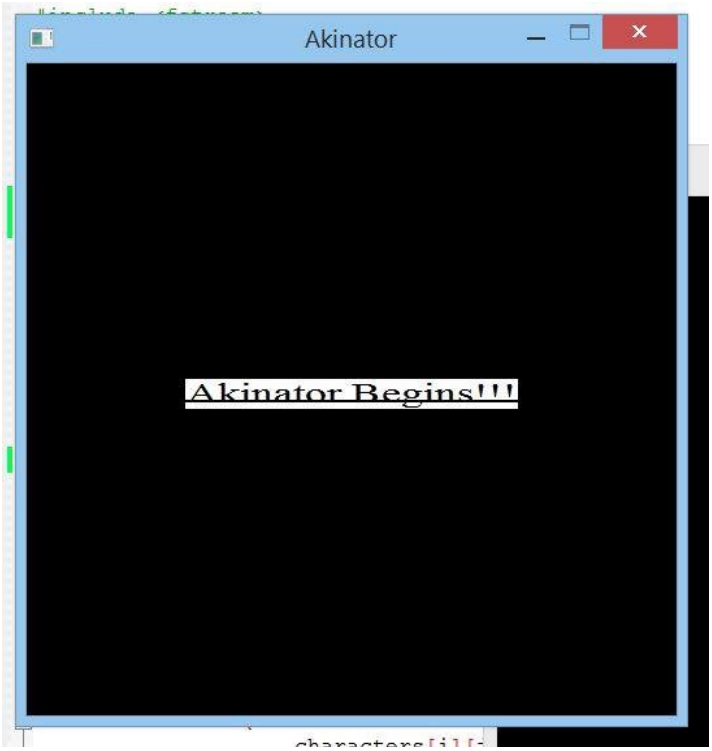
a) First Download WinBGIm from http://winbgim.codecutter.org/. Extract it
b) Copy graphics.h and winbgim.h files in include folder of your compiler directory
c) Copy libbgi.a to lib folder of your compiler directory
d) In code::blocks open Settings >> Compiler and debugger >>linker settings click Add button in link libraries part and browse and select libbgi.a file
e) In right part (ie. other linker options) paste commands -lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32
f) Click Ok

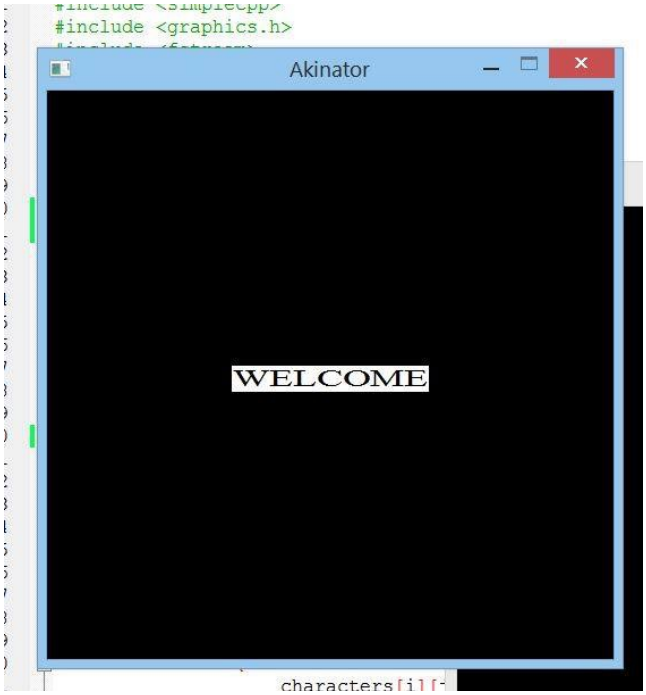
3. Open Simple Code Blocks(SimpleCPP)

