



ANIMARE

**A Graduation Project Book submitted to the Faculty of
Engineering, Cairo University in Partial Fulfillment of the B.Sc.
Degree in Computer Engineering**

Submitted by

**Bahi Ali
Gehad Mohsen**

**Berlanty Kerlos
Yasmine Alaa**

**Supervised by
Prof. Nevin Darwish**

July 10,2019

Acknowledgement

We are extremely grateful for those who have helped us, gave us support, ideas, inspiration, comments and guidance.

Firstly, we would like to express our sincere gratitude to our supervisor **Prof. Nevin Darwish** for her constant guidance, support and encouragement, immense knowledge in both technical and personal skills, and also for providing necessary information and direction. It was our honor to work under her supervision.

Secondly, we also want to thank Eng. Yehia Zakriaa for his help and precious comments.

Thirdly, we want to thank our families for their patience, support, encouragement, to whom we are pleased to present them this accomplishment.

Finally, we want to thank everybody helped in a way or another in letting this project come to life.

Bahi, Berlanty, Gehad, Yasmine

Abstract

Animated videos have become a topic of interest as they proved their great ability to simplify ideas as they clearly deliver your message, offer great fun, provide total creative freedom, ideal for all marketing channels and help visual learners to understand easily. **Animare** brings your ideas to live, it takes a text description and creates a 3D animated video for the described scene. It then, chooses suitable 3D models and applies mentioned actions to the descriptions and renders them to a video output.

Our product targets visual learners, educational organizations, video content creators and advertisement makers. **Animare** is a Desktop application that visualizes a written description. The system specifically analyzes the text using natural language processing techniques, chooses appropriate models from a standard pool, generates a static visual scene and finally apply the necessary motion using planning techniques and physics to get it ready for the user to display.

Contacts

Supervisor	Email
Prof. Nevin Darwish	ndarwish@eng.cu.edu.eg

Team members	Phone numbers	Email
Bahi Ali	01150797376	bahi.ali26@hotmail.com
Berlanty Kerlos	01225586443	berlnty456@gmail.com
Gehad Mohsen	01553512341	gehadmohsen9519@gmail.com
Yasmine Alaa	01143893764	yasminelgourisy@hotmail.com

Table of Contents

Acknowledgement.....	2
Abstract.....	3
Contacts.....	4
Table of Contents.....	5
List of tables.....	8
List of figures.....	9
1. Chapter 1: Introduction.....	11
1.1. Problem Definition.....	12
1.2. Project Idea.....	12
1.3. Motivation and Justification	12
1.3.1. Motivation.....	12
1.3.2. Educational Value	13
1.3.2.1. Technical Level.....	13
1.3.2.2. Non-Technical Level.....	13
1.4. Domains of application.....	13
1.5. Summary of Approach.....	13
1.5.1. Script Analysis	13
1.5.2. Model Generation	14
1.5.3. Static Visualization	14
1.5.4. Dynamic Visualization.....	14
1.6. Document Overview	14
2. Chapter 2 Market Survey & Feasibility Study	15
2.1. Intended Customers.....	16
2.2. Current Market.....	16
2.2.1. Wordseye.....	16
2.2.2. Text2Scene.....	16
2.2.3. Carsim.....	16
2.2.4. Antonia Antonova.....	16
2.2.5. Sceneseer	17
2.3. Market Survey Statics.....	17
2.4. Feasibility Study	19
2.4.1. Description of Products & Services.....	19
2.4.2. Technological Feasibility.....	19
2.4.2.1. Software Requirements.....	19
2.4.2.2. Hardware Requirements.....	19
2.4.3. Economical Feasibility	20
2.4.3.1. Cost.....	20

2.4.3.2.	Benefit.....	20
2.4.4.	Schedule Feasibility.....	20
2.4.5.	Availability.....	20
2.4.5.1.	Market Strategy.....	20
2.4.6.	Legal Feasibility.....	21
3.	Chapter 3 Necessary Background.....	22
3.1.	Necessary Background.....	23
3.1.1.	Natural Language Processing.....	23
3.1.1.1.	Dependency Parsing.....	23
3.1.1.2.	Coreference Resolution.....	25
3.1.1.3.	Tokenization.....	26
3.1.1.4.	Text Normalization.....	27
3.1.1.5.	Part of Speech tagging.....	28
3.1.2.	Planning.....	29
3.1.3.	Axis Aligned Bounding Box.....	29
3.1.4.	Impulse Resolution.....	30
3.1.4.1.	A basic understanding of simple vector math.....	30
3.1.4.2.	The ability to perform algebraic math on vector.....	31
3.1.4.3.	A basic understanding of simple physics.....	31
3.1.5.	Separating Axis Theorem.....	32
3.1.5.1.	Algorithm.....	32
3.1.5.2.	Intersection.....	32
3.1.6.	A* Search Algorithm.....	33
3.2.	Literature Review.....	34
3.2.1	Static scene positioning.....	34
3.2.2	Semantic Parsing for text to 3D Scene Generation.....	34
4.	Chapter 4 Design Specifications.....	35
4.1.	Assumption and/or Requirements.....	36
4.1.1.	Model Assumptions.....	36
4.1.2.	Action Assumptions.....	36
4.1.3.	Written text Assumptions.....	36
4.1.4.	Spatial relation Assumption.....	36
4.2.	Block Diagram.....	37
4.2.1.	Block Description.....	37
4.2.1.1.	GUI.....	38
4.2.1.2.	Script Analysis Module.....	39
4.2.1.3.	Model Generation.....	42
4.2.1.4.	Static Visualization.....	42
4.2.1.5.	Dynamic Visualization.....	43

5. Chapter 5	
Tools.....	45
5.1. Overview.....	46
5.2. System Tools and Libraries.....	46
5.2.1. Pycharm.....	46
5.2.2. Spacy.....	46
5.2.3. NLTK.....	4
6	
5.2.4. Colour 0.1.5.....	46
5.2.5. GitHub.....	47
5.2.6. Blender.....	47
5.2.7. Bpy.....	47
5.2.8. Binvov.....	47
5.2.9. Kivy.....	47
5.3. Physics Engine.....	48
5.3.1. Collision Detection.....	48
5.3.2. Impulse Resolution.....	48
5.4. Programming Languages.....	50
5.4.1. Python 2.7, 3.6.....	50
6. Chapter 6 Testing and Results	51
6.1. Test Scope	52
6.2. Testing Methodology	52
6.3. Test Cases and Results	53
6.3.1. Script Analysis Unit Testing	54
6.3.2. Model Generation Unit testing	60
6.3.3. Static visualization Unit testing	60
6.3.4. Dynamic visualization Unit testing	62
7. Chapter 7 Conclusion and Future Work.....	65
7.1. Conclusion.....	66
7.2. Future Work.....	66
References	67
Arabic summary	68

List of tables

Table 1 : test case 1	56
Table 2 : test case 2	57
Table 3 : test case 3	59
Table 4 : test case 4	60
Table 5 : test case 5	61
Table 6 : test case 6	62
Table 7 : test case 7	63
Table 8 : test case 8	63
Table 9: test case 9	64

List of figures

Fig 2.1: groups of people interested in reading and watching videos	15
Fig 2.2 : people interested/not interested in using Animare	16
Fig 2.3: fields of usage of Animare	16
Fig 2.4: people who are interested/not interested to use paid service	17
Fig 2.5: Schedule Feasibility	18
Fig.3.1 : Dependency parsing illustration	21
Fig 3.2 : Dependency tree illustration	24
Fig 3.3 : dependency tree characteristics	22
Fig 3.4 : illustration of the differences between projectivity and non-Projectivity property of dependency tree	22
Fig 3.5 : projective dependency tree example	23
Fig 3.6 : Non-projective dependency tree example	23
Fig 3.7 : Coreference resolution example	23
Fig 3.8 : example 1 on word normalization	25
Fig 3.9 : example 2 on word normalization	26
Fig 3.10 : explaining main part of speech	27
Fig 3.11: Axis Aligned Bounding Box	27
Fig 4.1 : block diagram of our system	35
Fig 4.2 : Boy model before and after scale deformation	39
Fig 4.3 : illustration action tree for actions shooting and walking	40
Fig 5.1 : Axis Aligned Bounding Box (AABB)	47

Chapter 1

Introduction



1.1 Problem Definition

Some people have problems with reading as Visual-Spatial Learners they have their own way of learning in which information is associated with images/figures. This learning style requires that learners first see what they are expected to know. Also kids who have difficulties in reading.

So Short animated videos are very useful it helps in:

- Making a visual explanation for an idea, like explaining for kids or explaining difficult idea for a person who is a beginner in some field.
- Making short animated videos as an advertisement.
- Imagining a scenario (for directors).
- Providing total creative freedom for expressing ideas.
- Creating entertaining content .

1.2 Project Idea

Animare is a latin verb that means “*give life to*” ,and that is what our project does, it gives life to words. The idea of our project is Visualization of a written script. The input will be a description of a scene in text format and the output will be the visualization of the scene in 3D animated video. The system specifically analyzes the text, extracts the models and their actions in the story then chooses appropriate models from a standard pool, generates a static visual and finally applies the necessary animation on the models

1.3 Motivation and Justification

This section is about our motivations for making this project.

1.3.1 Motivation

Our main motivations are:

- The great contributions that are being made in evolving the field of NLP and Computer Graphics nowadays.
- Animated videos have become a topic of interest as they proved their great ability to simplify ideas as they clearly deliver your message, offer great fun, provide total creative freedom, ideal for all marketing channels and help visual learners to understand easily.
- We found out from the results of our survey that 81.2% of people understand easily with watching videos more than reading, and they especially prefer animated videos.

1.3.2 Educational Value

Working on Animare has added new values not only to our technical experience, but also to our non-technical experiences that will help us in the future.

1.3.2.1 Technical Level

- How to use Blender python library (bpy) to deal with 3D models, generate animations, rendering 3D scenes .
- How to deal with the physics of models.
- How to determine reasonable positions for objects in the scene.
- Natural Language Processing techniques
- How to make good GUI with python

1.3.2.2 Non-Technical Level

- Self-Learning.
- Time planning.
- Project management.
- Marketing.
- Team working.

1.4 Domains of Application

Our targeted Market includes:

- Parents: They can use Animare to visualize a story to their kids, because images and animated videos grab their attention and make them understand well.
- Educational organizations : They can use it to simplify their idea to students
- Directors : Can use it in imagining a scenario
- Advertisements makers : marketeers that uses videos to target their customers.
- People who prefer watching than reading can use it for entertainment and watch the story rather than reading it

1.5 Summary of Approach

Animare is divided into four modules, **Script Analysis**, **Model Generation**, **Static Visualization**, and **Dynamic Visualization**.

1.5.1 Script Analysis

We start by analyzing the script and extract information from it, These information are: Models in the story and their actions, Relations between models to be used in positioning the models in the scene and The sequence of the actions

1.5.2 Model Generation

This module selects the best Model from the Models Set based in model's description that came from Script Analysis module and apply any needed transformation either scale deformation or object coloring

1.5.3 Static Visualization

The output of this module, is the location of every Model in the scene, It finds suitable initial positions for objects in the scene according to the spatial relations between them that came from Script Analysis module.

