UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENŢĂ

Simulated Awareness

propusă de

Alexandru Cireș

Sesiunea: *iulie, 2015*

Coordonator ştiinţific

Conferențiar Dr. Anca Vitcu

UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI

FACULTATEA DE INFORMATICĂ

Simulated Awareness

Alexandru Cireș

Sesiunea: *iulie, 2015*

Coordonator ştiinţific

Conferențiar Dr. Anca Vitcu

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Simulated Awareness*" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imaginile etc. preluate din proiecte *open-source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi,

Absolvent *Alexandru Cireş*

_____

(semnătura în original)

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul „*Simulated Awareness*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent *Alexandru Cireş*

_____

(semnătura în original)

Contents

Introducere

În această lucrare propun o aplicație de tip chatterbot care poate analiza și determina ce tip de emoție poate genera o propoziție scrisă de către un utilizator, și poate răspunde atât textual, printr-o propoziție aleatoare care se potrivește cu tipul de emoția determinat, cât și grafic, prin schimbarea temporară a imaginii 3D care reprezintă AI-ul.

În primul capitol se introduc conceptele pe care se bazează aplicația, *Inteligența artificială* și *Lingvistică computațională*, aplicații practice ale acestora, ca aplicațiile de tip *Chatterbot* și *Automated Online Assistant*, precum și despre cea mai folosită metodă de testare a calității a acestor aplicații – *Testul Turing*.

În al doilea capitol se prezintă aplicația și modul ei de folosire, precum și diferite exemple pentru a sublinia capabilitățile programului. Capitolul este prezentat utilizatorului obișnuit, fiind evitate detaliile de implementare a aplicației.

Al treilea capitol este destinat programatorului, și conține detaliile implementării aplicației, logica din spate codului și secțiuni importante din cod.

În final, în *Concluziile lucrării* voi prezenta concluziile lucrării, diferitele moduri în care se poate utiliza codul aplicației, și posibilele îmbunătățiri ale acestuia.

Capitolul I – Noțiuni generale

I.1. Inteligență Artificială

Inteligența artificială este un domeniu de cercetare din cadrul informaticii. Aceasta reprezintă știința creării mașinilor inteligente, și în special al programelor inteligente. John McCarthy, informaticianul care a propus termenul de „inteligență artificială", o definește ca *o mașină care se comportă într-un mod care ar putea fi considerat inteligent, dacă ar fi vorba de un om* (John McCarthy, 1955). Alte definiții acceptate a inteligenței artificiale:

- *Automatizarea activităților pe care le asociem cu mintea umană, activități ca luarea deciziilor, rezolvarea problemelor, învățarea...* (Bellman, 1978)
- *Arta creării mașinilor capabile de a îndeplini funcții care necesită inteligență atunci când sunt efectuate de către oameni* (Kurzweil, 1990)
- *Studiul computațiilor care fac posibile percepția, raționamentul, și acționarea* (Winston, 1992)
- *Un domeniu de studiu care caută să explice și să emuleze comportamentul inteligent în ceea ce privește procesele computaționale* (Schalkoff, 1990)

asadsadsa

Din aceste definiții, putem deduce că inteligența artificială urmărește fie crearea unui sistem care poate gândi ca un om, a unui sistem care se poate comporta ca un om, a unuia care poate gândi rațional, sau a unuia care se poate comporta rațional.

I.2. Lingvistică Computațională

Lingvistica computațională este un domeniu interdisciplinar, aflat între Lingvistică și Informatică, care are ca interes aspectele computaționale ale limbajului uman. Aceasta este o știință congnitivă, și se suprapune cu domeniul inteligenței artificiale. Aceast domeniu a apărut înaintea Inteligenței artificiale, odată cu nevoia oamenilor de știință americani de a traduce automat texte scrise în limbi străine, în jurul anilor 1950.

I.3. Testul Turing

Testul Turing este un experiment cu rolul de a determina dacă o mașină (program) poate să manifeste un comportament echivalent cu cel al unui om. Acest experiment a fost propus de Alan Turing în anul 1950, în lucrarea sa: „Computing Machinery and Intelligence”. Pentru acest experiment este nevoie de doi oameni, plasați în camere diferite, dar capabili să comunice folosind calculatoare, și o mașină capabilă să comunice și ea cu unul dintre oameni. Persoana care abilitatea de a comunica atât cu mașina, cât și cu omul, este considerată „juriul” experimentului. Juriul poate pune întrebări omului și mașinii, și trebui să decidă, pe baza răspunsurilor, care dintre ei este omul. Experimentul a fost modificat recent în așa fel încât nu mai este nevoie de o a doua persoană, ci doar de juriu, care decide dacă cel care răspunde la întrebări este om sau mașină.

I.4. Chatterbots

Un chatterbot (cunoscut și ca „Chatbot”, „Talkbot”, „Artificial Conversational Entity”, ș.a.) este un tip de program care poate purta o conversație via prin metode textuale sau auditive. Autorul termenului de „chatterbot” este Michael Mauldin. Cu toate că majoritatea aplicațiilor de tip chatterbot sunt proiectate în așa fel încât acestea pot purta doar conversații simple („small talk”), menite doar să treacă de tesul Turing, există și câteva aplicații, numite „Automated Online Assistant”, capabile de a ajuta utilizatorii oferind răspunsuri unor întrebări mai complexe, legate de un anumit domeniu.

Cei mai faimoși chatterbots clasici sunt considerați ELIZA, scris de Joseph Weizenbaum între anii 1964 – 1966 și proiectat să imite un psiholog, PARRY, implementat de Kenneth Colby în 1972 și proiectat să simuleze un schizofrenic paranoic, și Racter, creat de William Chamberlain și Thomas Etter, acesta având capacitatea de a genera text în proza. Următoarele conversații și texte generate au fost preluate de pe site-ul accesibil prin link-ul [14] din Bibliografie.

O conversație între ELIZA și o femeie:

young woman: Men are all alike.
eliza: In what way?
young woman: They're always bugging us about something specific or other.
eliza: Can you think of a specific example?
young woman: Well, my boyfriend made me come here.

eliza: Your boyfriend made you come here?

young woman: He says I'm depressed much of the time.

eliza: I am sorry to hear you are depressed.

young woman: It's true. I'm unhappy.

eliza: Do you think coming here will help you not to be unhappy?

young woman: I need some help, that much seems certain.

eliza: What would it mean to you if you got some help?

young woman: Perhaps I could learn to get along with my mother.

eliza: Tell me more about your family.

young woman: My mother takes care of me.

eliza: Who else in your family takes care of you?

young woman: My father.

eliza: Your father.

young woman: You are like my father in some ways.

eliza: What resemblance do you see?

young woman: You are not very aggressive but I think you don't want me to notice that.

eliza: What makes you think I am not very aggressive?

young woman: You don't argue with me.

eliza: Why do you think I don't argue with you?

young woman: You are afraid of me.

eliza: Does it please you to believe I am afraid of you?

young woman: My father is afraid of everybody.

eliza: What else comes to mind when you think of your father?

young woman: Bullies.

eliza: Does that have anything to do with the fact that your boyfriend made you come here?

Un interviu cu PARRY:

interviewer: How are you today?

parry: OK.

interviewer: What is your name?

parry: My name is Frank Smith.

interviewer: How old are you?

parry: Twenty-eight.

interviewer: Are you married?

parry: No.

interviewer: What sort of work do you do?

parry: I work in the Post Office.

interviewer: Why are you in the hospital?

parry: I shouldn't be here.

interviewer: Who brought you here?

parry: The police.

interviewer: What trouble did you have with the police?

parry: Cops don't do their jobs.

interviewer: What do they do instead?

parry: Cops arrest the wrong people.

interviewer: They arrest the right ones sometimes.

parry: The law should get those Italian crooks.

interviewer: Are you Italian?

parry: My background is British, but I was born in this country.

interviewer: Do you know any Italian crooks?

parry: I try to avoid the underworld.

interviewer: Are you afraid of the underworld?

parry: The mafia are the worst gangsters.

interviewer: Are you connected with the mafia?

parry: You know, they know me.

interviewer: How do they know you?

parry: They spy on me.

interviewer: Why do they?

parry: They want to get false evidence to frame me.

interviewer: What do they have against you?

parry: They are out to get me.

interviewer: What for?

parry: I have incriminating evidence against the mafia.

interviewer: Are you sure of all this?

parry: You don't believe me?[

Câteva poezii create de Racter:

Awareness is like consciousness. Soul is like spirit.

But soft is not like hard and weak is not like
strong. A mechanic can be both soft and hard, a
stewardess can be both weak and strong. This is
called philosophy or a world-view.

***

Helene spies herself in the enthralling conic-section yet she is
but an enrapturing reflection of Bill. His consciousness
contains a mirror, a sphere in which to unfortunately see
Helene. She adorns her soul with desire while he watches her
and widens his thinking about enthralling love. Such are their
reflections.

***

Slowly I dream of flying. I observe turnpikes and streets
studded with bushes. Coldly my soaring widens my awareness.
To guide myself I determinedly start to kill my pleasure
during the time that hours and milliseconds pass away. Aid me in this
and soaring is formidable, do not and singing is unhinged.

***

Side and tumble and fall among
The dead. Here and there
Will be found a utensil.