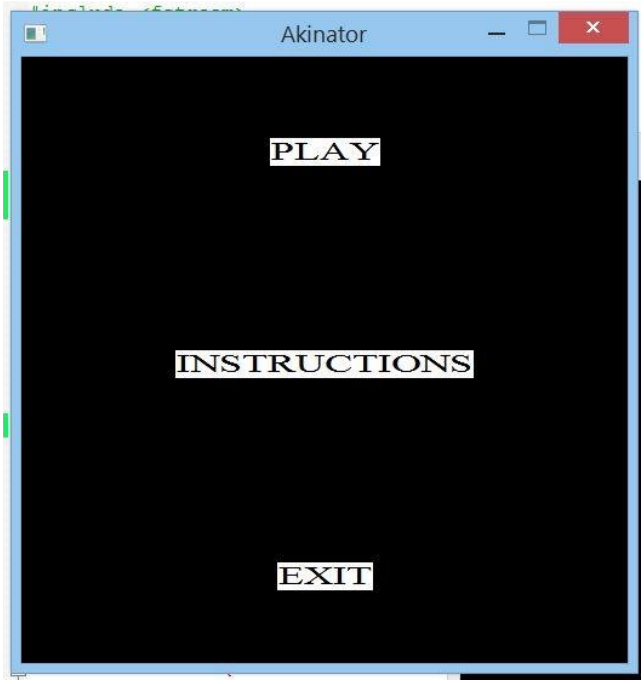
Navigate to source code directory and open source code. Run and compile it.

4. Implementation

Functionality:
This game is a single player game. The game starts with a welcome screen and afterwards a menu will appear which with the play, instructions and exit buttons. Click anywhere on the screen to go forward.

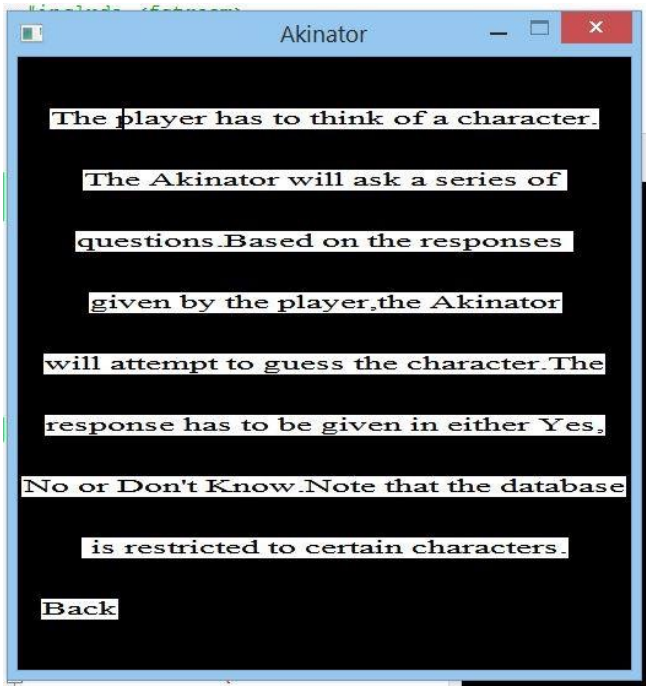






Clicking on the PLAY button starts the game

Clicking on INSTRUCTIONS button shows the instructions.



Clicking on EXIT button exits the game.

The player has to think of a character. On clicking the PLAY button the game will ask a series of questions which have to be answered in Yes, No or Don't Know. After asking some questions, the Akinator will give either of these results:

1. The name of the character.
2. No character has been found. What character did you think of?

In the event of the 2nd case occurring, the player has the option of entering the name of the character in another file, which can be later added to the database.

Note that the Akinator will give the correct response only if the player gives all the correct answers. A wrong answer for a character will eliminate all possibilities of that character being

guessed and no character will be shown. Also, the character has to be guessed from a pre-defined database which consists of Indian cricketers, Indian politicians, and Bollywood actors.

The player has to click within the dimensions of the boxes carefully; otherwise the wrong response will be recorded.

Supportability

1. Format: The code is formatted in the All Man style.

2. Naming convention used: Class, structure and function names - Camelcase starting with upper case letters.

Interfaces

1. User Interfaces

Terminal or Console to run the code.

2. Hardware Interfaces

Laptop with basic specifications.

3. Software Interfaces

Terminal or Console to run the code.