1.5.4 Dynamic Visualization

This is the last module , It is responsible for planning, applying the actions to the models and taking care for the order of actions then rendering and produces the video, The input for this module is the output of the previous 3 modules

The user interact with the system through a GUI interface that takes the text description and display the output video .

1.6 Document Overview

In this document, we are trying as much as possible to illustrate each and

every step we have been through since we started working on **Animare**. Here is the outline of this document:

Chapter 2:2 Presenting the market survey and feasibility study

Chapter 3: Explaining any necessary background needed throughout the whole project.

Chapter 4: Stating the system design, features provided and system specification.

Chapter 5: The tools and environment used in our implementation.

Chapter 6: Our test plan and methodology to ensure the accuracy of our product.

Chapter 7: Summarizing our findings and what we will do as our future work.

Finally, references and appendices where you will find the feasibility study.

Chapter 2

Market Survey



2.1. Intended Customers

Anyone wants to create an animated video to express his ideas could use our product.

But, Animare mainly targets:

- Educational organizations : They can use it to simplify their idea to students
- Directors : Can use it in imagining a scenario
- Advertisements makers : marketeers that uses videos to target their customers.
- Video content creators: people who provide videos with specific content on any social media (Youtube, Facebook, ...etc)

2.2. Current Market

These are our competitors in the Market.

2.2.1. Wordseye

- **Description:** A language-based 3D scene generation systems to let ordinary users quickly create 3D scenes without having to learn special software, acquire artistic skills, or even touch a desktop window-oriented interface. WordsEye is a system for automatically converting text into representative 3D scenes. WordsEye relies on a large database of 3D models and poses to depict entities and actions.
- **Output:** 3D static scenes.

2.2.2 Text2scene

- **Description:** A model that learns to interpret natural language describing a scene to generate an abstract pictorial representation. The pictorial representations generated by our model comprise the spatial distribution and attributes of the objects in the described scene.
- **Output:** 2D images.

2.2.3 Carsim

- **Description:** CarSim is an automatic text-to-scene conversion system. It analyzes written descriptions of car accidents and synthesizes 3D scenes of them. The conversion process consists of two stages. An information extraction module creates a tabular description of the accident and a visual simulator generates and animates the scene
- **Output:** 3D animated scenes (*only applied for Road Accident Reports*) .

2.2.4 Antonia Antonova

- **Description:** This project aims to build a deep learning pipeline that takes text descriptions and generates unique video depictions of the content described.

The crux of the project lies with the Generative Adversarial Network, a deep learning algorithm that pins two neural networks against each other in order to produce media that is unique and realistic.

- **Output:** 2D videos.

2.2.5 Sceneseer

- **Description:** An interactive text to 3D scene generation system that allows a user to design 3D scenes using natural language. A user provides input text from which we extract explicit constraints on the objects that should appear in the scene. Given these explicit constraints, the system then uses a spatial knowledge base learned from an existing database of 3D scenes and 3D object models to infer an arrangement of the objects forming a natural scene matching the input description
- **Output:** 3D scenes *(only applied for indoor scenes)*.

2.3 Market Survey Statistics

The questions of our market survey.

2.3.1 What makes you realize better reading or watching animated videos ?

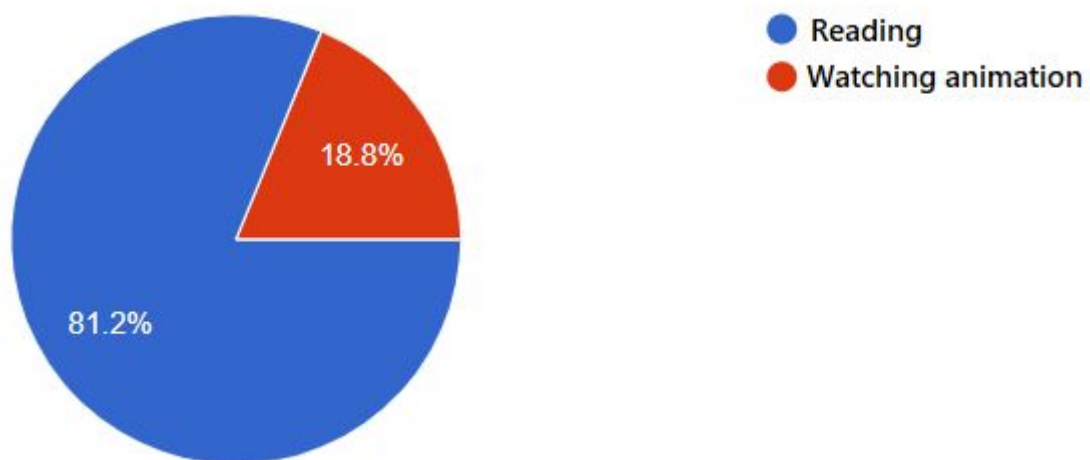


Fig 2.1 groups of people interested in reading and watching videos

2.3.2 if you have an application that generates animated videos from your description, would you use it?

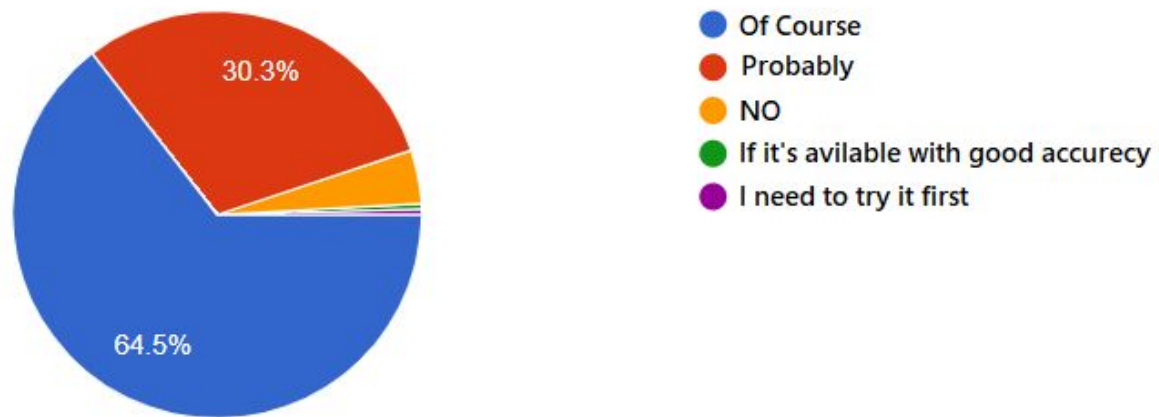


Fig 2.2 people interested/not interested in using Animare

2.3.3 In what would you use it for?

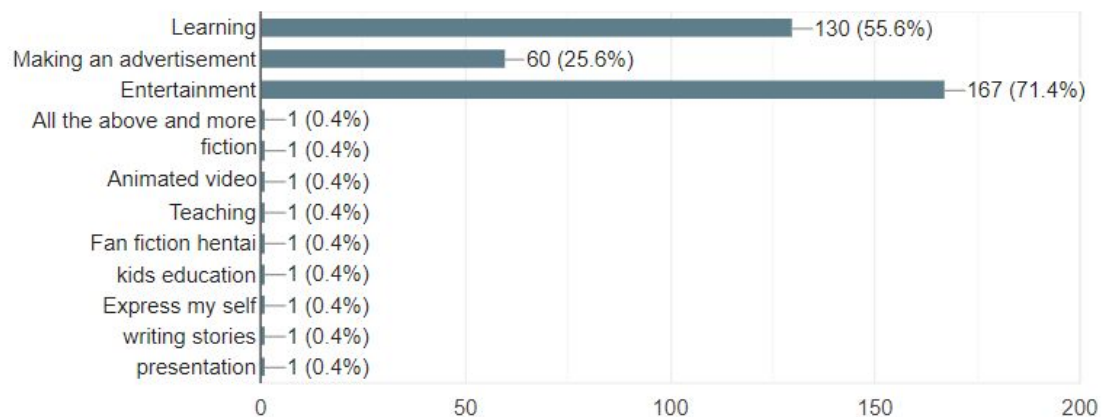


Fig 2.3 fields of usage of Animare

2.3.4 If it is a paid service, would you use it?

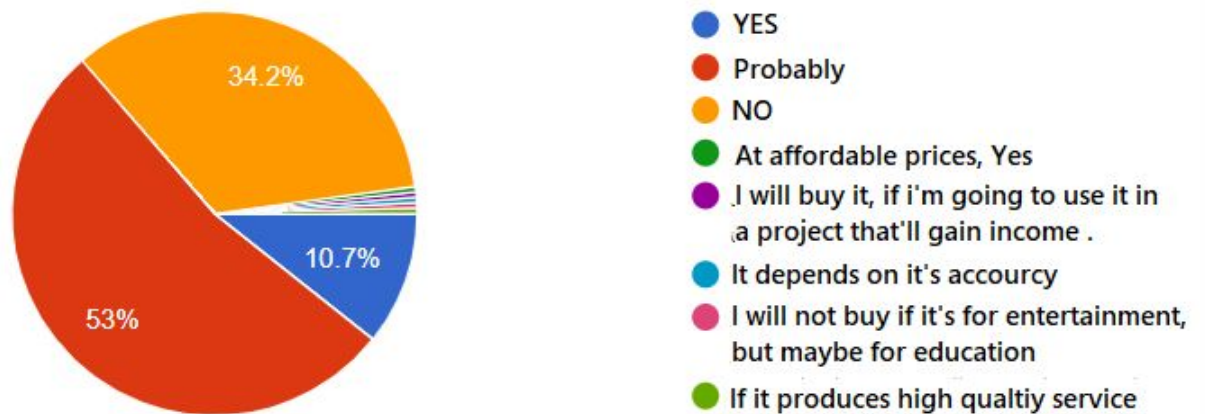


Fig 2.4 people who are interested/not interested to use paid service

2.4. Feasibility Study

2.4.1. Description of Products & Services

Desktop application used for visualization, that generates 3D animated videos from text description.

Input: text description.

Output: 3D animated video.

2.4.2. Technological Feasibility

The software and hardware specifications to run the project.

2.4.2.1. *Software Requirements*

- Operating system : Windows, macOS, Ubuntu
- SpaCy library
- Kivy library
- bpy library
- NLTK library
- Colour 0.1.5

2.4.2.2. *Hardware Requirements*

- Memory : 3 GB
- RAM : 8 GB

2.4.3. Economical Feasibility

2.4.3.1. Cost

We used in the development of our project tools and libraries that are open source and free licenced. So there is no development cost for our project.

2.4.3.2. Benefit

- Help visual learners in the learning process.
- It can help in making Ads and animated movies.
- Offers an easy method for expressing / explaining ideas.
- Offers a method of entertainment.
- Help animated movie creators and game developers to visualize their scenario/ ideas before implementation.
- Imagining a scenario (for directors).
- Cost Effective.