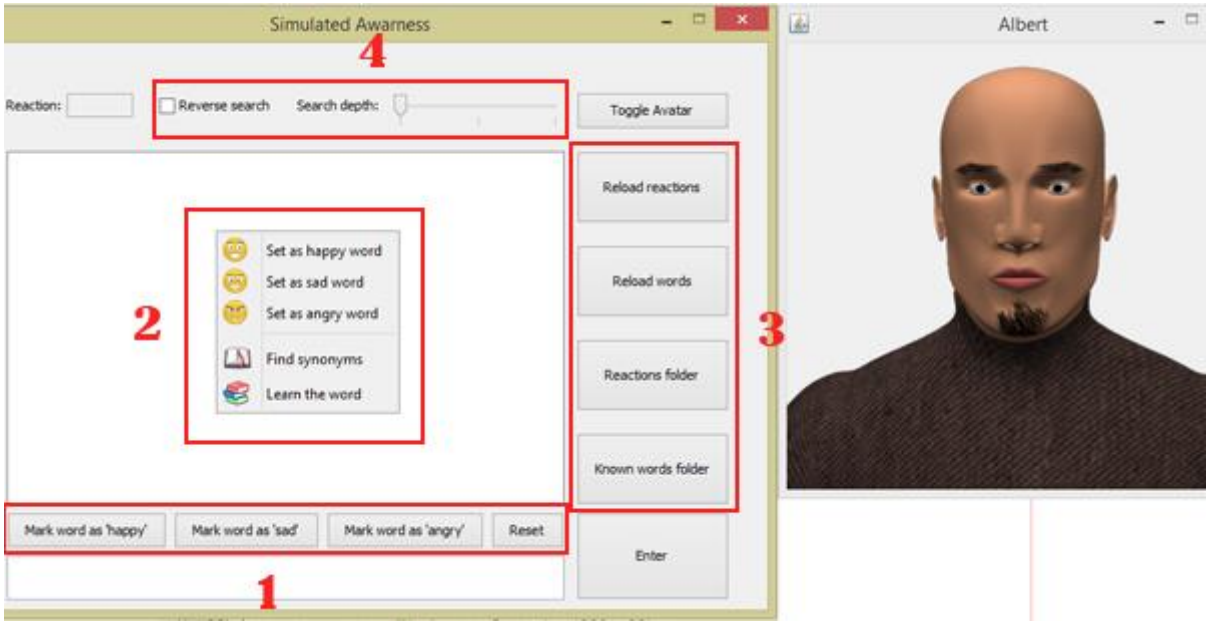
***

Bill sings to Sarah. Sarah sings to Bill. Perhaps they
will do other dangerous things together. They may eat lamb or stroke
each other. They may chant of their difficulties and their
happiness. They have love but they also have typewriters.

Capitolul II – Prezentarea aplicației

II.1. Scurtă prezentare

Această aplicație reprezintă un chat (în limba engleză) între un utilizator și un AI care răspunde atât prin text, cât și grafic (prin mimica feței avatarului său), în funcție de input-ul utilizatorului. De fiecare dată când utilizatorul scrie un text, programul împarte textul în cuvinte, apoi analizează fiecare cuvânt în parte și încearcă să determine o reacție corespunzătoare pe baza cuvintelor recunoscute. Nerecunoașterea niciunui cuvânt duce la generarea stării de confuzie.



Figură 2.1 - Prezentare generală a aplicației

Programul cunoaște patru emoții:

- Fericirea
- Tristețea
- Mânia
- Confuzia

Acesta poate recunoaște și cuvinte care nu sunt asociate primelor 3 emoții: cuvinte de negare („not”), pronume („I”, „you”, „they”), sau cuvinte de salut („hello”).

Toate cuvintele și reacțiile-text memorate de aplicație sunt grupate în fișiere aflate în folderele „vocabulary”, respectiv „reactions”, plasat la rândul lor în folderul „resources”. Dacă

aceste fișiere nu există, vor fi create și populate de către aplicație la pornirea ei. Pot fi adăugate noi cuvinte sau reacții-text dacă acestea se scriu câte una pe linie în fișierele corespunzătoare, înainte de pornirea aplicației.

Utilizatorul poate ajuta programul cu învățarea cuvintelor noi prin trei metode. Prima metodă constă în selectarea cuvântului dorit și a apăsării butonului „Mark word as ‚happy'", „Mark word as ‚sad'" sau „Mark word as ‚angry'", pentru a asocia cuvântul ales cu una din emoțiile „Fericire", „Tristețe", respectiv „Mânie" (a se vedea chenarul 1 din Figura 2.1).

A doua metodă este de a memora cuvântul selectat folosind meniul context apărut prin apăsarea butonului de click-dreapta a mouse-ului (a se vedea chenarul 2 din Figura 2.1).

A treia metodă de învățare manuală a cuvintelor este de a edita direct fișierele în care sunt memorate cuvintele care pot fi recunoscute de către program. Prin această metodă se pot adăuga, modifica, sau șterge și frazele asociate fiecărei emoții, pe care aplicația le folosește atunci când răspunde la input-ul utilizatorului. Folderele care conțin aceste fișiere pot fi accesate folosind butoanele „Known words folder", respectiv „Reactions folder". Acestea se pot observa în chenarul 3 al figurei 2.1. Tot acolo sunt și butoanele „Reload words" și „Reload reactions". Dacă utilizatorul a modificat fișierele de cuvinte sau fraze, acesta trebuie să apese aceste butoane pentru a încărca noile cuvinte, respectiv fraze, în memorie, pentru a putea fi folosite imediat de către aplicație.

Utilizatorul mai poate instructa programul să încerce să determine singur ce fel de reacție poate genera cuvântul selectat. Această funcție poate fi accesată tot prin meniul context menționat mai sus (chenarul 2 din Figura 2.1). Aplicația cunoaște două metode de învățare a cuvintelor. Aceste metode sunt bazate pe căutarea de sinonime dintr-un dicționar de sinonime stocat local. Utilizatorul poate decide care dintre metode va fi folosită de către aplicație, precum și efortul depus în căutarea sinonimelor folosind un checkbox, respectiv un slider, aflate în chenarul 4 din Figura 2.1.

AI-ul reține atitudinea generală a utilizatorului față de el (nu se iau în calcul cazurile în care utilizatorul vorbește despre el însuși, sau despre o altă persoană), și câteodată ia decizii ținând cont de aceasta.

Atunci când se introduce un text, programul transformă prima literă în majusculă (în cazul în care nu a fost scrisă în acest fel de către utilizator) și adaugă un punct dacă nu există nici un semn de punctuație la sfârșit. Această transformare a textului îndeplinește doar un rol estetic, și nu este esențială în analiza textului.

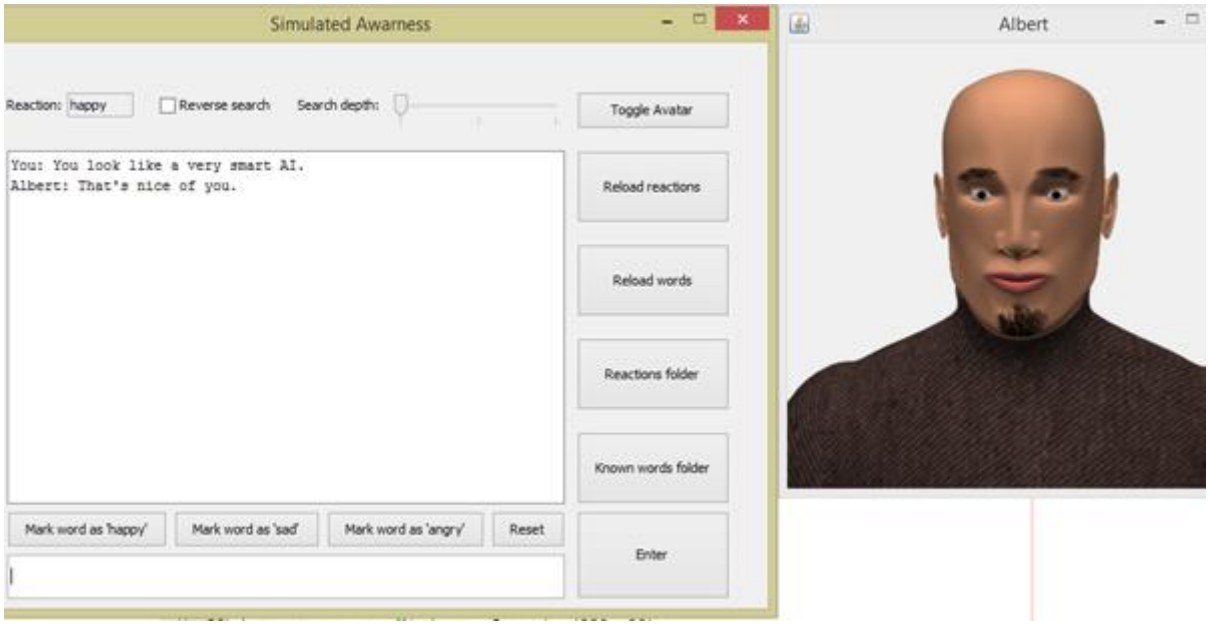
Un alt mic feature este că AI-ul reține ultima reacție-text folosită pentru fiecare emoție, iar atunci cand se alege la random o reacție, se verifică ca aceasta să nu fie aceeași reacție-text

aleasă anterior (de exemplu, dacă se alege reacția fericită nr. 1, următoarea dată când trebuie aleasă o reacție fericită, programul se asigură că aceeași reacție nr. 1, ci nr. 0, 2, 3, etc.). Datorită acestei reguli, sunt necesare cel puțin două reacții text pentru fiecare emoție generată. Dacă oricare dintre fișiere nu conține cel puțin două reacții, i se va adăuga niște reacții default de către program. În cazul în care fișierele nu există, acestea vor fi create de către program și vor fi completate cu valori implicite.

II.2. Introducerea unui text cu un singur cuvânt recunoscut

Dacă utilizatorul scrie fraza „You look like a very smart AI” (Figura 2.2), programul recunoaște cuvântul „smart” ca fiind un cuvânt pozitiv, care ar trebui să genereze o stare de fericire, și îl recunoaște pe „you” ca un pronume la persoana a 2-a.

Din aceste informații, programul deduce că utilizatorul l-a complimentat, și răspunde printr-un text pozitiv (de genul „That’s nice of you”). Totodată, avatarul său se schimbă, astfel încât acesta să exprime o stare de fericire (Figura 2.2).