4. Communications Interfaces

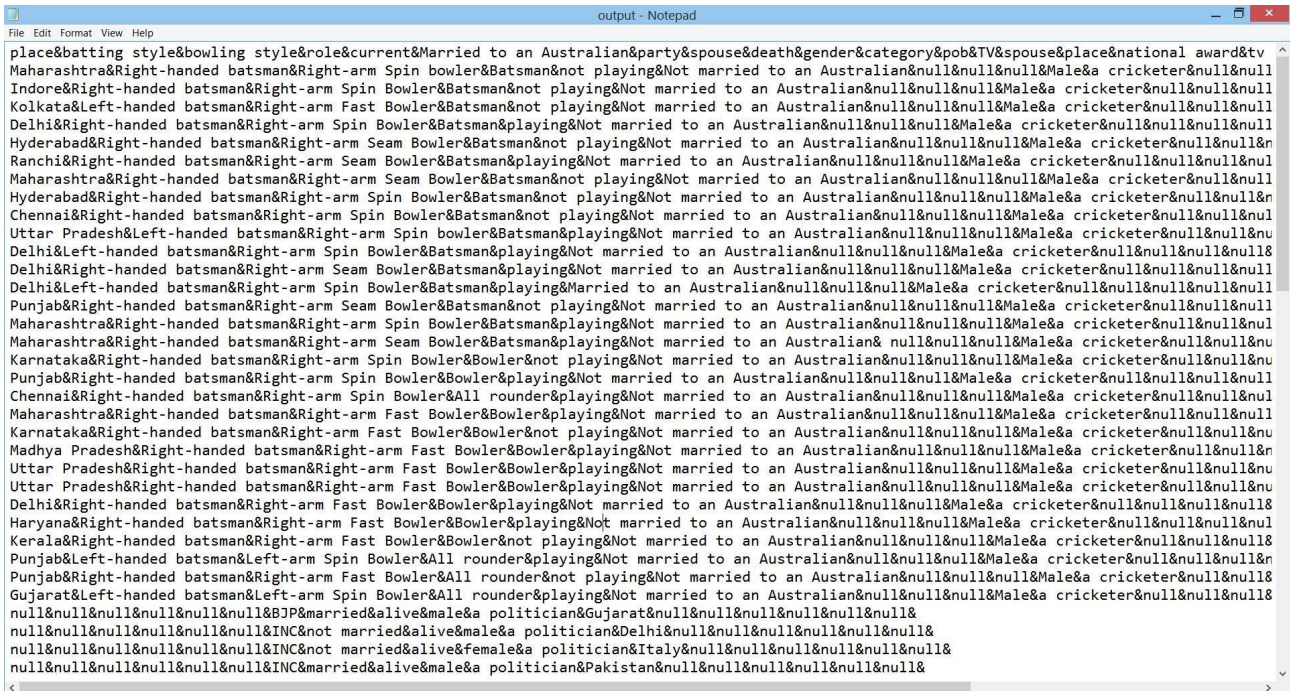
We are doing everything on single system. So, we don't need communication interface.

Design and Constraints

1. The algorithm of this game is based on decision trees. The questions are being traversed by using tree traversal. A lot of pain was encountered during designing the decision tree and determining the order in which questions will be asked.

The code for the next question to be asked was attached with each question. In case of the questions which branch into multiple questions, cases were included in the code using if-else structure.

2. The database had to be limited. We decided on Indian cricketers, politicians and Bollywood actors. The list of characters can be seen by the player in the source code folder(in file names.txt).



The database was scraped using BeautifulSoup package of Python. The Google Search direct results were scraped using Python and were written onto a file. Some manual editing was also done in designing the database. The database is such that each character can be differentiated.

Limitations

1. The game works on a limited database.
2. There is no allowance for wrong answers in the game. If the player enters one wrong answer, no player will be shown.
3. Many vague questions will be asked till the final character is reached due to the frequency approach followed i.e the variations in attributes are asked according to the frequency. For example, if the quality is gender, the variations will be male and female. Suppose the character to be guessed is female. If the number of males in the database is more, the question “Is your character a male” will be asked first. However if you click no, “Is your character a female” will be asked, which doesn’t make any sense in asking.