2.4.4. Schedule Feasibility

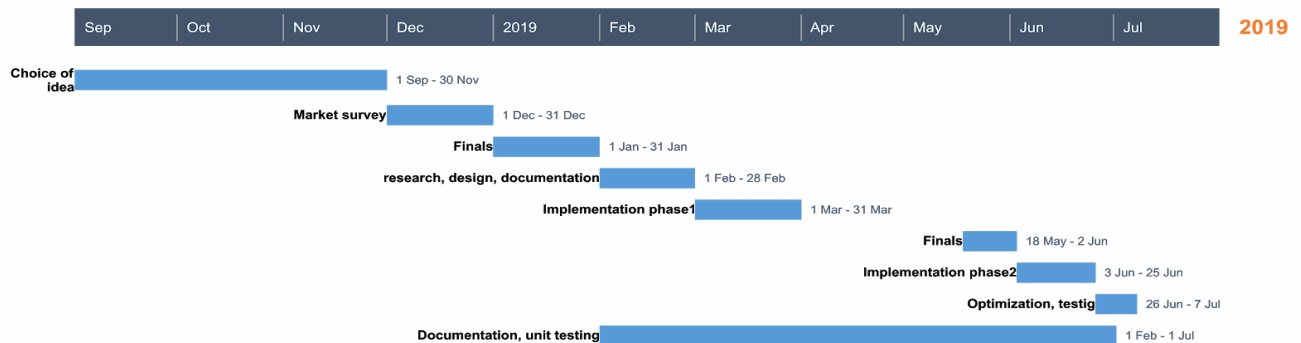


Fig 2.5 Schedule Feasibility

2.4.5. Availability

Animare will be available for everyone to use for free but later on after supporting a wider category of animations and models, some additional features will involve some charges to continue this project to be running. Also if it will be used in commercial user should have a licence that will cost some charges.

2.4.5.1. Marketing Strategy

- For non commercial users **Animare** will be offered for free but some additional features that users should pay for them as:
 - Exporting a video without watermark or sign of Animare.
 - Unlimited number of videos generated per day.
- For video content creators users could have full access for a limited time then they should pay for it, or they lose access to some features.
- For educational organisations they would be able to have full access to all the features for one semester (except for the watermark or program sign feature), then they should pay for the services or they won't have any access to all features.

In addition to that we could offer a workshop to show users how to use Animare efficiently to express their ideas clearly.
- After all **Animare** should have good tutorial and good user help to encourage customers to use it.

2.4.6. Legal Feasibility

Our project doesn't conflict with any intellectual property. Our project uses open source tools and implements concepts from papers published in international scientific conferences or journals.

Chapter 3

Necessary Background



3.1 Necessary Background

This section presents the science behind the project

3.1.1 Natural language processing

Is a subfield of computer science , information engineering , and artificial intelligence concerned with the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data.

3.1.1.1 Dependency parsing

3.1.1.1.1 What is Dependency Parsing?

Dependency parsing is the task of extracting a dependency parse of a sentence that represents its grammatical structure and defines the relationships between “head” words and words, which modify those heads.

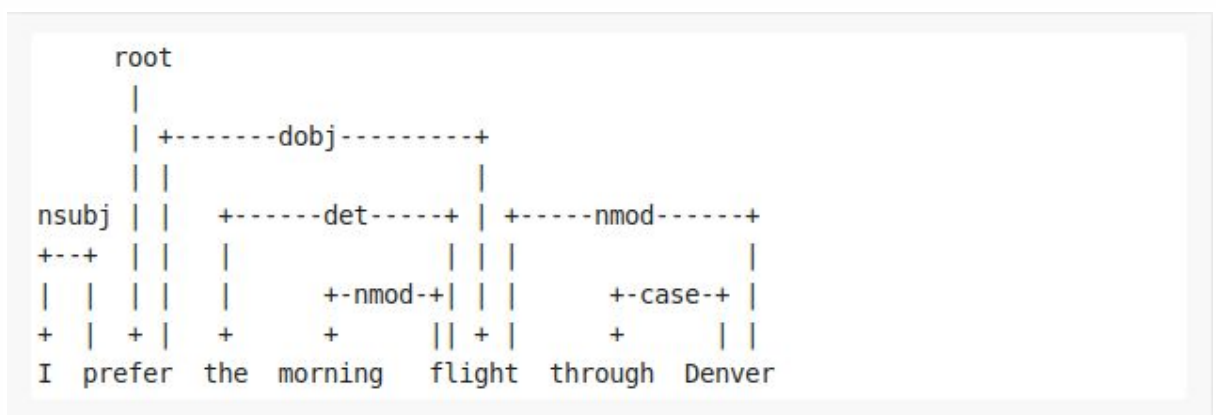


Fig.3.1 : Dependency parsing illustration

3.1.1.1.2 What is a Dependency Tree?

- Consists of lexical items linked by binary asymmetric relations called dependencies
- The arcs(links) indicate certain grammatical relations between words
- Each word depends on exactly one parent
- The tree starts with a root node

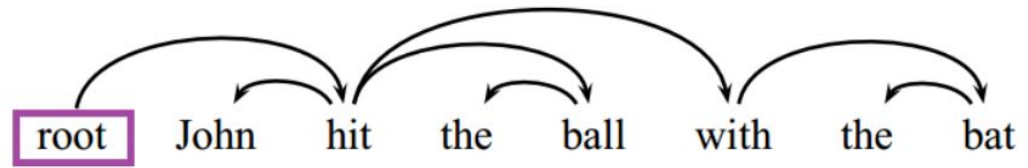


Fig 3.2 : Dependency tree illustration

Properties of a dependency tree

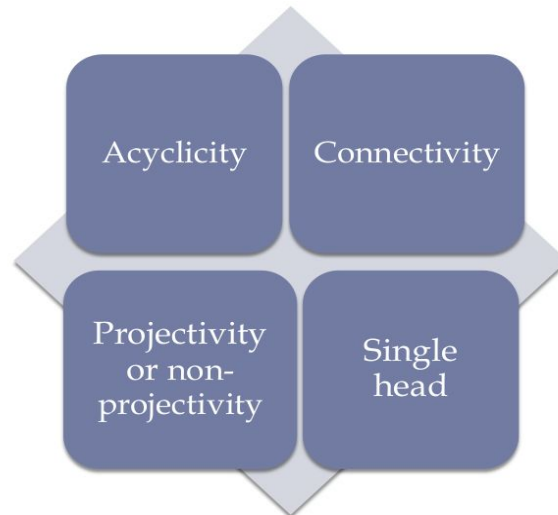


Fig 3.3 : dependency tree characteristics

3.1.1.1.3 Projectivity vs. Non-Projectivity

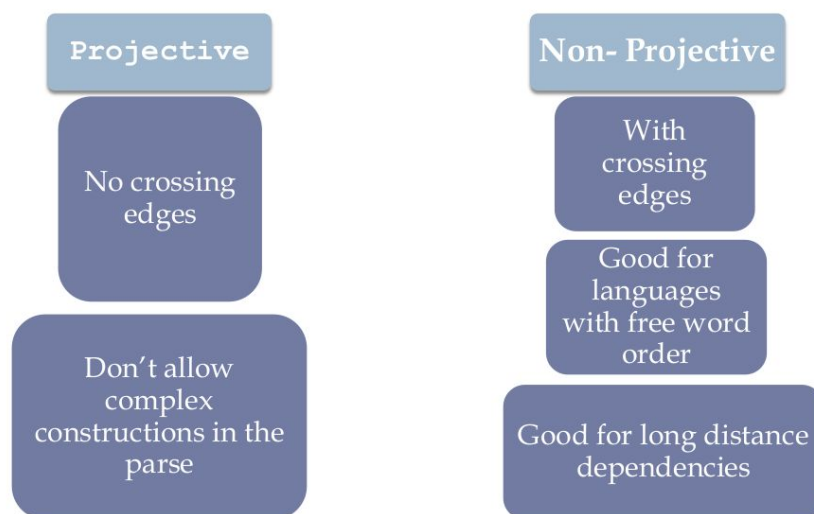


Fig 3.4 : illustration of the differences between projectivity and non-Projectivity property of dependency tree

Projective Dependency Tree

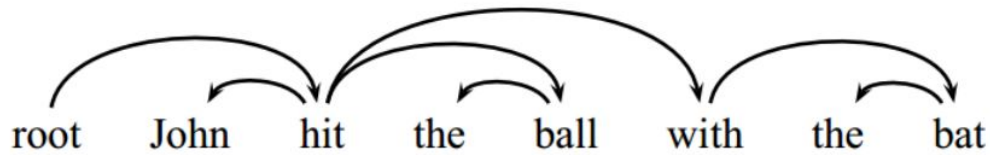


Fig 3.5 : projective dependency tree example

Non-Projective Dependency Tree

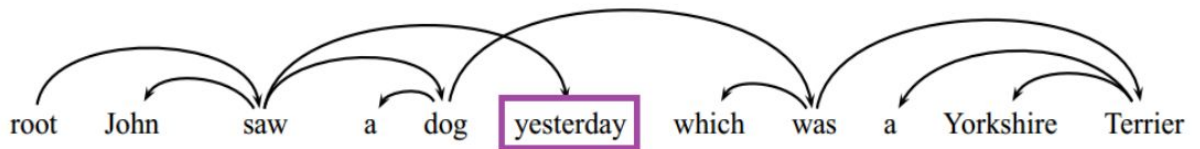


Fig 3.6 : Non-projective dependency tree example

3.1.1.2 Coreference Resolution

Is the task of finding all expressions that refer to the same entity in a text. It is an important step for a lot of higher level NLP tasks that involve natural language understanding such as document summarization, question answering, and information extraction.

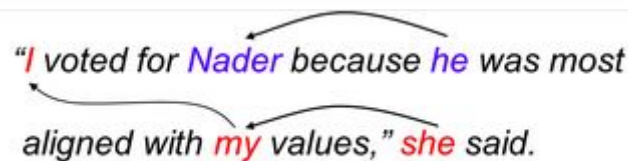


Fig 3.7 : Coreference resolution example

Linguistic Approach

- One of the first approaches to the problem
- Take advantage of linguistic of the text
 - ❑ Parse tree
 - ❑ Syntactic constraints
 - ❑ Semantic analysis
- Requires domain specific knowledge

Machine learning Approaches

1. Supervised learning approach

- ❑ Supervised learning is the machine learning task of inferring a function from a labelled training data
- ❑ The training data consist of a set of training examples
- ❑ A supervised learning algorithm analysis the training data and produces an inferred function , which can be used for mapping new examples

2. Unsupervised learning approach

- ❑ No supervision of wrong or right . Most are iterative methods.
- ❑ The objective function is to maximize the posterior probability of entities given collection of variables of current mention

Supervised Vs unsupervised approach

Unsupervised is generally preferred over supervised as :

- It doesn't need labeled data
- It doesn't need prior knowledge
- It doesn't subject to dataset limitation
- It often scales better than supervised approach

3.1.1.3 Tokenization

It is the first step of an NLP pipeline . It is the process of tokenizing or splitting a string, text into a list of tokens , so a word is a token in a sentence , and a sentence is a token in a paragraph.

Types of Tokenization :

1. **Sentence Tokenization** (Sentence segmentation)

Is the task of dividing a string of written language into its component sentences, and each sentence is called a token.

Example

Input : This is the first sentence. This is the second sentence.

Output : ['This is the first sentence', 'This is the second sentence']

2. **Word Tokenization**

Is the task of breaking up a sentence into words , and each word is called a token, perhaps at the same time throwing away certain characters, such as punctuation.

Example

Input: This is the first sentence.