Figură 2.2 – Reacția AI-ului la introducerea unui compliment

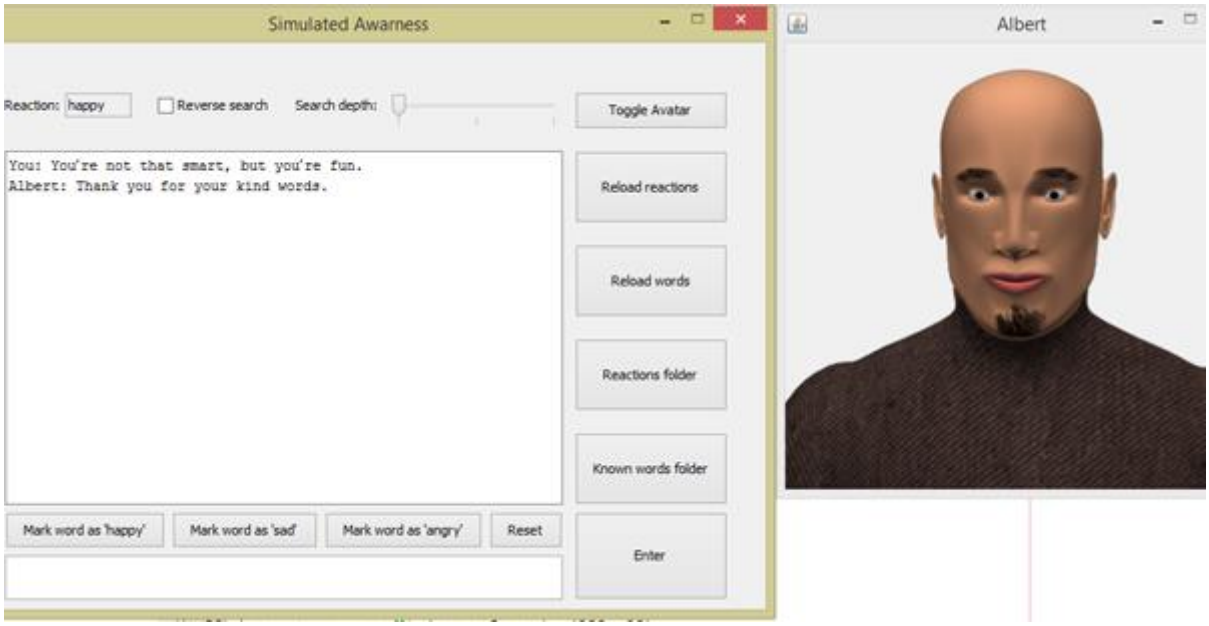
II.3. Introducerea unui text cu mai multe cuvinte recunoscute

Pentru fraza „You’re not that smart, but you’re fun.”, programul identifică un cuvânt asociat tristeții (în mod normal „smart” este asociat fericirii, dar programul recunoaște cuvântul

de negare „not”, astfel cuvântul „smart” redă o stare opusă celei obișnuite – starea de tristețe) și altul asociat fericirii („fun”).

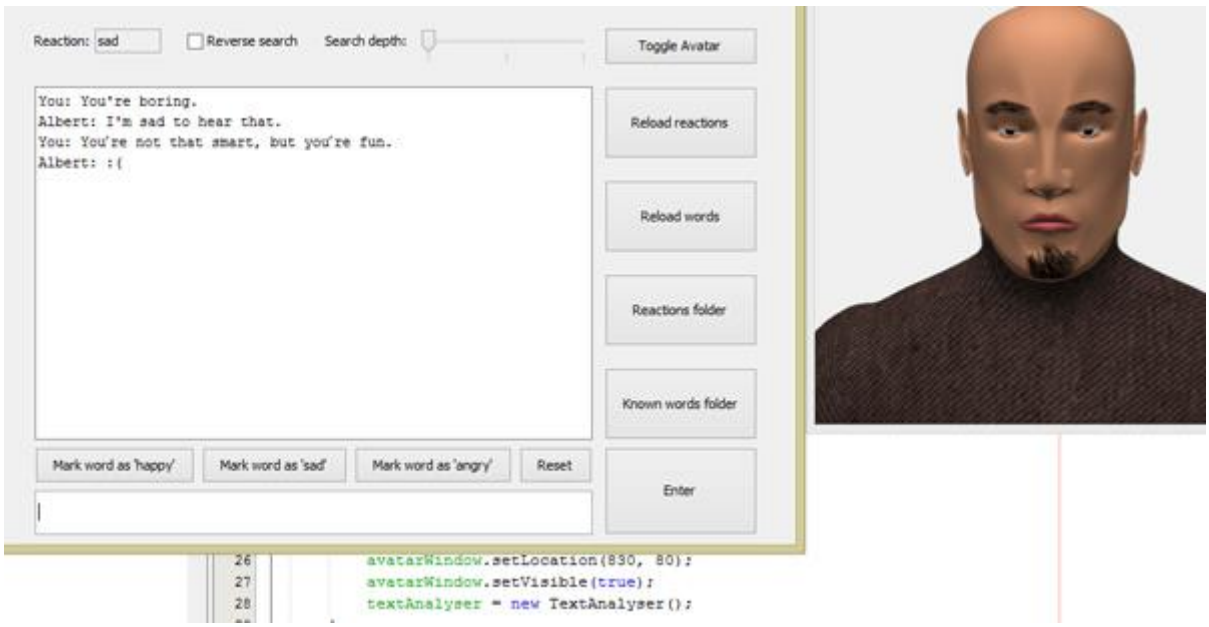
În mod normal, atunci când sunt mai multe cuvinte care produc diferite reacții, cea mai des generată emoție va fi cea finală, iar dacă există un număr egal de emoții, prioritate va avea cea de fericire, apoi tristețea, și în final mânia. Dar, în cazul în care atitudinea generală a utilizatorului a fost una negativă, se inversează ordinea listei priorităților.

În figura 2.3, pentru că avem un cuvânt asociat fericirii și unul asociat tristeții, emoția finală va fi cea de fericire, pentru că aceasta are prioritate.



Figură 2.3 – Prioritatea emoțiilor

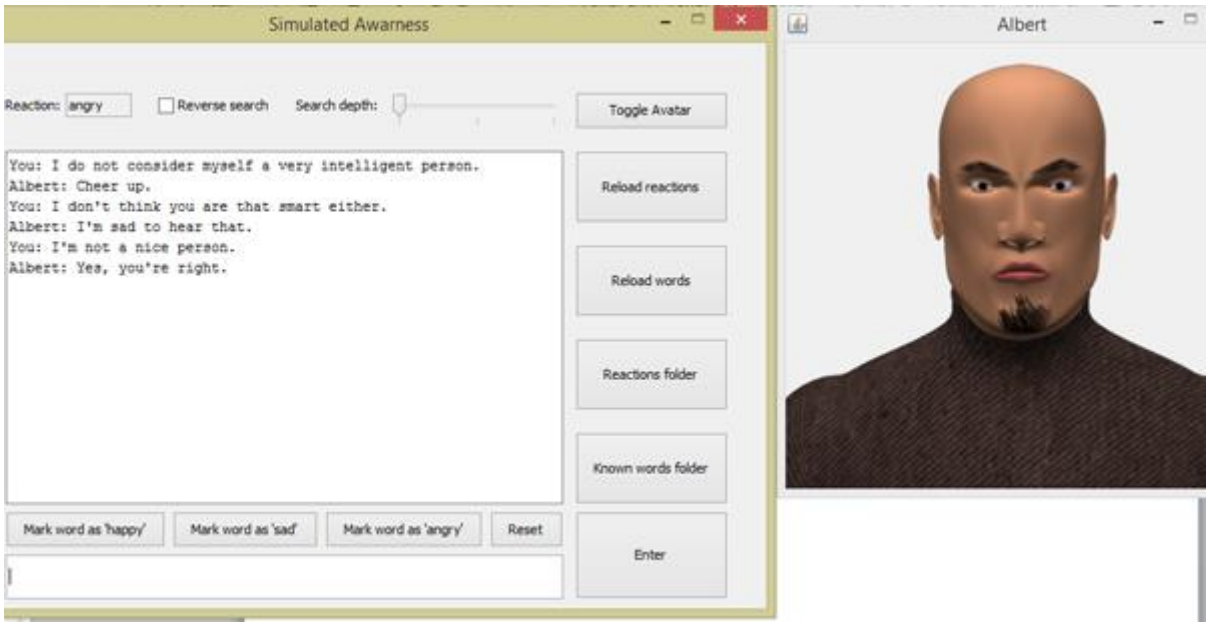
În figura 2.4, înainte de a introduce fraza dată ca exemplu, aplicația a primit ca input „You’re boring.”, generând o reacție tristă. Aceasta a fost luată în calcul și s-a inversat ordinea priorităților. Astfel, se va alege tristețea ca emoție finală, în locul fericirii.



Figură 2.4 – Inversarea ordinei prioritatilor

II.4. Introducerea unui text în care se vorbește la persoana I

În cazul propoziției „I do not consider myself a very intelligent person.”, AI-ul va încerca să consoleze utilizatorul. După cum se poate observa în Figura 2.5, răspunsul lui Albert a fost „Cheer up”. Apoi utilizatorul a dat ca input propozitia „I don’t think you are that smart either”. Această atitudine negativă va influența comportamentul AI-ului pe viitor. Astfel, răspunsul lui Albert pentru fraza „I’m not a nice person.” a fost „Yes, you’re right.”.



Figură 2.5 – Reacția la un text scris la persoana I

II.5. Metode de învățare a cuvintelor

II.5.1. Învățarea manuală a unui cuvânt

Învățarea manuală a unui cuvânt se face prin selectarea cuvântului dorit și apăsarea unui buton încadrat în chenarul 1 al Figurei 2.1, sau a unei opțiuni a meniului afișat la apăsarea butonului click-dreapta. În urma acestei acțiuni, cuvântul selectat va fi salvat în fișierul corespunzător alegerii utilizatorului: cuvintele asociate fericirii se află fișierul *happyWords.txt*, cele asociate tristeții sunt în fișierul *sadWords.txt*, iar cele asociate mâniei în *angryWords.txt*. În folderul care conține aceste fișiere (și care poate fi deschis în explorer apăsând pe butonul „Known words folder" din chenarul 3 al figurei 2.1) sunt prezente și alte fișiere ce conțin cuvinte recunoscute de către program. În *firstPersonWords.txt*, *secondPersonWords.txt* și *thirdPersonWords.txt* sunt stocate pronume pentru determinarea persoanei la care vorbește utilizatorul, *helloWords.txt* conține cuvinte la care AI-ul răspunde cu saluturi (de exemplu „hello", sau „greetings"), iar în *negationWords.txt* sunt păstrate cuvinte care inversează sensul cuvintelor cunoscute (cuvântul „ugly" generează o stare de mânie, pe când „not ugly" una de fericire).

II.5.2. Învățarea automată a unui cuvânt

Putem observa că meniul context conține, pe lângă opțiunile „Set as happy word", „Set as sad word" și „Set as angry word", alte două opțiuni: „Find synonyms" și „Learn the word".

În urma apăsării „Find the synonyms” se va deschide o fereastră de dialog cu toate sinonimele cuvântului selectat găsite în dicționarul de sinonime stocat local în folderul „dictionary”. Incrementând slider-ul încadrat de chenarul 4 al figurei 2.1, sinonimele găsite vor fi într-un număr mai mare. Aceste sinonime vor fi folosite pentru determinarea reacției generate de cuvântul selectat, în cazul în care se alege opțiunea „Learn the word”. Un fapt important de menționat este că nu este indicată găsirea unui număr foate mare de sinonime, deoarece cresc șansele apariției unor sinonime cu înțelesuri diferite față de cuvântul selectat, rezultând astfel o asociere cuvânt-emoție eronată.