5. Test Cases

TEST CASE 1:

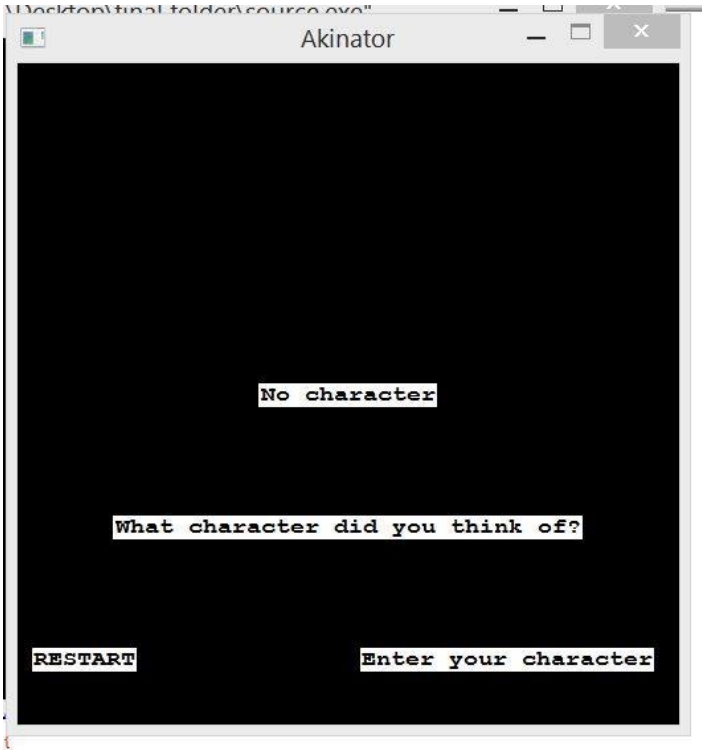
In the first test case we took our character to be Gautam Gambhir. The gameplay leading to Gautam Gambhir has been shown in the video.

TEST CASE 2:

In the second test case we took our character to be Smriti Irani. The gameplay leading to Smriti Irani has been shown in the video.

TEST CASE 3:

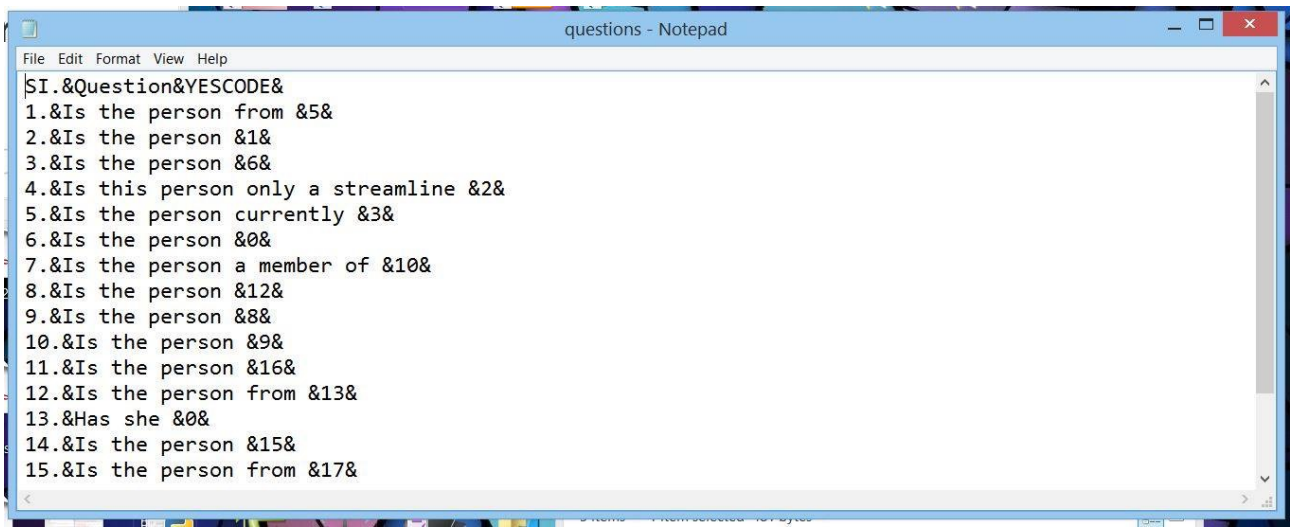
We checked for no character. The gameplay leading to no character has been shown in the video.