Output: ['This' , 'is' , 'the' , 'first' , 'sentence']

3.1.1.4 Text Normalization (Word Normalization)

Is the process of transforming text into a single canonical form.

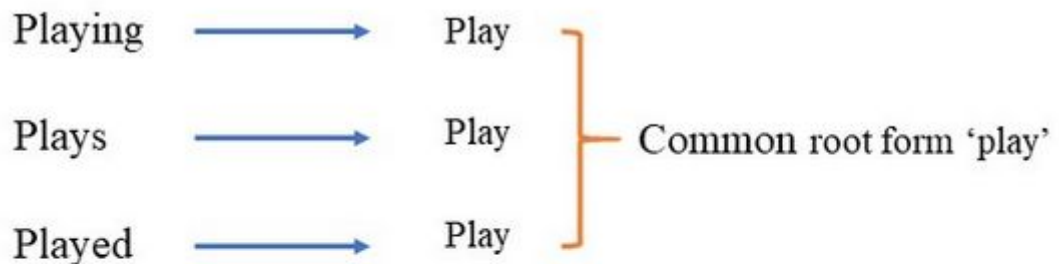
Inflected Language :

Languages we speak and write are made up of several words often derived from one another. When a language contains words that are derived from another word as their use in the speech changes is called **Inflected Language**.

In grammar, inflection is the modification of a word to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood. An inflection expresses one or more grammatical categories with a **prefix, suffix or infix**, or another internal modification such as a vowel change.

An inflected word(s) will have a common root form.

Examples



am, are, is → be

Car cars, car's, cars' → car

Fig 3.8 : example 1 on word normalization

Using above mapping a sentence could be normalized as follows

:

the boy's cars are different colors  the boy car be differ color

Fig 3.9 : example 2 on word normalization

Text Normalization techniques :

1. Stemming

Is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.

Example

cats -----> cat
Trouble , Troubling , Troubled ----->
troubl
University ,universal ,universities, universe ----->univers

2. Lemmatization

Unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called **Lemma**.

Example

runs, running, ran are all forms of the word *run*, therefore *run* is the lemma of all these words.

3.1.1.5 Part of Speech Tagging

It is a process of converting a sentence to forms – list of words, list of tuples (where each tuple is having a form (*word, tag*)). The tag in case of is a part-of-speech tag, and signifies whether the word is a noun, adjective, verb, and so on.

The part of speech explains how a word is used in a sentence. There are eight main parts of speech - nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions and interjections.

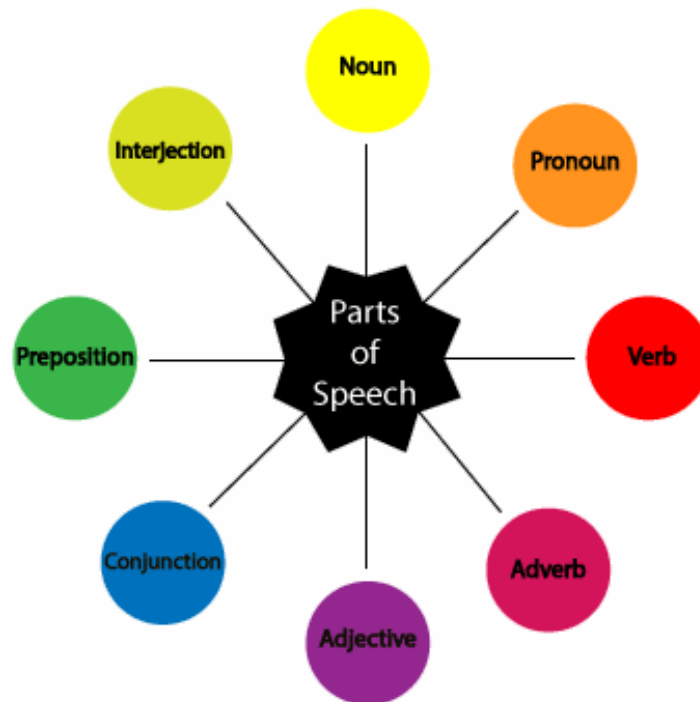


Fig 3.10 : explaining main part of speech

3.1.2 Planning

The planning in Artificial Intelligence is about the decision making tasks performed by computer programs to achieve a specific goal. The execution of planning is about choosing a sequence of actions with a high likelihood to complete the specific task.

3.1.3 Axis Aligned Bounding Box

An Axis Aligned Bounding Box (AABB) is a box that has its four axes aligned with the coordinate system in which it resides. This means it is a box that cannot rotate, and is always squared off at 90 degrees (usually aligned with the screen). In general it is referred to as a "bounding box" because AABBs are used to bound other more complex shapes.

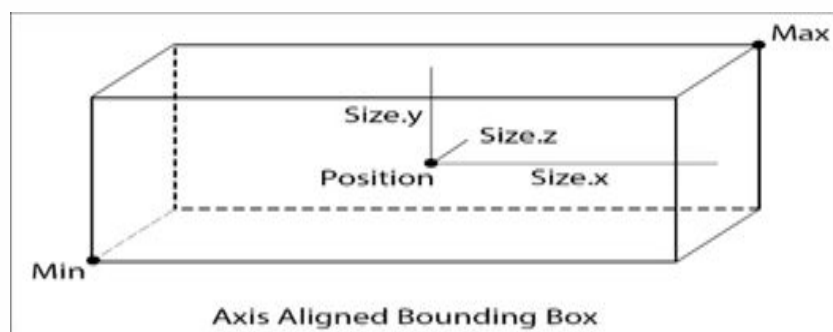


Fig 3.11 Axis Aligned Bounding Box

3.1.4 Impulse Resolution ^[1]

Impulse resolution is a particular type of collision resolution strategy. Collision resolution is the act of taking two objects who are found to be intersecting and modifying them in such a way as to not allow them to remain intersecting.

By resolving detected collisions we place a restriction upon movement such that objects cannot remain intersecting one another. The idea behind impulse resolution is to use an impulse (instantaneous change in velocity) to separate objects found colliding. In order to do this the mass, position, and velocity of each object must be taken into account somehow: we want large objects colliding with smaller ones to move a little bit during collision, and to send the small objects flying away. We also want objects with infinite mass to not move at all.

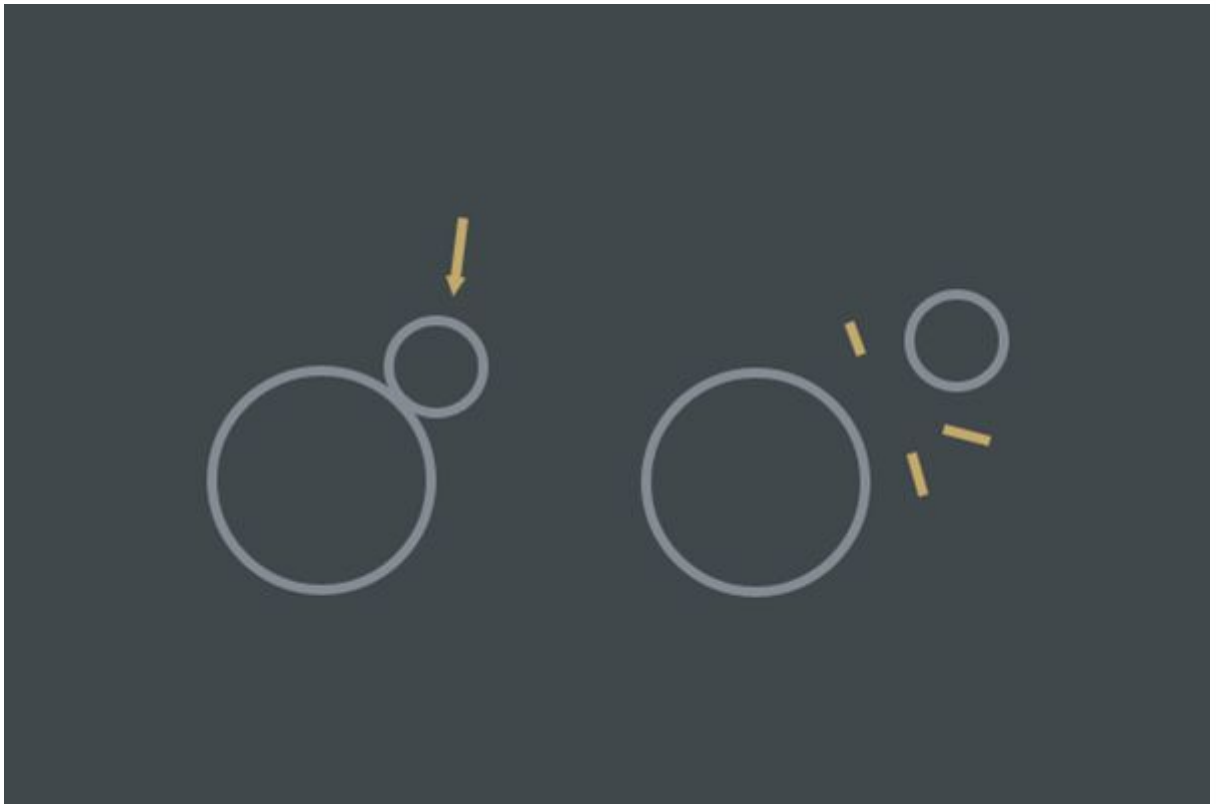


Fig 3.12 Impulse resolution

3.1.4.1 A basic understanding of simple vector math ^[2]

A vector is an object that has both a magnitude and a direction. Geometrically, we can picture a vector as a directed line segment whose length is the magnitude of the vector and with an arrow indicating the direction. The direction of the vector is from its tail to its head.



Vector math representation

Tail point $= (x_1, y_1, z_1)$

Head point $= (x_2, y_2, z_2)$

$X = x_2 - x_1$ $Y = y_2 - y_1$ $Z = z_2 - z_1$

$$V = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3.1.4.2 The ability to perform algebraic math on vectors

$$V_1 = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \quad V_2 = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix}$$

- Addition

$$V_1 + V_2 = \begin{bmatrix} X_1 + X_2 \\ Y_1 + Y_2 \\ Z_1 + Z_2 \end{bmatrix}$$

- Subtraction

$$V_1 - V_2 = \begin{bmatrix} X_1 - X_2 \\ Y_1 - Y_2 \\ Z_1 - Z_2 \end{bmatrix}$$

- Dot product

$$V_1 \cdot V_2 = x_1 * x_2 + y_2 * y_2 + Z_1 * Z_2$$

- Scaler

$$V_3 = e * V_1 = \begin{bmatrix} X_1 * e \\ Y_1 * e \\ Z_1 * e \end{bmatrix}$$

3.1.4.3 A basic understanding of simple physics

- **Newton's Law of Restitution [3]**

All this is saying is that the velocity after a collision is equal to the velocity before it, multiplied by some constant. This constant represents a "bounce factor".

$$V' = e * V$$

V: velocity before collision

e: bounce factor

V': velocity after collision

- **Momentum [4]**

Another vector measurement. Momentum is in the same direction as velocity. Scientists calculate momentum by multiplying the mass of the object by the velocity of the object. It is an indication of how hard it would be to stop the object.