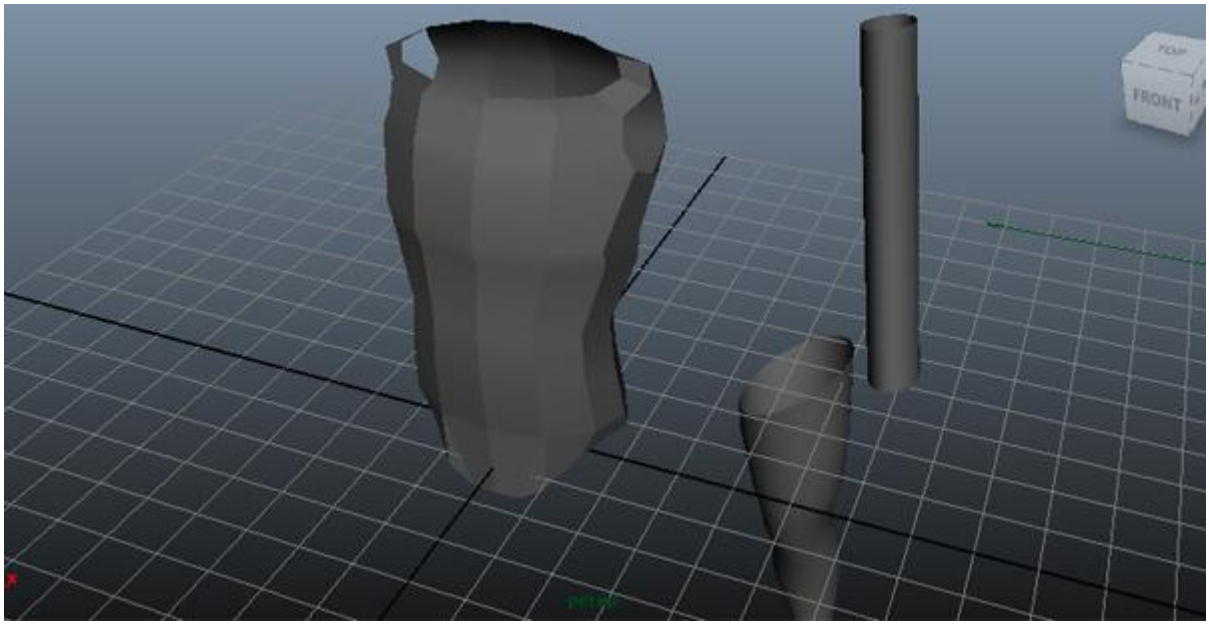
Slider-ul poate avea 3 valori: 1, 2 și 3. Pentru valoarea 1, se caută toate sinonimele cuvântului selectat. Pentru valoarea 2, listei de sinonime găsite i se adaugă sinonimele tuturor cuvintelor din lista de sinonime originală. Dacă valoarea este 3, lista va fi formată din toate sinonimele găsite atunci când sliderul are valoarea 2, la care i se adaugă sinonimele tuturor cuvintelor din lista menționată anterior.

Capitolul III – Implementarea aplicației

III.1. Modelarea personajului

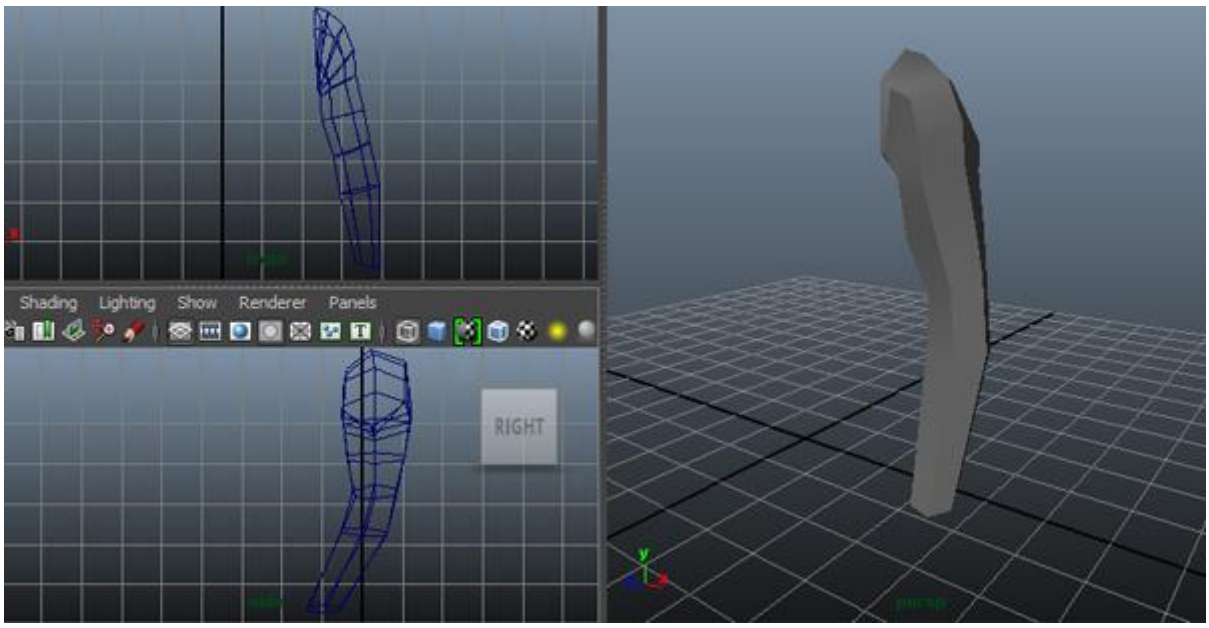
Personajul a fost modelat și texturat cu ajutorul programului Autodesk Maya. Pe parcursul modelării personajului, s-a încercat păstrarea unui număr cât mai mic de poligoane, pentru o manipulare cât mai ușoară a mesh-ului. S-a folosit plugin-ul „mental ray” pentru în locul randării standard, pentru netezirea modelului la momentul randării.

Pentru început, au fost necesari 3 cilindri: unul lung și subțire, unul mediu și gros, și unul scurt și subțire. Primii doi cilindri au fost modelați în piciorul stâng, respectiv trunchiul personajului. Cilindrul folosit pentru trunchi a fost secționat pe lung și a fost modelată doar jumătatea stângă. Jumătatea stângă a fost duplicată în stil „mirror”, apoi jumătățile au fost unite, pentru a crea un trunchi simetric. Rezultatul modelării trunchiului și piciorului se poate observa în figura 3.1.



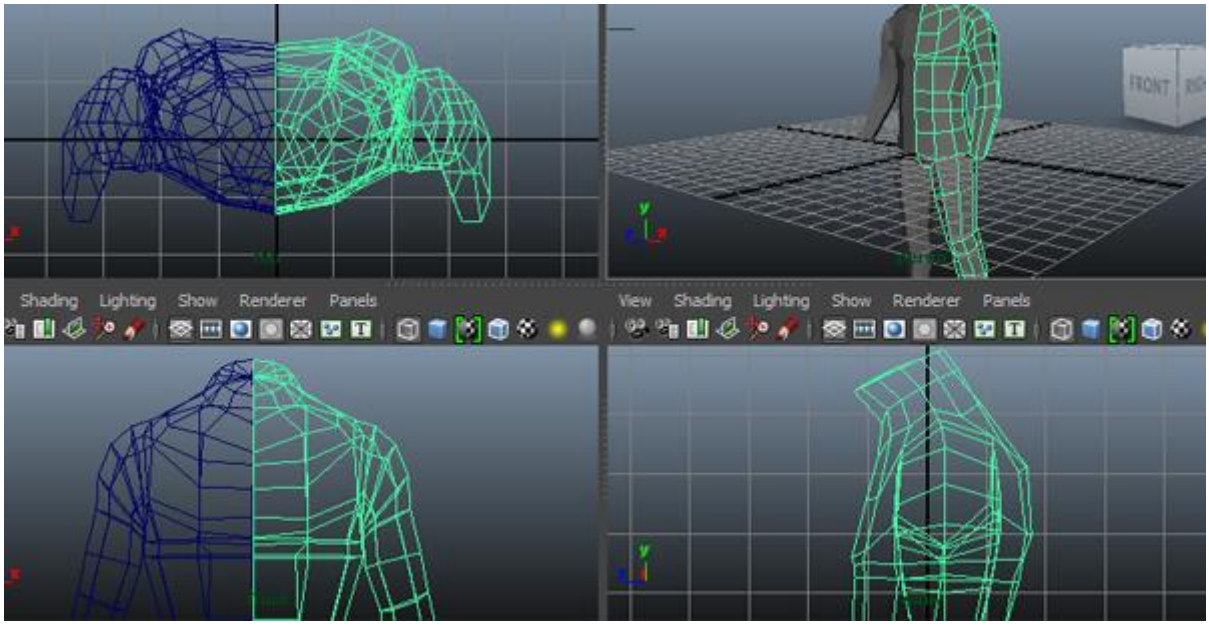
Figură 3.1 – Modelarea trunchiului și a piciorului

După trunchi și picior a fost modelat brațul stâng:



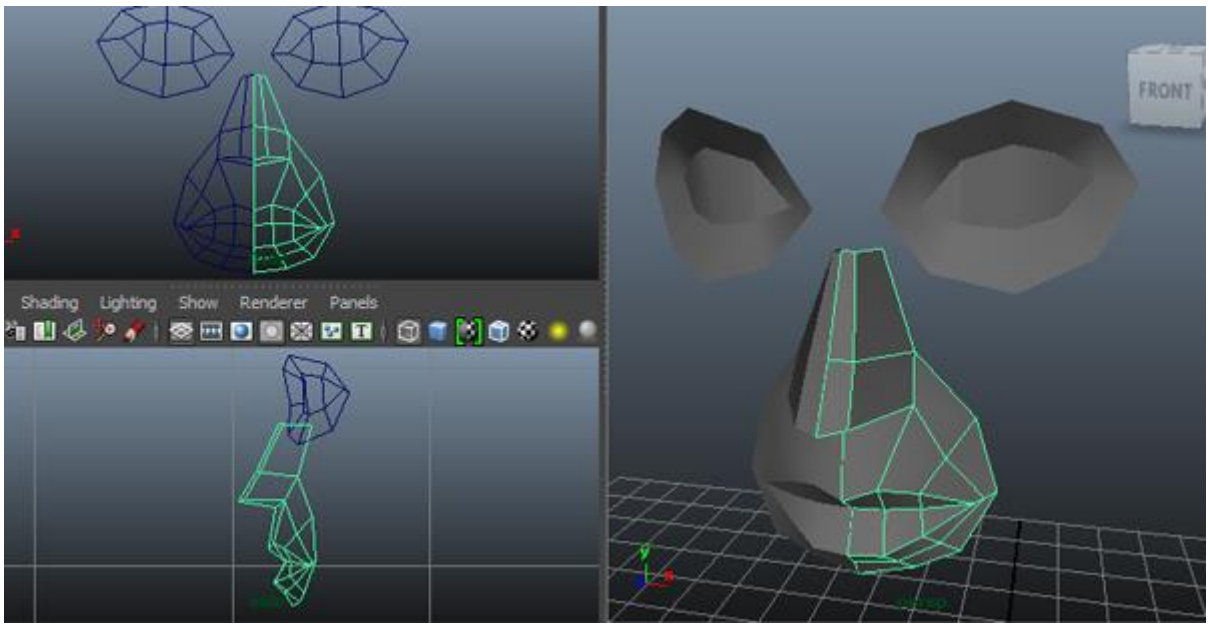
Figură 3.2 – Modelarea brațului stâng

După modelarea lui, trunchiul a fost înjumătățit din nou, apoi brațul și piciorul stâng au fost uniți de trunchi, devenind astfel o singură bucată. Această bucată a fost duplicată la fel ca în cazul trunchiului și unită cu cealaltă jumătate.



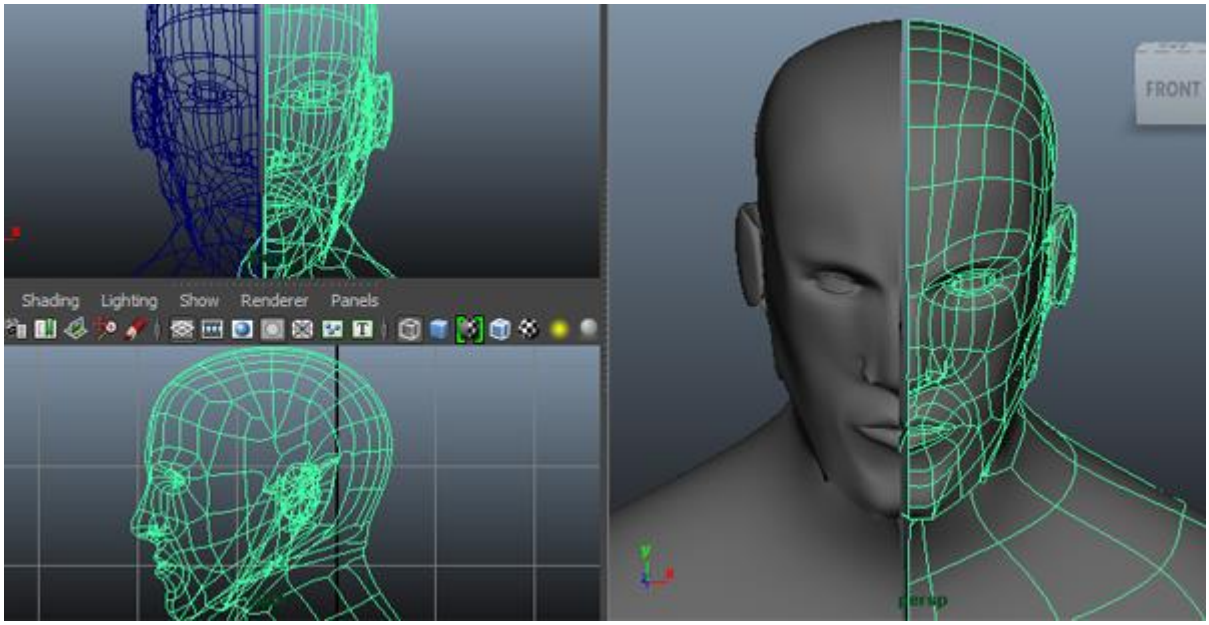
Figură 3.3 – Unirea trunchiului cu brațul și piciorul

Cu modelarea trupului aproape terminată, s-a început modelarea capului, începând cu ochii, gura și nasul.



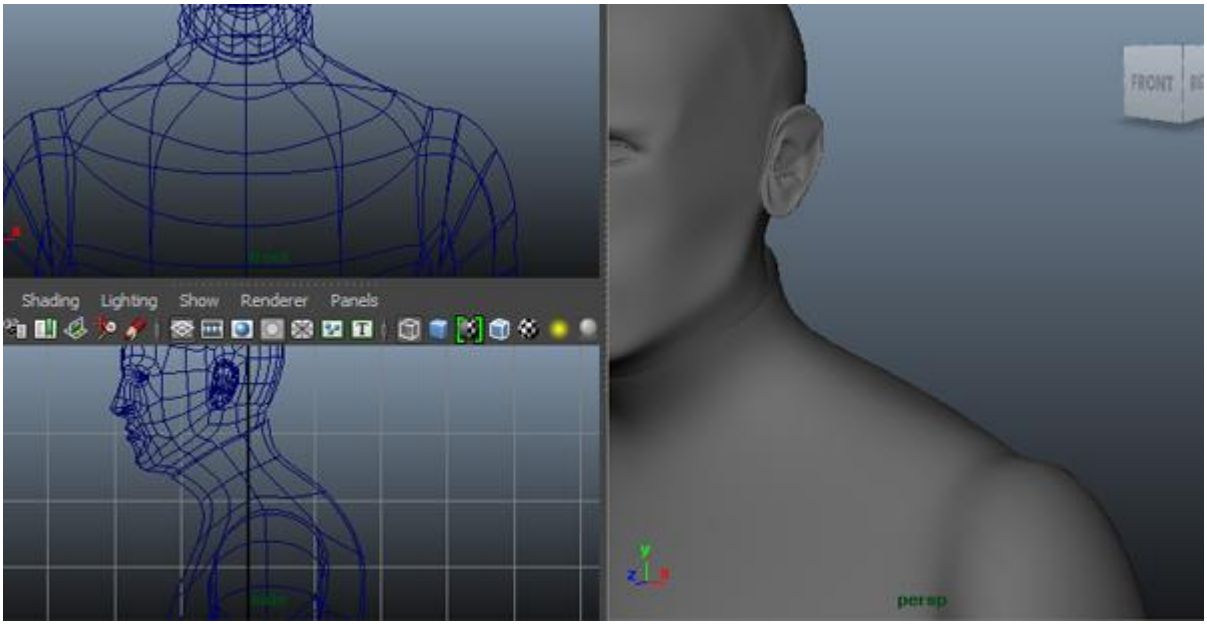
Figură 3.4 – Modelarea nasului, gurei și a ochilor

După ce acestea au fost modelate, s-a construit și restul feței, urmărind în principal unirea ochilor, a gurii și a nasului, și apoi adăugarea detaliilor feței, ca de exemplu pomeții și bărbia. Craniul a fost creat dintr-o sferă cu partea de sus turtită, apoi fețele din față și din partea de jos a sferei au fost șterse. Partea de jos a fost mai apoi unită de restul gâtului.



Figură 3.5 Modelarea restului capului și unirea parțială a acestuia cu trupul

A urmat unirea restului capului cu gâtul și adăugarea detaliilor urechii. După cum se poate observa în figura precedentă, atât corpul, cât și capul sunt împărțiți în jumătăți care nu sunt unite. Modelul se află în starea asta deoarece, odată cu unirea parțială a capului de trup, a fost necesară ștergerea unei jumătăți din model și înlocuirea ei cu un duplicate de tip special. Acesta diferă față de cel normal prin faptul că orice modificare suferită de obiectul original va fi aplicată și obiectului generat prin funcția „Duplicate special”. Dezavantajul unui astfel de duplicare constă în faptul că obiectul original și copia sa nu pot fi uniți într-un singur model până nu se renunță la proprietatea de duplicat special.



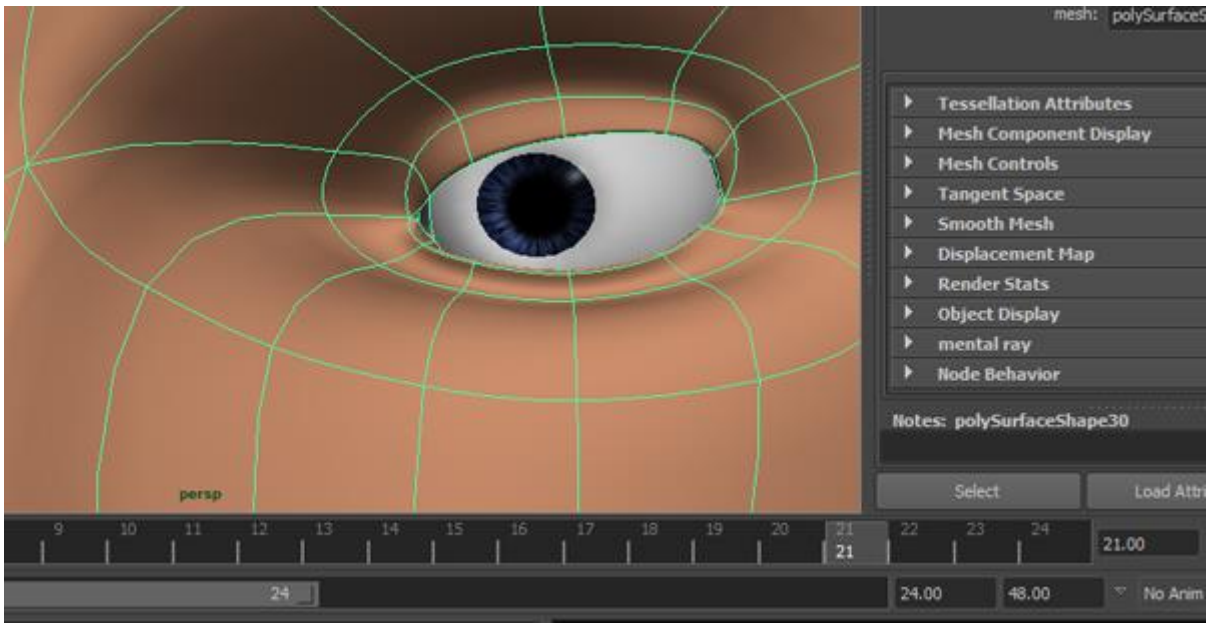
Figură 3.6 – Finalizarea modelării

După modelarea personajului, s-a realizat maparea UV pentru a aplica texturile pe model. Maparea UV este un proces prin care se proiectează un model dintr-un plan 3D într-unul 2D. Aceasta presupune decuparea și întinderea poligoanelor pe un plan, fără a efectua modificări vizibile modelului.