6. Challenges faced & solutions

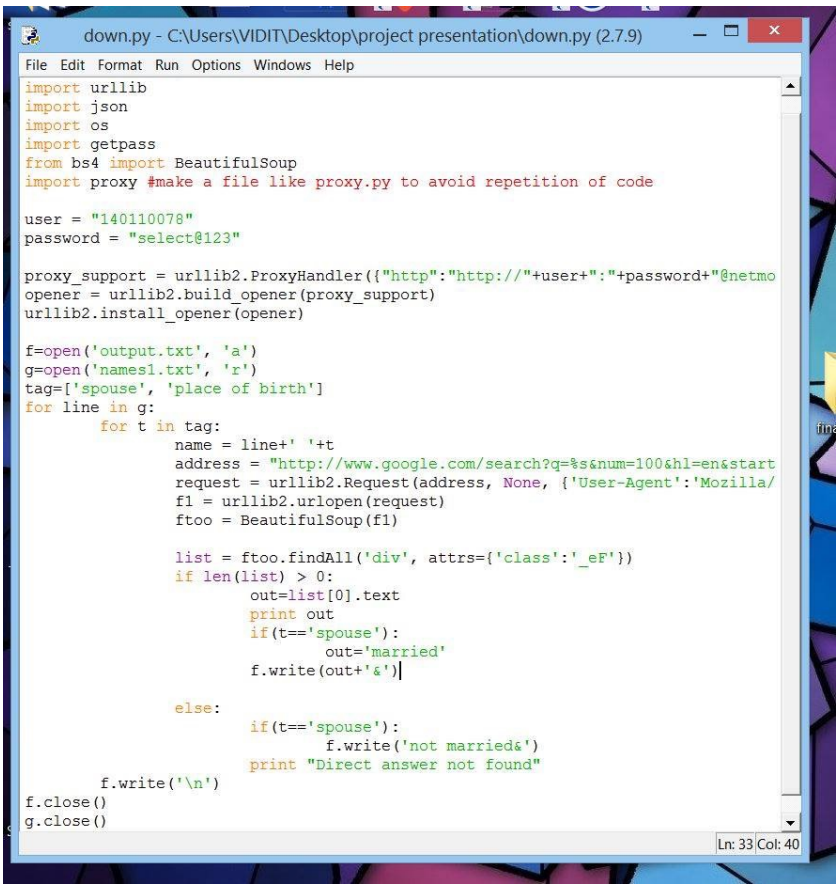
- The algorithm of the programme is based on decision trees. A problem was being faced involving how to design the decision tree i.e. how to go to the child questions and how to go through them.