3.1.5. Separating Axis Theorem [5]

The Separating Axis Theorem, SAT for short, is a method to determine if two convex shapes are intersecting. The algorithm can also be used to find the minimum penetration vector which is useful for physics simulation and a number of other applications.

3.1.5.1. Algorithm

SAT states that: “If two convex objects are not penetrating, there exists an axis or which the projection of the objects will not overlap.”

3.1.5.1.2. Intersection

, for all axes, the shape's projections overlap, then we can conclude that the shapes are intersecting.

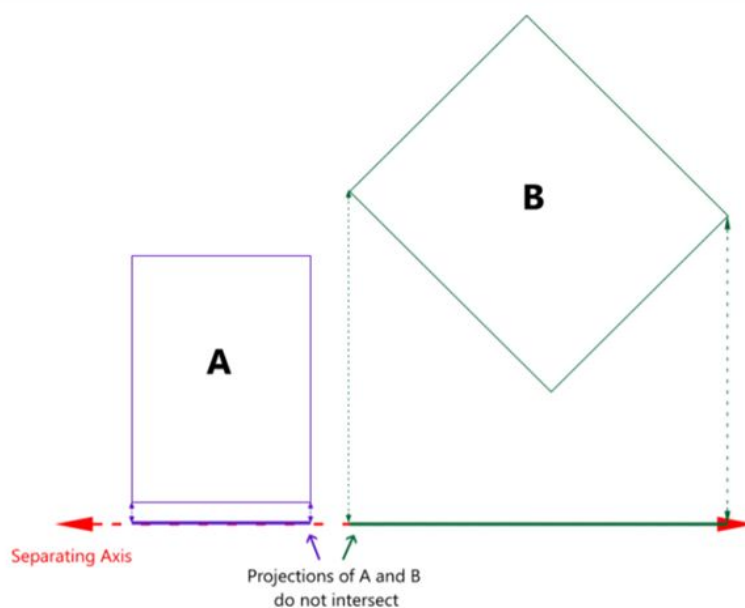


Fig 3.13 Representing two objects A and B projection

3.1.6 A* Search Algorithm

A* is a complete and efficient path finding algorithm and optimal for admissible heuristic.

A* Algorithm evaluate nodes by combining two functions

1. $g(n)$ “uniform Search”: the movement cost to move from the starting point to a given square on the grid, following the path generated to get there.
2. $h(n)$ “Greedy Best first search: the estimated movement cost to move from that given square on the grid to the final destination

$$f(n) = g(n) + h(n)$$

Algorithm

1. Select first node , evaluate f where $g(n) = 0$.
2. If (not at goal)
 - 2.1. Expand Selected node to its tree nodes & evaluate their $f(n)$, where $g(n)$ predefined & $h(n) = h$ (straight line distance between node and the goal)
 - 2.2. select min $f(n)$ node between all fringe nodes.
3. . Else exit with goal found.

3.2 Literature Review

3.2.1 Static scene positioning

In this paper [6] they use some approach for a reasonable positioning of objects in the environment according to spatial relations between them, this approach consists of three stages :

- Stage one is a polygon to voxel conversion process which converts the input polygon objects to their voxelized representations for processing. This stage is executed once during a preprocessing pass for each 3D model in the input collection.
- The second stage uses the voxel representation to extract the object's surfaces and partition them into spatially coherent regions.
- The third stage selects and validates a location from the regions that satisfies the spatial relationship for the object. The final stage adds the object to the scene.

3.2.2 Semantic Parsing for Text to 3D Scene Generation

In the work of paper [7] they propose text-to-scene generation as an application for semantic parsing . This is an application that grounds semantics in a virtual world that requires understanding of common, everyday language. In text to scene generation, the user provides a textual description and the system generates a 3D scene.

They first parse the input text into a scene template, which places constraints on what objects must be present and relationships between them. Next, using priors from a spatial knowledge base, the system expands the scene template by inferring additional implicit constraints. Based on the scene template, they select objects from a dataset of 3D models and arrange them to generate an output scene.

Chapter 4

Design Specifications



4.1 Assumption and/or Requirements

4.1.1 Model Assumptions

The input object must be in the set of our selected models in the database [bed, chair, ball, plate, TV, table, bat, box, car, toy, knife, desk, piano, gun] all the models of the same object are similar the only features that's different from one model to the other is the size and color of the object , environment models are football playground, room, street and kitchen and there's no alteration in the environment models ,human models are provided according to 4 features gender , age , hair color and size the only significant change in a model is according to the gender and the age feature change in hair or size will result in the same model but with different hair and different size.

4.1.2 Action Assumptions

The set of action of which the story can be composed of is limited to running , walking, sitting, standing, sleeping, waking, kicking, hitting, talking, car movement, throwing, talking, pointing, playing with a toy, dancing, clapping, falling , fighting, puching, jumping, waving, crying and piano playing

4.1.3 Written text Assumptions

All the features of an object must be specified at the first time of its mention in the input text

For example the following case isn't valid :

There are two boxes. The first box is red. The second box is green.

Must be :

There is a red box. There is another green box.

4.1.4 Spatial relations Assumptions

Relation between models is on,in,behind,front, right or left. There must be an environment model that is of the largest width *length. If the relation between objects and a container object the addition of their length *width is smaller than or equal to the container's length *width.

4.2. Block Diagram

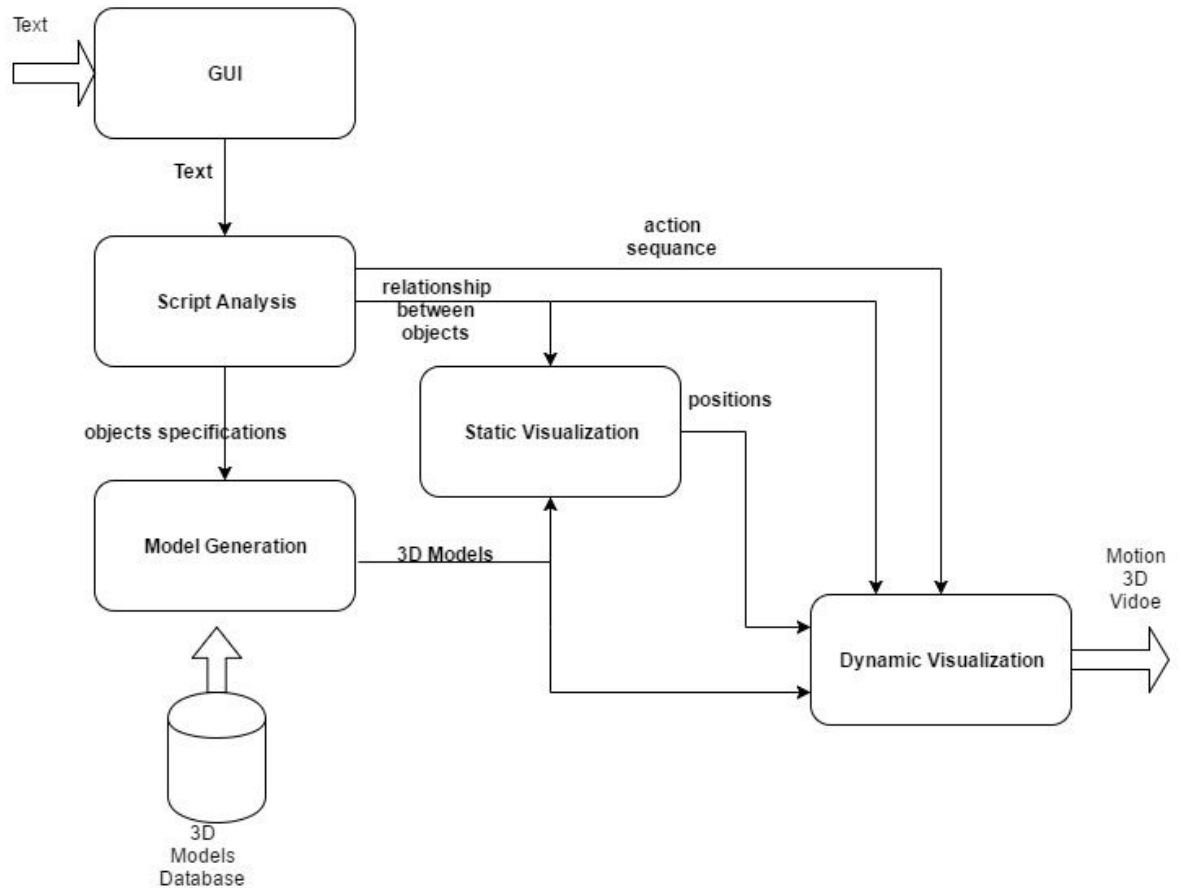


Fig 4.1 : block diagram of our system

4.2.1 Block description

4.2.1.1 GUI

Description :

This module is the interface that the user deals with. It's implemented using Kivy library. It contains text editor to take text input from user and a video player to display the program output.

- **Input :** Text file containing short paragraphs for a short story .
- **Output :** Video file.

4.2.1.2 Script Analysis module

Description :

This module is responsible for syntactic and semantic analysis of the input text , as syntactic analyzer is responsible for dependency parsing which is a graph of words and relations (dependencies) between them within a sentence , then the semantic analyzer uses the parsing graph output and detect the models in the scene, their characteristics , spatial relations , actions , actions sequence and also infer models and environment which aren't explicitly mentioned in the text .

- **Input :** Text file containing short paragraphs for a short story .
- **Output :**
 1. File containing the names of the models (explicitly mentioned and inferred) in the scene associated with their characteristics and their unique id .
 2. File containing the spatial relations between models in the scene
 3. File containing actions associated with their subject and object(s) mentioned in the input text .
 4. Sequence of actions

4.2.1.2.1 syntactic analyzer

In this stage we parse the input text and get the dependency parsing tree to be used for the following stages in this module .

4.2.1.2.2 Detect models with their characteristics in scene

In this stage we detect the names of the models with their characteristics and give them a unique id in the form of a map which maps each model to its unique id .

The main followed steps as the following :

1. We specify synonyms for the available humans models and their associated features in our dataset , we specify synonyms for boy , girl , man and woman , and we also specify synonyms for mutual models' type that may be a boy or a girl , or a man or a woman .
2. We get the coreference resolution (*he , she , it pronouns*) mentioned in the text and the referenced noun in order to help us determine the gender of mutual models' type .
3. We get dependency parsing tree generated from syntactic analyzer
4. We make sentences tokenization and get tokenized sentences

5. For each tokenized sentence :
 1. Detect model type
 - We first detect model type by searching in available synonyms of humans for a unique model type and return the corresponding model type
 - If it isn't found in previous search , we search in the synonyms for mutual models' type for humans , and if it found then we get its gender from the coreference get in step 2 , otherwise return its default .
 - If it isn't detected as a human then return its name if it is included in our dataset otherwise neglect it.
 2. Detect model characteristics
 - We set a default characteristics for a human as [not old , no hair color , medium height] , and for non human models as [no color , medium size]
 - We search for the available features for each model type and neglect the others.
 3. Get a unique id for each model
 4. Append the model name , model characteristics and unique id and save it as an explicit model in the scene .
6. Detect inferred models (aren't explicitly mentioned)
 - We use statistical learning to get the most likely support for an explicit object (inferred object)
 - Search if there is a relation between this object and the inferred one in the list of the models' relations extracted from the stage of finding the spatial relations between models , if found a relation this don't add the inferred object as a new model in the scene otherwise add it with a default features.