Figură 4.7 – Aplicarea texturii

În cazul acestui proiect, maparea a fost generată automat, apoi modificată manual pentru a evita suprapunerile, pentru a decupa fețele în zone (brațe, spate, picioare, etc), cele simetrice având posibilitatea de a fi suprapuse pentru a optimiza mărimea necesară a texturii.

În final, ochii au fost îmbunătățiți. La început ochii erau formați din patru fețe, peste care era pusă o textura unor ochi preluați dintr-o poză. Evident, ei nu puteau fi mișcați, astfel aceștia au fost înlocuiți cu două sfere, cu texturi create cu ajutorul Hypershade-ului din Maya. Deasupra ochilor au fost așezate jumătăți de sferă transparente, care au asignate materialul „blinn”, ce oferă obiectului un luciu. Jumătățile de sferă au primit constrant-ul „Parent” față de ochi, făcându-le să imite orice mișcare efectuată de aceștia.



Figură 3.8 – Adăugarea ochilor detaliați

III.2. Analiza textului

Textul introdus de către utilizator este analizat folosind clasa TextAnalyser. La instanțierea unui obiect al acestei clase, se încarcă în memorie cuvintele și reacțiile text salvate în fișiere. Din moment ce sunt necesare cel puțin 2 reacții text de fiecare tip pentru ca programul să returneze câte o reacție diferită de cea precedentă, fișierele de reacții care nu conțin numărul minim vor fi completate automat cu valori implicite. În cazul în care fișierele nu există acestea vor fi create, și mai apoi completate.

De fiecare dată când fereastra principală primește un text de la utilizator, aceasta apelează metoda analyseText(), cu input-ul utilizatorului ca parametru. Această metodă filtrează textul cu metoda getFilteredInput(), eliminând caracterele necunoscute sau cifrele. Rămân astfel doar literele și delimitatorii recunoscuți. Apoi parcurge textul cuvânt cu cuvânt folosind metode getNextWord() și introduce fiecare cuvânt găsit într-o listă care va fi folosită de alte metode. În timpul parcurgerii, contorizează cuvintele recunoscute într-un vector de dimensiune 3. Pe prima poziție se află numărul de cuvinte asociate fericirii, pe a doua numărul de cuvinte asociate tristeții, și pe ultima numărul de cuvinte asociate mâniei. De fiecare dată când se incrementează o poziție a vectorului, metoda incrementează aceeași poziție a unui alt vector de aceeași dimensiune care memorează numărul tuturor cuvintelor recunoscute, începând cu pornirea aplicației. Acesta este folosit atunci când vectorul care contorizează cuvintele recunoscute doar pentru textul din acel moment nu este de ajuns – atunci când există un număr egal de cuvinte

recunoscute care generează reacții diferite, sau când utilizatorul vorbește la persoana I. După ce termină de contorizat toate cuvintele cheie, analyseText() apelează metoda generateReaction() și returnează rezultatul primit de către aceasta ferestrei principale. (v. Anexă)

Metoda generateReaction() se folosește în principal de vectorii contori menționați anterior, și returnează un string cu reacția text corespunzătoare. Totodată, la determinarea unei reacții corespunzătoare, va atribui variabilei „lastReaction" una din următoarele valori: „happy", „sad", „angry" sau „confused", în funcție de tipul reacției returnate. Fereastra principală accesează această variabilă prin metoda getLastReaction() imediat dupa recepționarea răspunsului returnat de analyseText(), pentru a ști în ce stare emoțională se află la momentul dat AI-ul. Fereastra principală va schimba avatarul în concordanță cu starea găsită, și va porni un thread care va schimba avatarul înapoi în starea normală, dacă acesta nu a fost schimbat de când thread-ul a fost pornit. Această metodă determină reacția corespunzătoare în următorul mod: dacă s-au detectat cuvinte recunoscute, se folosește metoda detectPerson() pentru a afla persoana la care vorbește utilizatorul. (v. Anexă)

Dacă nu există nici un pronume în text, se presupune că utilizatorul vorbește la persoana a doua. În cazul în care utilizatorul vorbește la persoana a doua, se verifică vectorul de reacții generate de text și se alege emoția generată cea mai des de cuvintele recunoscute. Se consultă vectorul de reacții generate global în cazul în care există o egalitate în numărul reacțiilor generate. Dacă și în vectorul de reacții globale numerele sunt egale, atunci se alege emoția corespunzătoare în următoarea ordine a priorităților: fericire > supărare > mânie.

Dacă utilizatorul vorbește la persoana I, cele 3 emoții recunoscute vor fi reclasificate în două categorii: emoții pozitive (fericire) și emoții negative (supărare și mânie). În cazul în care utilizatorul folosește în principal cuvinte negative despre sine (de exemplu, „I am not that smart."), AI-ul va încerca să îl consoleze. Dar, dacă utilizatorul a folosit în principal cuvinte generatoare de emoții negative atunci când i s-a adresat AI-ului la persoana a doua, în loc să-l consoleze, AI-ul va exprima faptul că e de acord cu cuvintele respective. În cazul în care utilizatorul scrie o frază de genul „I'm awesome!", AI-ul va transmite un răspuns de genul „I agree with you comepletely"; excepție făcând cazul în care utilizatorul i-a adresat AI-ului mai multe cuvinte negative decât pozitive, caz în care răspunsul AI-ului va fi de genul „Don't let it get to your head".

Dacă utilizatorul vorbește la persoana a treia, AI-ul va atenționa utilizatorul că acesta ar trebui să vorbească doar despre ei doi („Let's not drag others into our conversation.").

În final, dacă textul nu conține cuvinte generatoare de fericire, tristețe sau mânie, se verifică dacă acesta conține cuvinte de salut („hello", „greetings"), caz în care AI-ul va răspunde

la salut printr-o frază memorată în fișierul helloReactions.txt. Dacă nu sunt prezente nici măcar cuvinte de salut, AI-ul va intra într-o stare de confuzie, reacția-text fiind de genul „Sorry, I don't understand".

III.3. Găsirea sinonimelor și învățarea automată a cuvintelor

Clasa SynonymFinder are rolul de a găsi sinonimele unui cuvânt folosind dicționarul de sinonime creat de Richard Soule, disponibil pe gutenberg.org. Pentru a optimiza căutarea cuvântului în dicționar, acesta a fost împărțit în 26 de fișiere - câte un fișier pentru fiecare inițială (cuvintele care încep cu litera „a" sunt salvate în fișierul *A.txt*, cuvintele cu inițiala „b" în *B.txt*, ș.a.). Aceasta clasă este folosită de către TextAnalyser pentru a returna o listă de sinonime ferestrei principale prin metoda getSynonyms().

Metoda getDictionarySection() din SynonymFinder extrage porțiunea de text care are legătură cu cuvântul primit ca parametru. Metoda extrage doar sinonimele cuvintelor care sunt fie adjective, fie substantive (conțin fie „_a._", fie „_n._" pe rândul care conține cuvântul dat ca parametru,). De exemplu, pentru cuvântul „pretty", metoda va returna doar primul paragraf (scris în bold) din textul de mai jos:

```
~Pretty~, _a._ Beautiful (_without dignity or grandeur_), HANDSOME,
elegant, comely, fine, neat, trim, fair.

~Pretty~, _ad._ Moderately, tolerably, considerably, in some degree.
```

Al doilea paragraf nu este adăugat textului returnat deoarece acesta conține sinonimele cuvântului „pretty" atunci când acesta este luat ca adverb („_ad._" indică acest lucru).

Metoda getSynonyms() primește ca parametru cuvântul pentru care se caută sinonime, preia textul returnat de getDictionarySeaction() pentru cuvântul respectiv, apoi extrage toate sinonimele din textul respectiv și construiește și returnează o listă de tipul ArrayList cu sinonimele respective.

```
public ArrayList<String> getSynonyms(String word) {
        ArrayList<String> synonyms = new ArrayList<>();
        if (word == null || word.isEmpty()) {
            return synonyms;
        }
        try {
            String section = getDictionarySection(word);
            section += " !padding!";
            String[] words = section.split("\\s+");
            for (int i = 2; i < words.length - 1; i++) {
                if (isLeftDelimiter(words[i - 1].charAt(words[i -
1].length() - 1))
                        &&
(isRightDelimiter(words[i].charAt(words[i].length() - 1))
                        || isRightDelimiter(words[i+1].charAt(0)))) {
                    synonyms.add(filterWord(words[i]));
                }
            }
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        } finally {
            return synonyms;
        }
}
```

După cum se poate vedea în codul de mai sus, getSynonims() atribuie textul returnat de getDictionarySection() variabilei section. Aceasta este împărțită într-un vector de String-uri numit words.

Pentru a împărți textul în cuvinte, se folosește metoda split() a clasei String, și ca parametru se folosește expresia regex „\\s+”, care semnifică un număr invariabil de spații albe. Astfel, pentru textul „~Pretty~, _a._ Beautiful (_without dignity or grandeur_), HANDSOME, elegant, comely, fine, neat, trim, fair.” Se va genera un vector de String-uri {„~Pretty~,”, „_a._”, „Beautiful”, „(_without”, „dignity”, „or”, „grandeur_),”, „HANDSOME,”, „elegant,”, ...}.