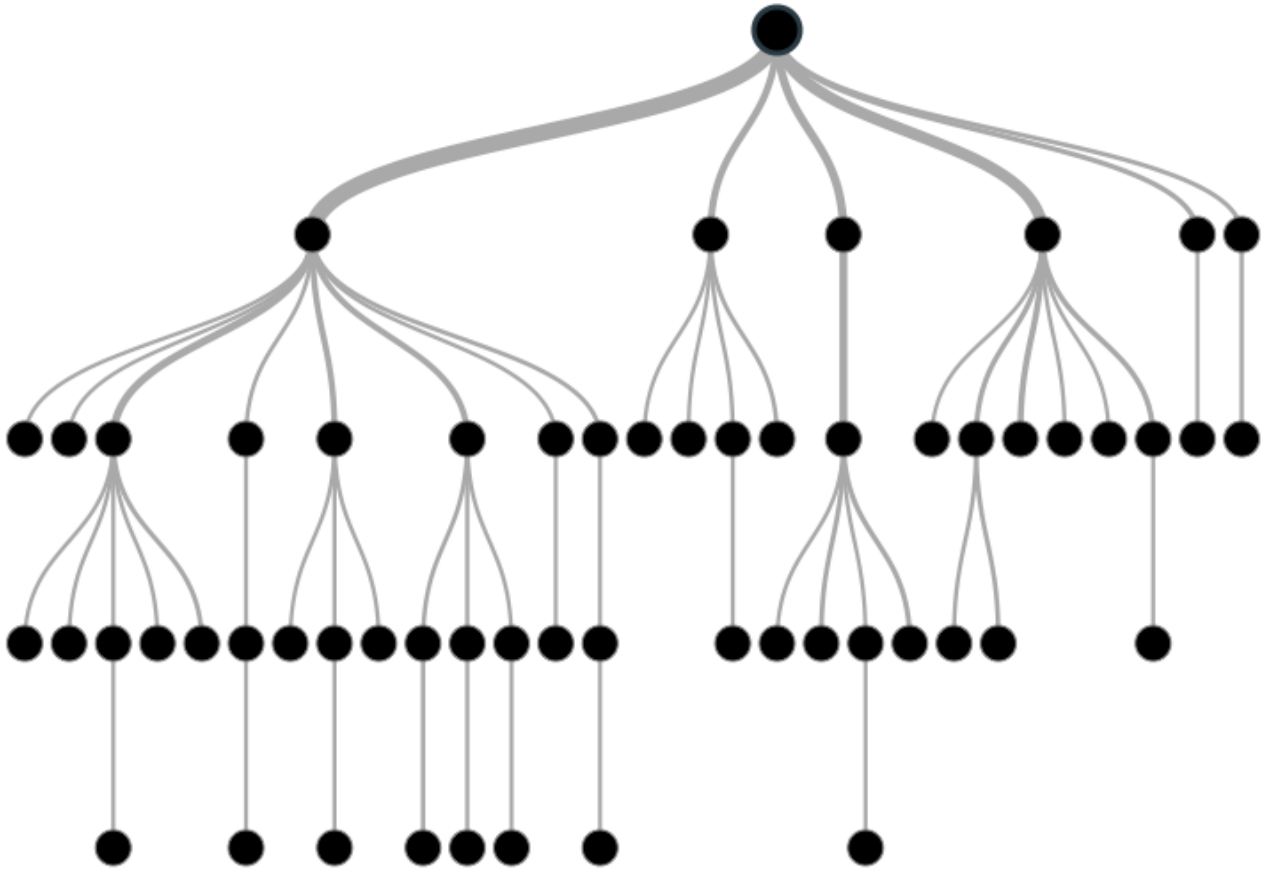
  This was solved by adding a YESCODE for every question i.e. the question number which the programme will go to if a YES answer is given.



- Deciding the type of characters and attributes the database would store, and designing the decision tree, was a tough task.

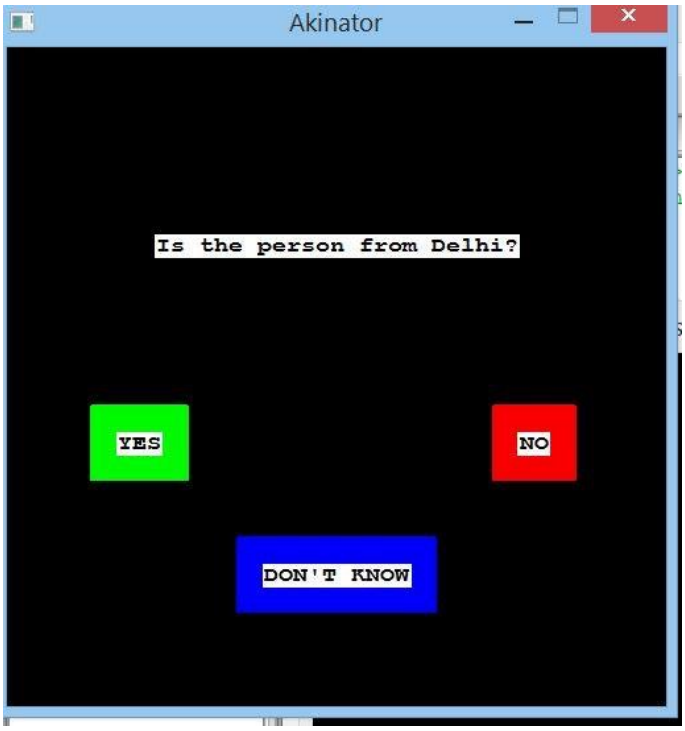
  We initially decided to scrape the attributes of cricketers and footballers from websites like cricinfo.com. However we were unable to do so. So we decided to take out some attributes using Google Direct search. This was done using BeautifulSoup library of Python.