4.2.1.2.3 Extract actions associated with their subject and object mentioned in the input text

In this stage we detect the actions mentioned in the input text with their associated subject and object .

The main followed steps as the following :

1. Define set of available actions as mentioned in the Action Assumptions.
2. We get coreference map which map each object to its pronoun to help us know the referenced subject or object associated with the verb .

3. We get dependency parsing tree generated from syntactic analyzer.
4. We make sentences tokenization and get tokenized sentences.
5. For each tokenized sentence :
 - Find a token with verb tag , and check if it is found in the available actions then proceed in the following steps , otherwise neglect it.
 - For each verb tag
 - ❑ Find its object if it isn't referenced , otherwise find its reference by using the coreference map from step 1 by knowing the index of the current pronoun.
 - ❑ Find its subject(s) if they aren't referenced , otherwise find their references by using the coreference map from step 1 by knowing the index of the current pronoun(s) .
 - ❑ Append the action name with the associated subject and object as an extracted action in the action list.

4.2.1.2.4 Extract the spatial relations between models in the scene

1. Define set of available spatial relations as mentioned in the spatial relations Assumptions.
2. We get dependency parsing tree generated from syntactic analyzer.
3. We make sentences tokenization and get tokenized sentences.
4. For each tokenized sentence :
 - Find a token with ADP tag , and check if it is found in the available spatial relations then proceed in the following steps , otherwise neglect it.
 - ❑ Find model name before the preposition by traversing the dependency parsing tree.
 - ❑ Find model name after the preposition by traversing the dependency parsing tree.
 - ❑ Append the name of the spatial relation with the associated nouns participate in this relation as an extracted spatial relation in the spatial relation list.

4.2.1.3 Model Generation

The model generation is part that's responsible for interpreting the characters specifications to 3D models, first it searches for the object that's intended then it applies deformation in the model shape to fit the intended size and then it applies the intended modification color in the object materials and produces a 3D model.

For example a model with the specification :

object = 'boy'

hair color ='red'

size = 'tall'

The module retrieves the boy model from the dataset then it applies the transformation in shape converting the shape to a tall boy and then check whether the boy has red hair or not if he doesn't it changes the hair to red.

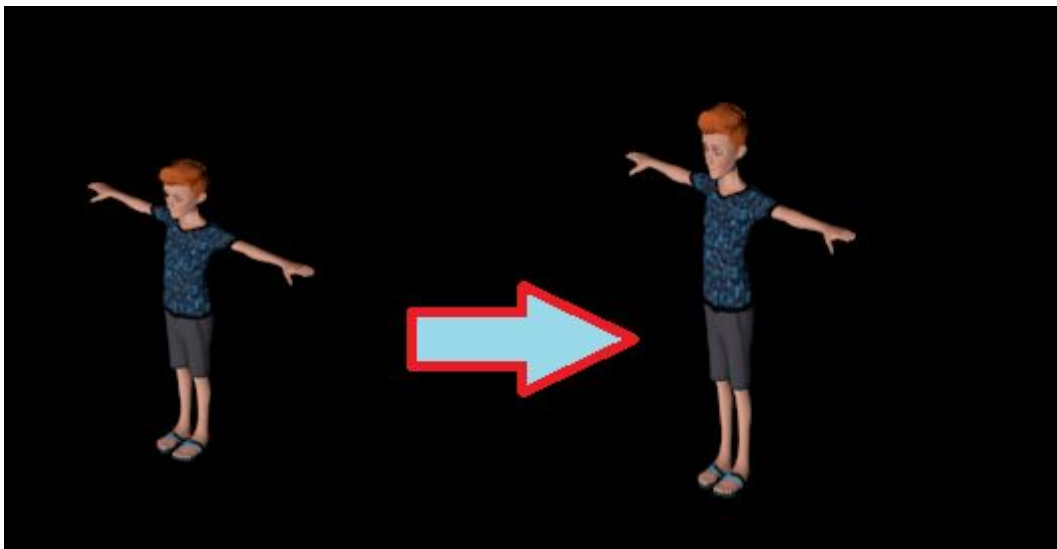


Fig 4.2 : Boy model before and after scale deformation

4.2.1.4 Static Visualization

Description :

It is the module that is responsible for composing the static scene before applying the actions to the scene. The operation passes by the following steps:

4.2.1.4.1 Voxilization

A preprocessing step in which converting 3D models into voxels objects, to make it easier later at dealing with them as 3D arrays.

4.2.1.4.2 Spatial positioning

Using the spatial information that is extracted by Script analysis module, the module tries to find the best location for each object in the scene. It takes into consideration:

- The spatial relations
- Boundary box and dimensions of each object
- Free spaces and occupied spaces of environment

To make good and reasonable allocations of objects and avoiding collisions

- **Input** : spatial relations between models.
- **Output** : 3D coordinates of each model.

4.2.1.5 Dynamic Visualization

Description :

The part where the scene is constructed and rendered to mp4 video, first loading the 3d models and the action's animation of the basic action in the planning output, unlinking all the action from scene apply any deformation needed on models, construct plans from every action so that every action is constructed from basic action, loading and parsing the action animation tree for every action, setting nla (non linear action) strips for every model according action planning, apply collision detection and impulse resolution ,construct the path finding for movement actions and then render the scene into mp4 video.

4.2.1.4.1 Planning

This part is responsible for creating plans for every action in the action sequence since the input actions are composed of basic action for example the boy is sitting on chair number which exists in location x_1 , y_1 and the action boy sets on char 2 which exists in location x_2, y_2 , so the boy must get up from chair one walks to chair 2 location (x_2, y_2) rotate to site on the chair then sites on the chair .

This module is implemented using Graph planning algorithm with classical planning assumptions.

4.2.1.4.2 Action animation tree

A tree contains the actions their effect on objects and their animation
Example : a shoots b

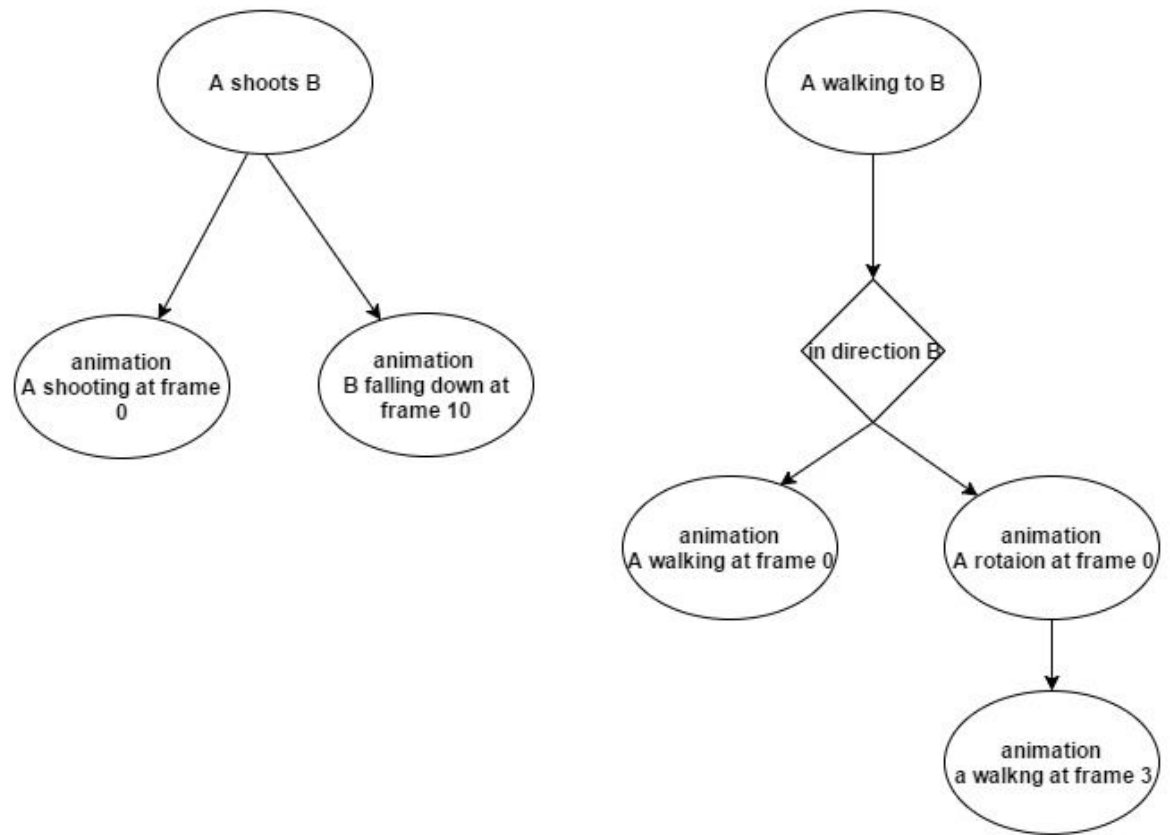


Fig 4.3 : illustration action tree for actions shooting and walking

4.2.1.4.1 Setting NLA (non linear action) strips

After receiving the list of animation for every action in every plan ,we need to apply those actions on our models this is done by extraction the stripes of the animation and apply them to our model this is achievable by extraction the frame of the animation and apply them on the intended model NLA (none linear actions) framed.

4.2.1.4.4 Collision detection and impulse resolution.

The non animated object (none humans) most of those objects doesn't have any animation frames associated to them their animation is just the matter of interacting with humans and other objects ie (Collision detection and rebound) this is done by first detecting collisions then calculating the new velocity by calculating the momentum and its effect on the object velocity.

4.2.1.4.5. Path finding

Path finding is executed using A* algorithm with Chebyshev distance as main heuristic function and the path coast is set to 1 in normal step and increased for rotation steps by 2 to reduce the rotations in the movement this

also reduces the time consumed by the path because rotation movement take additional frames to animate rotation movement so the shortest path is no't the effect path, the search grid represents every millimeter on the environment's xy plane it's set to zero and every obstacle projection on xy plane is set it its id, so that objects all allowed to walk on its location, the movement step in the 8 directions and size of the movement is the area of the moving object (it's projection in the xy plane).

Chapter 5

Tools



5.1 Overview

This chapter presents the tools and programs that have been used in building **Animare**. It starts with the IDEs and server used. Then, the libraries and their importance in the project.

5.2 System Tools and Libraries

5.2.1 PyCharm

- PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.
- We use PyCharm as python IDE .



5.2.2 Spacy

- SpaCy is an open-source library for advanced Natural Language Processing in Python.
We use spacy in the Syntactic Analysis stage to produce dependency parsing tree , and for sentence tokenization,
- use neuralcoref from it to for coreference resolution analysis.



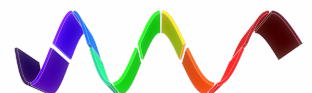
5.2.3 NLTK

- Is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.
- We use WordNetLemmatizer from nltk.stem for word stemming .



5.2.4 Colour 0.1.5

- Is a Python colour science package implementing a comprehensive number of colour theory transformations and algorithms. It is open source and freely available and under the New BSD License terms.
- We use Colour package to check if a word is a color or not in feature extraction stage.