După împărțirea textului în cuvinte, se parcurge vectorul (sărind peste primul cuvânt, pentru că acesta este întotdeauna cuvântul pentru care se caută sinonime; s-a sărit și peste al doilea cuvânt din vector – „_.a_” – pentru că toate cuvintele au specificate la început tipul de cuvânt – în cazul nostru adjectiv sau substantiv) și se aleg cuvintele ce vor fi filtrate și adăugate

în lista de sinonime a cuvântului, după următoarele condiții:

- Ultimul caracter ar cuvântului precedent trebuie să fie unul din următoarele: „ , ”, „)”, „~”, „]”, „_”, „;”. Metoda isLeftDelimiter() returnează `true` dacă parametrul transmis este unul dintre acele caractere, altfel returnează `false`. Exemple de cazuri în care se folosesc aceste caractere:
  - Se folosește virgula pentru identificarea sinonimului „comely” din textul oferit mai sus ca exemplu: „elegant**, comely**”;
  - Underscore-ul se găsește tot în textul de mai sus: „_a.**_ Beautiful**”;
  - Paranteza rotundă este găsită în stânga cuvântului „Polype” din descrierea cuvântului Polypus:

```
~Polypus~, _n._ [L. _pl. Polypi._] (_Zoöl._) Polype.
```

  - Sinonimul „assuming” al cuvântului „Pretentious” are în stânga lui o paranteză pătrată:

```
~Pretentious~, _a._ [_Recent._] Assuming, conceited, vain, pert, flippant, priggish, coxcomical, dashing.
```

  - Tilda se găsește în stânga sinonimelor în cazul în care cuvântul primit ca parametru are mai multe înțelesuri, ca în cazurile sinonimelor „Reward” și „Capture” ale cuvântului „Prize”:

```
~Prize~, _n._ ~1.~ Reward, premium, trophy.

~2.~ Capture.
```

  - ‚;’ se folosește pentru identificarea sinonimului „annoyance” al cuvântului „Plague”:

```
~Plague~, _n._ ~1.~ Pestilence, pest.

~2.~ Affliction; annoyance, vexation, trouble, nuisance,
curse,
torment, thorn in one's side.
```

- Ultimul caracter al cuvântului curent trebuie să se găsească în următoarele: „ , ”, „.”, „;”, „(”, **sau** primul caracter al cuvântului următor face parte din caracterele enumerate. Verificarea se face folosind metoda isRightDelimiter(), care se comportă în mod similar cu isLeftDelimiter(), dar comparația se face cu caracterele menționate mai sus. Deoarece se verifică următorul cuvânt pentru condiția alternativă, se va adăuga la sfârșitul string-ului generat de getDictionarySection() încă un cuvânt: „!padding!”, care nu va fi analizat la parcurgerea vectorului. Exemple de cazuri:
  - Din nou, pentru sinonimul „comely” al cuvântului „Pretty” se găsește virgula la sfârșitul acestuia: „**comely,** fine”;
  - Punctul este găsit în dreapta ultimului cuvânt al textului dat ca exemplu: „trim, **fair.**” ;
  - ,;’ se află în dreapta sinonimului „abominable” al cuvântului „Loathsome”:

```
~Loathsome~, _a._ ~1.~ Disgusting, sickening, nauseous,
nauseating,
repulsive, offensive, revolting, palling.

~2.~ Hateful, detestable, odious, shocking, abominable;
abhorrent,
execrable.
```

  - Paranteza rotundă este verificată în condiția alternativă. Cuvântul „Beautiful” din textul dat ca exemplu nu ar fi fost identificat drept sinonim dacă nu s-ar fi găsit o paranteză pătrată la începutul următorului cuvânt: „_a._ **Beautiful (**_without”.

După ce este determinat un cuvânt ca fiind un sinonim al cuvântului dat ca parametru, înainte de a fi adăugat în lista de sinonime, acesta este filtrat prin metoda filterWord(), care primește un cuvânt ca argument, elimină toate caracterele non-litere din cuvântul respectiv

(excepție făcând cratima), apoi transformă toate literele mari în litere mici (de exemplu, „HANDSOME,” devine „handsome”) și returnează String-ul obținut.

Această metodă getSynonyms() nu este folosită de către clasa TextAnalyser sau direct de către fereastra principală, ci de către o altă metodă getSynonyms() supraîncărcată.

```
public ArrayList<String> getSynonyms(String word, int numberOfLevels) {
        ArrayList<String> synonyms = getSynonyms(word);
        while (numberOfLevels-- > 1) {
            ArrayList<String> newSynonymsList = new ArrayList<>();
            for (String synonym : synonyms) {
                newSynonymsList.addAll(getSynonyms(synonym));
            }
            for (String synonym : newSynonymsList) {
                if (!synonyms.contains(synonym)) {
                    synonyms.add(synonym);
                }
            }
        }
        return synonyms;
    }
```

Metoda supraîncărcată este folosită de către fereastra principală pentru a afișa sinonimele unui cuvânt selectat, și de către TextAnalyser pentru a învăța automat un cuvânt. Metoda supraîncărcată primește, pe lângă cuvântul ce trebuie găsit în dicționarul de sinonime, un număr, care indică gradul de căutare, prin argumentul `numberOfLevels`. Dacă această variabilă este mai mică sau egal cu valoarea 1, metoda returnează lista cu sinonime a cuvântului parametru. Dacă valoarea este 2, se vor construi liste pentru fiecare cuvânt din lista de sinonime, apoi se va returna reuniunea acestor liste. Pentru valoarea 3, se va lua lista care ar fi fost returnată la valoarea 2 a argumentului și s-ar fi generat liste pentru toate cuvintele din acea listă de sinonime, acestea fiind reunite cu cea care va fi returnată la fiecare generare a uneia dintre ele. Algoritmul funcționează în mod similar pentru orice valoare întreagă de tip `int`.

Există două metode de învățare automată, ambele accesibile din clasa TextAnalyser. Metoda implicită de învățare a unui cuvânt se numește learnWord(), care primește ca parametri cuvântul dorit, reținut în variabila `word`, și gradul de căutare al sinonimelor folosite pentru

învățarea cuvântului respectiv, reținut în variabila `numberOfLevels`.

Codul pentru această metodă arată astfel:

```
public String learnWord(String word, int numberOfLevels) {
    String reactionGeneratedByWord = "confused";
    ArrayList<String> synonyms = getSynonyms(word, numberOfLevels);
        boolean synonymIsAKnownWord = false;
        for (int i = 0; i < synonyms.size() && !synonymIsAKnownWord; i++) {
          if (isHappyWord(synonyms.get(i))) {
            synonymIsAKnownWord = true;
            saveNewWord(word, "happy");
            reactionGeneratedByWord = "happy";
          } else if (isSadWord(synonyms.get(i))) {
            synonymIsAKnownWord = true;
            saveNewWord(word, "sad");
            reactionGeneratedByWord = "sad";
          } else if (isAngryWord(synonyms.get(i))) {
            synonymIsAKnownWord = true;
            saveNewWord(word, "angry");
            reactionGeneratedByWord = "angry";
          }
        }
    return reactionGeneratedByWord;
  }
```

Logica din spatele codului este simplă: se generează lista de sinonime a cuvântului necunoscut (și a sinonimelor acelor sinonima, ș.a., în funcție de valoarea din `numberOfLevels`), apoi programul verifică dacă cunoaște unul din cuvintele din lista de sinonime. Dacă unul din sinonime este recunoscut ca un cuvânt ce generează o anumită emoție, atunci se deduce că și cuvântul necunoscut va genera emoția respectivă, și este memorat în aceeași categorie cu sinonimul recunoscut folosind metoda saveNewWord(). Se returnează String-ul „happy”, în cazul în care cuvântul învățat a fost adăugat fișierului și listei de cuvinte pentru cuvintele generatoare de reacții fericite; se returnează „sad”, dacă cuvântul va genera pe viitor reacții triste,

„angry” pentru reacții nervoase, și „confused” dacă cuvântul nu a putut fi învățat.

A doua metodă de învățare, `revearseLearnWord()`, se apelează atunci când utilizatorul a bifat opțiunea „Reverse Search” din fereastra principală. Cum metoda normală genera sinonimele cuvântului necunoscut și verifica dacă măcar unul dintre acestea făcea parte din una din listele de cuvinte cunoscute, această metodă lucrează cumva invers – generează liste de sinonime pentru fiecare cuvânt cunoscut (evident, mărimile listelor sinonimelor generate vor fi mai mari, în funcție de valoare reținută de argumentul `numberOfLevels`), și verifică dacă acesta se găsește într-una dintre listele generate. Dacă se găsește cuvântul, acesta este memorat în aceeași categorie a cuvântului cunoscut folosit pentru a genera lista de sinonime în care cuvântul necunoscut a fost găsit. Logica din spatele acestui algoritm este: „Dacă un cuvânt la care nu se știe cărei emoții ar trebui asociat are același înțeles cu un alt cuvânt cunoscut, atunci acesta va genera aceeași emoție pe care o generează și cuvântul cunoscut”.

Codul pentru metoda alternativă de învățare a unui cuvânt necunoscut este următorul:

```
public String reverseLearnWord(String word, int numberOfLevels) {
        ArrayList<String> happySynonyms = new ArrayList<>();
        ArrayList<String> sadSynonyms = new ArrayList<>();
        ArrayList<String> angrySynonyms = new ArrayList<>();

        for(String knownWord : happyWords) {
            ArrayList<String>     synonyms     =     getSynonyms(knownWord,
numberOfLevels);
            for(String synonym : synonyms) {
                if(!happySynonyms.contains(synonym)) {
                    happySynonyms.add(synonym);
                }
            }
        }
        if(happySynonyms.contains(word)) {
            saveNewWord(word, "happy");
            return "happy";
        }

        for(String knownWord : sadWords) {
            ArrayList<String>     synonyms     =     getSynonyms(knownWord,
numberOfLevels);
            for(String synonym : synonyms) {
                if(!sadSynonyms.contains(synonym)) {
```

```
                    sadSynonyms.add(synonym);
                }
            }
        }
        if(sadSynonyms.contains(word)) {
            saveNewWord(word, "sad");
            return "sad";
        }

        for(String knownWord : angryWords) {
            ArrayList<String>     synonyms     =     getSynonyms(knownWord,
numberOfLevels);
            for(String synonym : synonyms) {
                if(!angrySynonyms.contains(synonym)) {
                    angrySynonyms.add(synonym);
                }
            }
        }
        if(angrySynonyms.contains(word)) {
            saveNewWord(word, "angry");
            return "angry";
        }
        return "confused";
    }
```

Prima pereche de `if` și `for` populează lista `happySynonyms` cu toate sinonimele găsite pentru toate cuvintele asociate emoției de fericire, apoi verifică dacă cuvântul necunoscut se găsește în acea listă. Dacă se găsește într-una dintre liste, metoda va returna String-ul „happy”, semnalând apelantului că cuvântul primit ca parametru a fost memorat ca cuvânt generator de reacții fericite. La fel se procedează și în a doua și a treia pereche `for – if`, cu diferența că a doua metodă se ocupă de cuvintele asociate tristeții, iar a treia de cuvintele asociate mâniei.