down.py - C:\Users\VIDIT\Desktop\project presentation\down.py (2.7.9)
File Edit Format Run Options Windows Help
import urllib
import json
import os
import getpass
from bs4 import BeautifulSoup
import proxy #make a file like proxy.py to avoid repetition of code
user = "140110078"
password = "select@123"
proxy_support = urllib2.ProxyHandler({"http":"http://"+user+":"+password+"@netmo
opener = urllib2.build_opener(proxy_support)
urllib2.install_opener(opener)
f=open('output.txt', 'a')
g=open('names1.txt', 'r')
tag=['spouse', 'place of birth']
for line in g:
for t in tag:
name = line+' '+t
address = "http://www.google.com/search?q=%s&num=100&hl=en&start
request = urllib2.Request(address, None, {'User-Agent':'Mozilla/
f1 = urllib2.urlopen(request)
ftoo = BeautifulSoup(f1)
list = ftoo.findAll('div', attrs={'class':'_eF'})
if len(list) > 0:
out=list[0].text
print out
if(t=='spouse'):
out='married'
f.write(out+'&')
else:
if(t=='spouse'):
f.write('not married&')
print "Direct answer not found"
f.write('\n')
f.close()
g.close()
Ln: 33 Col: 40



- In case the player does not know the answer to a question, he should have the option of choosing a Don't Know option in addition to YES or NO.

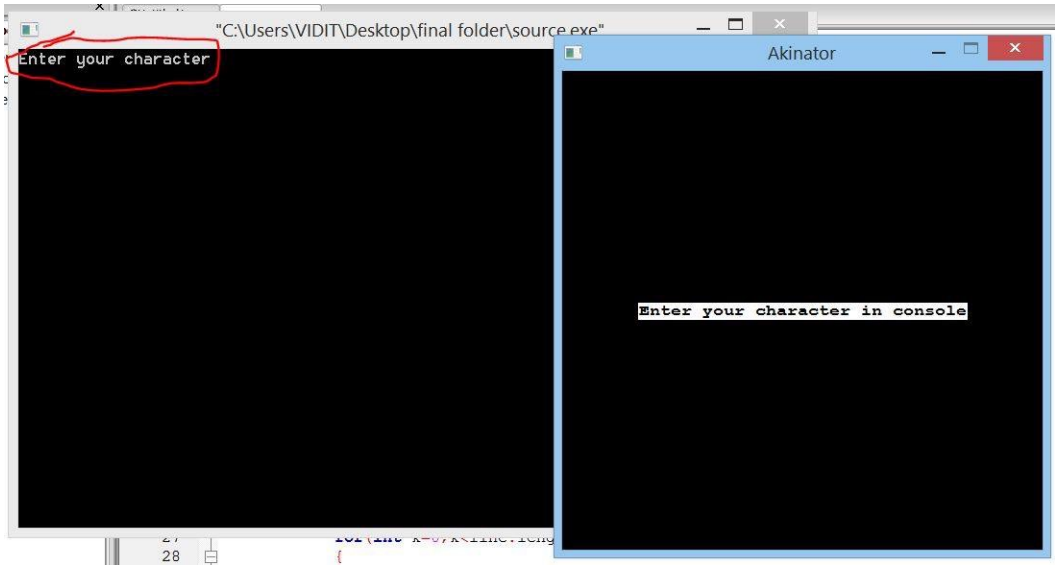
The Don't Know feature was implemented. If their is no variation in a quality left for a particular attribute, then control moves to next quality(question). If their are no more questions left, the game terminates. If none of the above cases happen, the loop will simply jump to next iteration and ask question