5.2.5 GitHub

- GitHub is an American company that provides hosting for software development version control using Git.
- We use GitHub to manage our project files .



5.2.6 Blender

- Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games.
- We use for preprocessing of 3D models and as a viewer for scene debugging.



5.2.7 bpy

- Blender's Python Library
- We use it for loading models, applying animations and rendering the scene to a video.

5.2.8 binvox

- binvox is a program that reads a 3D model file, rasterizes it into a binary 3D voxel grid, and writes the resulting voxel file.
- We use it in preprocessing to convert models to voxel files.

5.2.9 Kivy

- Open source Python library for rapid development of applications that make use of innovative user interfaces, such as multi-touch apps.
- We use it for creating GUI.



5.3 Physics Engine

5.3.1 Collision Detection

First step is the calculation of Axis Aligned Bounding Box (AABB) of every object.

Every AABB can be represented by two points the maximum point and the minimum

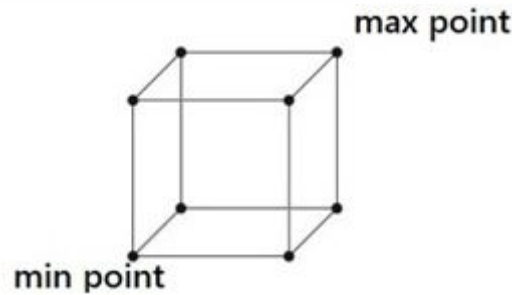


Fig 5.1: Axis Aligned Bounding Box (AABB)

This form allows an AABB to be represented by two points. The min point represents the lower bounds of the x, y and z axis, and max represents the higher bounds - in other words, they represent the top left and bottom right corners.

Collision detection is calculated according to Separating Axis Theorem by checking where the projections of the AABB intersect in all of the three axes XYZ.

5.3.2 Impulse Resolution

We'll start with our two objects that have been found to be intersecting:

Equation 1

$$\mathbf{V}^{AB} = \mathbf{V}^B - \mathbf{V}^A$$

\mathbf{V}^{AB} is the relative velocity from A to B. This equation ought to be expressed in terms of the collision normal \mathbf{n} - that is, we would like to know the relative velocity from A to B along the collision normal's direction:

Equation 2

$$\mathbf{V}^{AB} \cdot \mathbf{n} = (\mathbf{V}^B - \mathbf{V}^A) \cdot \mathbf{n}$$

Equation 3

$$\mathbf{V}' = \mathbf{e} * \mathbf{V}$$

All this is saying is that the velocity after a collision is equal to the velocity before it, multiplied by some constant. This constant represents a "bounce factor". Knowing this, it becomes fairly simple to integrate restitution into our current derivation:

Equation 4

$$\mathbf{V}^{AB} \cdot \mathbf{n} = -e * (\mathbf{V}^B - \mathbf{V}^A) \cdot \mathbf{n}$$

Negative signed because In Newton's Law of Restitution, \mathbf{V}' , the resulting vector after the bounce, is actually going in the opposite direction of \mathbf{V} . So we represent opposite directions by a negative sign.

Now we need to be able to express these velocities while under the influence of an impulse. Here's a simple equation for modify a vector by some impulse scalar j along a specific direction \mathbf{n} :

Equation 5

$$\mathbf{V}' = \mathbf{V} + j * \mathbf{n}$$

We have a unit vector \mathbf{n} which represents a direction. We have a scalar j which represents how long our \mathbf{n} vector will be. We then add our scaled \mathbf{n} vector to \mathbf{V} to result in \mathbf{V}' . This is just adding one vector onto another, and we can use this small equation to apply an impulse of one vector to another.

There's a little more work to be done here. Formally, an impulse is defined as a change in momentum.

Equation 6

$$\text{Impulse} = \text{mass} * \text{Velocity}.$$

$$\text{Velocity} = \frac{\text{Impulse}}{\text{mass}} \quad \therefore \mathbf{V}' = \mathbf{V} + \frac{j * \mathbf{n}}{\text{mass}}$$

We need to be able to express an impulse using j in terms of two different objects. During a collision with object A and B, A will be pushed in the opposite direction of B:

Equation 7

$$V'A = V_A + \frac{j \cdot n}{\text{mass } A}$$

$$V'B = V_B - \frac{j \cdot n}{\text{mass } B}$$

These two equations will push A away from B along the direction unit vector n by impulse scalar (magnitude of n) j.

• Equation 8

$$(V^A - V^B + \frac{j \cdot n}{\text{mass } A} + \frac{j \cdot n}{\text{mass } B}) \cdot n + e \cdot (V^B - V^A) \cdot n = 0$$

Now need to solve the magnitude of this direction. The magnitude which is an unknown in our case is j. we must isolate j and solve for it.

• Equation 9

$$(V^B - V^A) \cdot n + j \cdot \left(\frac{j \cdot n}{\text{mass } A} + \frac{j \cdot n}{\text{mass } B} \right) \cdot n + e \cdot (V^B - V^A) \cdot n = 0$$

$$\therefore (1+e) ((V^B - V^A) \cdot n) + j \cdot \left(\frac{j \cdot n}{\text{mass } B} + \frac{j \cdot n}{\text{mass } B} \right) \cdot n = 0$$

$$\therefore j = - \frac{(1+e)((V^B - V^A) \cdot n)}{\frac{1}{\text{mass } A} + \frac{1}{\text{mass } B}}$$

Now we can calculate the velocity of by calculating the impulse and subtracting it from each object (substituting in equation 7).

5.4 Programming Languages

5.4.1 Python 2.7, 3.6

- Python is an interpreted, object-oriented, high-level, general-purpose programming language.
- We use python as a programming language for implementing NLP module with its stages, Static visualization module, Dynamic visualization module, Model vgeneration module and GUI.



Chapter 6

Testing and Results



6.1 Test scope

Features tested:

- GUI:
 - Input validation.
 - Correct interaction between activities.
- Script Analysis Module:
 - Extracting objects , their characteristics and their spatial relations.
 - Extracting actions sequence.
 - Extracting inferred objects.
- Static Visualization Module :
 - Getting reasonable 3D coordinations for each object.
- Dynamic Visualization:
 - Adding animations and applying actions to the scene.
 - Reasonable animations
 - Good physics .

6.2 Testing Methodology

Functional Testing:

- Unit testing: Testing each module independent of other modules.
- Regression testing: Re-testing modules after modifying them.
- Integration testing: Testing of different modules integrated together.

6.3 Test cases and results

6.3.1 Script Analysis Unit testing

6.3.1.1 Output description :

1. Model characteristics

[Model_name,Model_characteristics,Model_id]

where Model_characteristics for

- human : gender , age , hair color , height , copy_number
- not human : color , size , copy_number

2. Models spatial relations

[relation_name,Model_name_beforePreposition,Model_id_BeforePreposition,Model_name_AfterPreposition,Model_id_AfterPreposition]

3. Infered models and spatial relations

- Infered models with the same format as Model characteristics
- Infered spatial relations with same format as models spatial relations

This output is added to **Model characteristics** and **Models spatial relations** output file.

4. Models Actions

[Action_name , Subject_id , Object1_id , Object2_id , Action_position_InText]

5. Actions sequence

[Sequence number , Subject's ID , Action name , Object 1 ID , Object 2 ID]

6.3.1.2 Test cases :

Test number	1
Description	This test case tests if the module identify more than one instance of the same object and differentiate between them in the spatial relations and extracting a right spatial relations , it also tests that the pronoun (he) refers to the appropriate noun
Input	<p><u>Short story</u></p> <p>There was a large blue table in a playground. There were three white bottles. The first bottle and the second bottle were on the table.</p> <p>There was another small red table. Also, there was a small yellow ball. The ball and the third bottle were on the red table.</p> <p>There were two large brown chairs. The first chair was on the right of the small table , while the second chair was on the left of the small table.</p> <p>There was a tall player with a red hair. He was on the left of the blue table. There was another short player with a black hair.</p> <p>The short player was sitting on the second chair. Then , the short player was waving to the tall player . So , the tall player also waved to the short player.</p> <p>Then , the short player ran towards the tall player. After that , the tall player was talking to the short player.</p>

Output	<p>1. <u>Models characteristics</u></p> <p>table blue 2 1 playground none 1 2 bottle white 1 3 bottle white 1 4 bottle white 1 5 table red 0 6 ball yellow 0 7 chair brown 2 8 chair brown 2 9 man 0 red 2 10 man 0 black 0 11</p> <p>2. <u>Models spatial relations</u></p> <p>in table 1 playground 2 on bottle 4 table 1 on bottle 3 table 1 on bottle 5 table 6 on ball 7 table 6 right chair 8 table 6 left chair 9 table 6 left man 10 table 1</p> <p>3. <u>Infered models and spatial relations</u></p> <p>infered_models : [] infered_relations : []</p> <p>4. <u>Models Actions</u></p> <p>sit 11 9 -1 126 wave 11 10 -1 138 wave 10 11 -1 150 run 11 10 -1 162 talk 10 11 -1 175</p> <p>5. <u>Actions sequence</u></p> <p>0 1 sit 5 -1 0 2 sit 6 -1 0 3 sit 7 -1 1 3 walk 8 -1 2 1 sit 7 -1 2 2 dance -1 -1</p>
---------------	---

State	PASSED
--------------	---------------

Table 1 - test case 1

Test number	2
Description	This test case tests noun synonyms and detecting the right gender , it also tests the action sequence .
Input	<p><u>Short story</u></p> <p>There is a tall boy and his beautiful short sister with a black hair and their tall mother with a red hair in a kitchen. There are two small white chairs and a red big chair. also, there is a large brown table and a small black table. There are two big white plates. The first plate and the second plate on the brown table. There is some food on the first plate. There is a small white television on the black table. The first white chair and the second white chair are behind the brown table. Also , the first white chair is on the right of the second white chair. The boy is sitting on the first white chair but the girl is sitting on the second white chair and the mother is sitting on the red chair. After sometime , the mother walks towards the brown table , then the boy sits on the red chair and the girl was dancing at the same time.</p>
Output	<p>1. <u>Model characteristics</u></p> <p>boy 0 none 2 1 girl 0 black 0 2 woman 0 red 2 3 kitchen none 1 4 chair white 0 5 chair white 0 6 chair red 2 7 table brown 2 8 table black 0 9 plate white 2 10 plate white 2 11 food none 1 12 television white 0 13</p>

	<p>2. <u>Models spatial relations</u></p> <p>on plate 10 table 8 on plate 11 table 8 on food 12 plate 10 on television 13 table 9 behind chair 6 table 8 behind chair 5 table 8 right chair 5 chair 6</p> <p>3. <u>Infered models and spatial relations</u></p> <p>infered_models : [] infered_relations : []</p> <p>4. <u>Models Actions</u></p> <p>sit 1 5 -1 130 sit 2 6 -1 140 sit 3 7 -1 150 walk 3 8 -1 162 sit 1 7 -1 171 dance 2 -1 -1 180</p> <p>5. <u>Actions sequence</u></p> <p>0 1 sit 5 -1 0 2 sit 6 -1 0 3 sit 7 -1 1 3 walk 8 -1 2 1 sit 7 -1 2 2 dance -1 -1</p>
State	PASSED