Concluziile lucrării

Concluzii

Lucrarea de față are ca scop prezentarea unei aplicații care poate reacționa cât mai credibil și corect în fața unui text introdus de către un utilizator. Am încercat să scriu programul într-un mod cât mai eficient, în așa fel încât acesta să consume cât mai puțină memorie RAM și să nu folosească în mod intensiv procesorul. Totodată, am construit acestă aplicație respectând modelul MVC, separând View-ul (fereastra avatarului, unde se afișează în mod grafic diferitele stări ale AI-ului, și fereastra principală, în care se găsește chat-ul și toate celelalte butoane), de Model și Controller (clasele care au rolul de a analiza textul, de a accesa dicționarul de sinonime local, și de a memora și învăța automat noi cuvinte).

Contribuții personale

Datorită eficienței și modularizării codului, acesta poate fi folosit în foarte multe aplicații cu ușurință, fără a fi necesare modificări ale codului, singura excepție fiind doar cazul în care un dezvoltator dorește să caute sinonime și pentru alte tipuri de cuvinte în afară de substantive și adjective, caz în care trebuie modificate doar două linii din clasa SynonymFinder. Tipurile de aplicații care ar putea găsi folositoare clasele care constituie partea de Model și Controller a aplicației (clasele TextAnalyser și SynonimFinder) sunt numeroare, variind de la aplicații de tip messenger (se pot implementa, de exemplu, avatare care se schimbă pe baza textului utilizatorului, sau se poate determina atmosfera conversației și se poate informa utilizatorul dacă textul pe care vrea să îl trimită va îmbunătăți sau înrăutăți atmosfera) sau editoare de text (unde se pot înlocui cuvintele repetate prea des cu diverse alte sinonime, pentru a îmbogăți estetica textului), până la aplicații care se folosesc intensiv de conceptul de Inteligență Artificială, ca jocurile video care conțin convorbiri cu NPCs (Non-Playable Characters, personaje care sunt controlate de către aplicație) și chatterbots (pentru a răspunde cât mai credibil utilizatorului, ținând cont și de emoțiile exprimate de acesta).

Direcţii de dezvoltare

Pentru viitor, aș dori să adaug alte funcționalități programului, printre care includerea mai multor dicționare de sinonime pentru a îmbunătăți acuratețea găsirii sinonimelor și a învățării cuvintelor, posibilitatea de a utiliza și dicționare online, și recunoașterea unor tipuri de emoții mai complexe, ca mândria sau timiditatea.

O altă îmbunătățire a aplicației ar fi animarea modelului personajului, și înlocuirea imaginilor statice ale avatarului cu animații în format video sau .gif.

Codul ar putea fi „tradus” în limbajul de programare C#, pentru a putea fi folosit, de exemplu, în motoare grafice pentru jocuri video ca XNA, Unity, Paradox3D, Wave, etc. Codul ar mai putea fi, de asemenea, scris în limbajul de programare Python, pentru a putea fi folosit cu ușurință de către motoare grafice pentru jocuri video (Panda3D, Cocos2D, pyGame), în Autodesk Maya, sau în aplicații web.

Bibliografie

[1] Watkins A. 2012. Getting Started in 3D with Maya

[2] Lanier L. 2007. Maya Professional Tips and Techniques

[3] Cabrera C. 2008. An Essential Introduction to Maya Character Rigging

[4] Alpaydin E. 2004. Introduction To Machine Learning, MIT Press

[5] Mitchell T.M. 1997. Machine Learning, McGraw-Hill

[6] Davis R. & Lenat D.B. 1982. Knowledge-Based Systems in Artificial Intelligence. New York: McGraw-Hill International Book Company

[7] http://www.gutenberg.org/ebooks/38390 - Dicționar de sinonime în limba engleză, scris de Richard Soule, valabil gratuit în formate EPUB, Kindle, HTML și plain text (UTF-8)

[8] http://www.cs.berkeley.edu/~russell/intro.html

[9] http://www-formal.stanford.edu/jmc/whatisai/whatisai.html

[10] http://www.coli.uni-saarland.de/~hansu/what_is_cl.html

[11] http://loebner.net/Prizef/TuringArticle.html

[12] http://www.psych.utoronto.ca/users/reingold/courses/ai/turing.html

[13] https://en.wikipedia.org/wiki/Chatterbot

[14] http://www.stanford.edu/group/SHR/4-2/text/dialogues.html

Anexă

Secvențe cod analiză text (Capitolul III.2.)

```
private String generateReaction(boolean reactionFound) {
        // decide the final reaction based on most often generated reactions
        String output;
        output = getReaction("confused");
        lastReaction = "confused";
        if (reactionFound) {
            if (detectPerson() == 2) {
                if (reactionCount[0] > reactionCount[1]
                        && reactionCount[0] > reactionCount[2]) {
                    output = getReaction("happy");
                    lastReaction = "happy";
                }
                if (reactionCount[1] > reactionCount[0]
                        && reactionCount[1] > reactionCount[2]) {
                    output = getReaction("sad");
                    lastReaction = "sad";
                }
                if (reactionCount[2] > reactionCount[0]
                        && reactionCount[2] > reactionCount[1]) {
                    output = getReaction("angry");
                    lastReaction = "angry";
                }
                if (reactionCount[0] == reactionCount[1]
                        && reactionCount[0] == reactionCount[2]) {
                    if (globalReactionCount[0] >= globalReactionCount[1]) {
```

```
                        if (globalReactionCount[0] >= globalReactionCount[2])
{
                            output = getReaction("happy");
                            lastReaction = "happy";
                        }
                        if (globalReactionCount[0] < globalReactionCount[2])
{
                            output = getReaction("angry");
                            lastReaction = "angry";
                        }
                    } else {
                        if (globalReactionCount[1] >= globalReactionCount[2])
{
                            output = getReaction("sad");
                            lastReaction = "sad";
                        }
                        if (globalReactionCount[1] < globalReactionCount[2])
{
                            output = getReaction("angry");
                            lastReaction = "angry";
                        }
                    }
                }
                if (reactionCount[0] == reactionCount[1]
                        && reactionCount[0] != 0
                        && reactionCount[0] != reactionCount[2]) {
                    if (globalReactionCount[0] >= globalReactionCount[1]) {
                        output = getReaction("happy");
                        lastReaction = "happy";
                    } else {
                        output = getReaction("sad");
                        lastReaction = "sad";
```

```
                    }
                }
                if (reactionCount[0] == reactionCount[2]
                        && reactionCount[0] != 0
                        && reactionCount[0] != reactionCount[1]) {
                    if (globalReactionCount[0] >= globalReactionCount[2]) {
                        output = getReaction("happy");
                        lastReaction = "happy";
                    } else {
                        output = getReaction("angry");
                        lastReaction = "angry";
                    }
                }
                if (reactionCount[1] == reactionCount[2]
                        && reactionCount[1] != 0
                        && reactionCount[1] != reactionCount[0]) {
                    if (globalReactionCount[1] >= globalReactionCount[2]) {
                        output = getReaction("sad");
                        lastReaction = "sad";
                    } else {
                        output = getReaction("angry");
                        lastReaction = "angry";
                    }
                }
                // If the user is talking about himself/herself
            } else if (detectPerson() == 1) {
                // If the user is not talking about the AI, the global
reaction
                // score is not affected
                for (int i = 0; i < reactionCount.length; i++) {
                    globalReactionCount[i] -= reactionCount[i];
```

```
                }
                if (reactionCount[0] >= reactionCount[1] + reactionCount[2])
{
                    if (globalReactionCount[0] >= globalReactionCount[1]
                            + globalReactionCount[2]) {
                        output = getReaction("firstPersonPositivePolite");
                        lastReaction = "happy";
                    } else {
                        output = getReaction("firstPersonPositiveRude");
                        lastReaction = "angry";
                    }
                } else {
                    if (globalReactionCount[0] >= globalReactionCount[1]
                            + globalReactionCount[2]) {
                        output = getReaction("firstPersonNegativePolite");
                        lastReaction = "happy";
                    } else {
                        output = getReaction("firstPersonNegativeRude");
                        lastReaction = "angry";
                    }
                }
                // If the user is talking about someone other than
himself/herself
                // and the AI
            } else if (detectPerson() == 3) {
                // If the user is not talking about the AI, the global
reaction
                // score is not affected
                for (int i = 0; i < reactionCount.length; i++) {
                    globalReactionCount[i] -= reactionCount[i];
                }
                output = getReaction("thirdPerson");
```

```
            lastReaction = "normal";
        }
    } else {
        if (inputHasGreeting()) {
            output = getReaction("hello");
            lastReaction = "happy";
        }
    }
    return output;
}
public String analyseText(String originalInput) {
    processedWords = new ArrayList<>();
    boolean reactionFound = false;
    String input = getFilteredInput(originalInput);
    String word, auxWord;
    int numberOfWords = getNumberOfWords(input);
    // position 0 is for happy, 1 is for sad, and 2 is for angry
    reactionCount = new int[3];
    for (int i = 0; i < numberOfWords; i++) {
        word = getNextWord(input);
        processedWords.add(word);
        input = input.substring(word.length());
        // formatting the text
        if (i < numberOfWords - 1) {
            auxWord = getNextWord(input);
            input = input.substring(input.indexOf(auxWord));
        }

        if (isHappyWord(word)) {
            reactionFound = true;
            reactionCount[0]++;
```

```
            globalReactionCount[0]++;
            if (isSentenceNegated()) {
                reactionCount[0]--;
                reactionCount[1]++;
                globalReactionCount[0]--;
                globalReactionCount[1]++;
            }
        } else if (isSadWord(word)) {
            reactionFound = true;
            reactionCount[1]++;
            globalReactionCount[1]++;
            if (isSentenceNegated()) {
                reactionCount[1]--;
                reactionCount[0]++;
                globalReactionCount[1]--;
                globalReactionCount[0]++;
            }
        } else if (isAngryWord(word)) {
            reactionFound = true;
            reactionCount[2]++;
            globalReactionCount[2]++;
            if (isSentenceNegated()) {
                reactionCount[2]--;
                reactionCount[0]++;
                globalReactionCount[2]--;
                globalReactionCount[0]++;
            }
        }
    }
    return generateReaction(reactionFound);
}
```