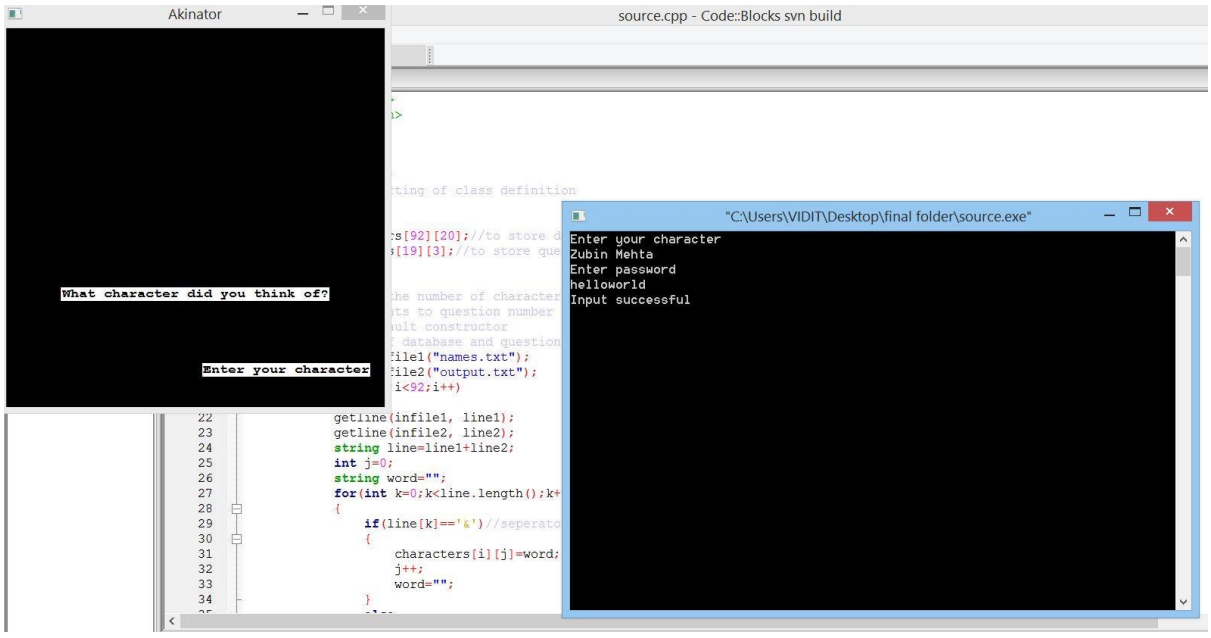


related to next quality.

- We had decided to add a feature which will allow a player to add a character to the database. The final addition into the database will be allowed by any of the team members of the project, after the entry is verified by any of the team member. The final entry into the database will happen through a password which will be known to the team members only. This is being done so that no bogus entries are made in the database.

This feature has been implemented. At the end of the game, if a character was not successfully guessed, the game will give an option of entering a character's name. The input has to be done on command line console. The instructions to enter will be given. The name will be added to a file only after entering a password. The name will not go in the main database. It can be entered afterwards.





Future Work

1. A probabilistic approach may be added so as to reduce the number of questions asked before determining the character.

2. A larger database can be designed.

3. Can store the responses given by different user which will help in determining the frequency of the path to the personality.

References

1. Akinator, the Game: en.akinator.com

2. www.cs.unm.edu/~pdevineni/papers/La.pdf

3. http://www.crummy.com/software/BeautifulSoup/

4. http://stats.espncricinfo.com/ci/engine/stats/index.html

5. http://forbesindia.com/lists/2013-celebrity-100/1439/1

6. http://www.mensxp.com/special-features/today/8053-26-best-indian-political-leaders-of-all-time.html

7. Link to download SimpleCpp: http://www.cse.iitb.ac.in/~ranade/simplecpp/

8. Link to download Borland BGI graphics(graphic.h package): http://winbgim.codecutter.org/

9. Link to instructions to include header file in SimpleCodeBlocks: http://stackoverflow.com/questions/20313534/how-to-use-graphics-h-in-codeblocks

10. Link to youtube video: https://www.youtube.com/watch?v=IH7kGmKO7Ec