Table 2 - test case 2

Test number	3
Description	This test case tests the scene inference including models inference and spatial relation inference.
Input	<p><u>Short story</u></p> <p>There is an old tall woman with a white hair. There is a smart short guy.</p>

	<p>There is a large pink car and another small black car. The guy is behind the large car while the woman is in front of the black car.</p> <p>There is a blue chair on the right of the woman. The woman is tired so , she sits on the chair. The guy wants to help the woman going to the large car. So , the guy walks towards the woman , then the guy and the woman walks towards the pink car. The woman becomes so happy.</p>
Output	<p>1. <u>Model characteristics</u></p> <p>woman 1 white 2 1 boy 0 none 0 2 car pink 2 3 car black 0 4 chair blue 1 5 street none 1 6</p> <p>2. <u>Models spatial relations</u></p> <p>behind boy 2 car 3 front woman 1 car 4 right chair 5 woman 1 in car 3 street 6 in car 4 street 6</p> <p>3. <u>Inferred models and spatial relations</u></p> <p>inferred_models : [['street', 'street', ['none', -1, 'first'], 6]] inferred_relations : [['in', 'car', 3, 'street', 6], ['in', 'car', 4, 'street', 6]]</p> <p>4. <u>Models Actions</u></p> <p>sit 1 5 -1 69 walk 2 1 -1 91 walk 2 3 -1 102 walk 1 3 -1 102</p> <p>5. <u>Actions sequence</u></p> <p>0 1 sit 5 -1 0 2 walk 1 -1 1 2 walk 3 -1 1 1 walk 3 -1</p>

State	PASSED
-------	--------

Table 3 - test case 3

Test number	4
Description	This test cases tests the scene inference with only spatial relation inference but with existing model .
Input	<p><u>short story</u></p> <p>There is a smart tall boy in a room. There are two huge red chairs and a small black table and a big brown table. The black table is behind the first chair.</p> <p>There is a small white ball on the black table. A green ball on the brown table. The green ball is on the right of a small white television.</p> <p>There is an old short man. The boy walks towards the first chair and sits on the first chair.</p> <p>The old man walks towards the second chair and sits on the second chair.</p>
Output	<p>1. <u>Model characteristics</u></p> <p>boy 0 none 2 1 room none 1 2 chair red 2 3 chair red 2 4 table black 0 5 table brown 2 6 ball white 0 7 ball green 1 8 television white 0 9 man 1 none 0 10</p> <p>2. <u>Models spatial relations</u></p> <p>in boy 1 room 2 behind table 5 chair 3 on ball 7 table 5 on ball 8 table 6 right ball 8 television 9 on television 9 table 5</p>

	<p>3. <u>Infered models and spatial relations</u></p> <p>infered_models : [] infered_relations : [['on', 'television', 9, 'table', 5]]</p> <p>4. <u>Models Actions</u></p> <p>walk 1 3 -1 79 sit 1 3 -1 85 walk 10 4 -1 95 sit 10 4 -1 101</p> <p>5. <u>Actions sequence</u></p> <p>1 1 walk 3 -1 2 1 sit 3 -1 2 10 walk 4 -1 3 10 sit 4 -1</p>
State	PASSED

Table 4 - test case 4

6.3.1 Model Generation Unit testing

Test number	5
Description	Testing generation of models
Inputs	Models_chars (type hair age' size): boy green 1 1 man none 1 1 woman green 1 1 table white None 2 chair red None 0 bed blue None 1
Output	

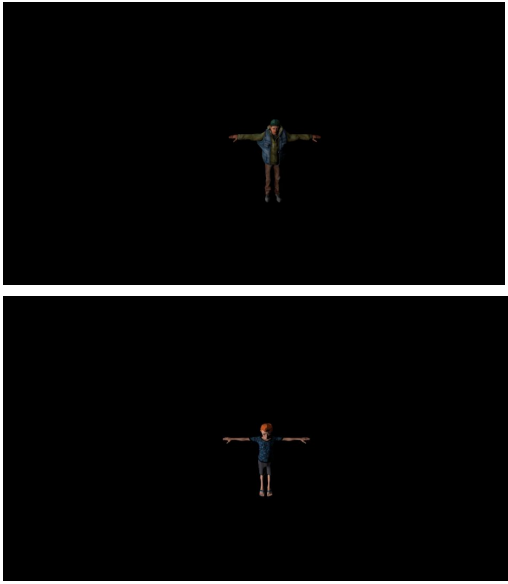
	
State	Passed

Table 5 - test case 5

6.3.1 Static visualization Unit testing

Test number	5		
Description	Testing valid output coordination for spatial relations input.		
Input	Relations: on plate 10 table 8 on plate 11 table 8 on sandwich 12 plate 10 on television 13 table 9 behind chair 6 table 8 behind chair 5 table 8 right chair 5 chair 6	Models: boy 1 woman 3 chair 5 chair 7 table 9 plate 11	girl 2 kitchen 4 chair 6 table 8 plate 10 sandwich 12 television 13



Output	
States	PASSED
Note:	<p>In comparison of our work with Wordseye we give it the same description but it couldn't analyse it, so we made the description simpler but it couldn't allocate more than five objects, while in this scene Animare allocated 13 objects.</p> 

Table 6 - test case 6

6.3.1 Dynamic visualization Unit testing

Testing path finding in dynamic visualization module

6.3.1.1 A* Algorithm

The testing is done using pathfinding library which has a built in A*

by running the two programs (built in A* and our A*) 1000 times with an input Matrix [100][100] with random obstacles and new start and end each time

The testing Output:

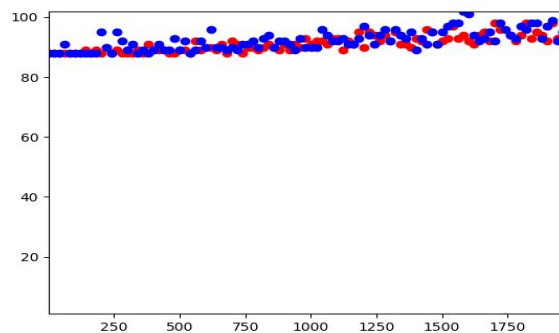


Fig 6.1 : The result of plotting length of two programs path with the number of obstacles

6.3.1.2 Applying Actions


Test number	7
Description	Testing applying sequence of actions on a boy model
Inputs	ActionSequence: 0 boy clap 1 boy jump 2 boy run chair
Output	 Video's Link: http://bit.do/eYwYM-
State	Passed

Table 7 - test case 7


Test number	8
Description	Testing applying sequence of actions on a boy model
Inputs	ActionSequence: 0 boy clap 1 boy jump 2 boy sit chair
Output	 <p>Video's Link: <u>http://bit.do/eYw8n-</u></p>
State	Passed

Table 8 - test case 8

6.3.2 Integration Testing


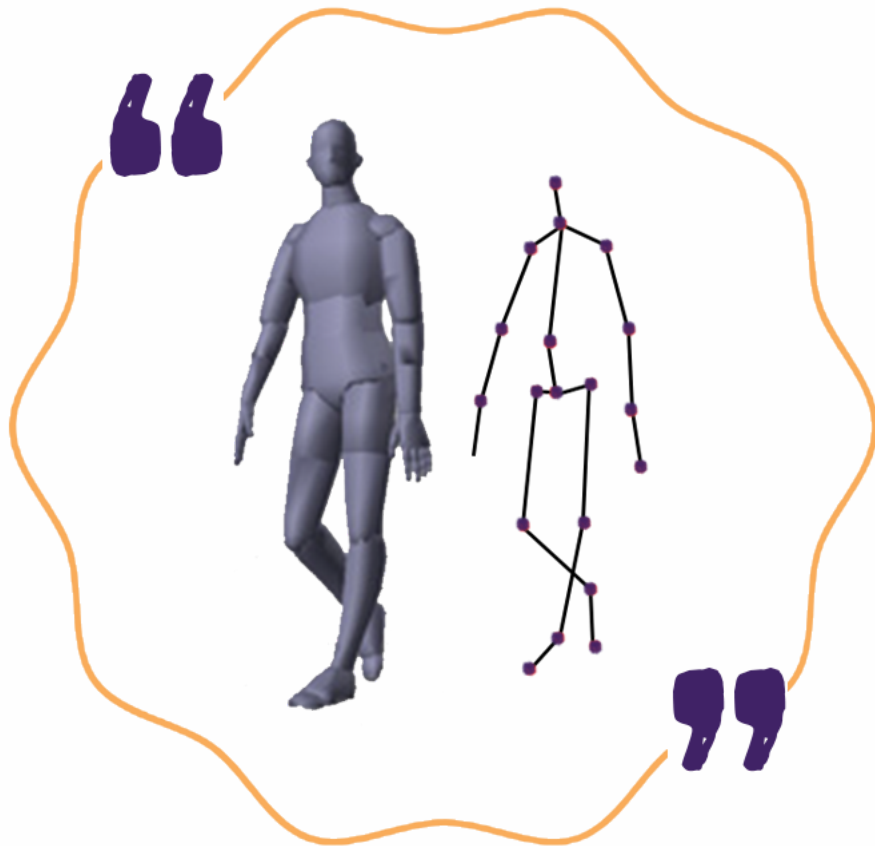
Input	<p><u>short story:</u></p> <p>There is a tall boy with his beautiful short black hair sister and their tall mother with red hair in a kitchen. There are two small white chairs and one large red chair. There are also a large brown table and a small black table.</p> <p>There are two big white plates. The first plate and the second plate are on the brown table. There is also some food on the first plate.</p> <p>There is a small white television on the black table. The first white chair and the second white chair are behind the brown table.</p> <p>In addition, the first white chair is on the right of the second white chair.</p> <p>The boy is sitting on the first white chair but the girl is sitting on the second white chair and the mother is sitting on the red chair.</p> <p>After sometime, the mother walks towards the brown table, then the boy walks to the red chair and the girl is jumping at the same time.</p>
Output	 <p>Video's Link:</p> <p>http://bit.do/eYEAM-</p>
State	PASSED

Table 9 - test case 9

Chapter 7

Conclusion and Future Work



7.1 Conclusion

Animare is feasible and has been acceptable to our friends as a preliminary user usage. **Animare** is very good related to script analysis , positioning and animation comparing with our competitors (like Wordseye). We found that many people are interested and in need to some tool like **Animare** to help them in learning and entertainment,so as long as it supports a wide range of modules and animations it would be interesting and a way for creativity to shine.

7.2 Future Work

We have several future work directions. Following is a recommend list:

- Add More platforms (mobile app, website...)
- Improve positioning by training on 3D scenes datasets. It is suggested to collect and construct 3D scenes for various environments and situations to create a dataset, then use it later in training to make better positioning.
- Enhance physics of the models and increase the number of models and their corresponding actions in the available pool
- Add a facility to the user to be able to upload models
- Provide an Arabic text description handler
- Provide a voice input handler.

References

- [1]: Collision Resolution
- [2]: A basic understanding of simple vector math
- [3]: Newton's Law of Restitution
- [4]: Momentum
- [5]: Separating Axis Theorem for Oriented Bounding Boxes
- [6]: Lee M. Seversky and Lijun Yin. Real-time Automatic 3D Scene Generation from Natural Language Voice and Text Descriptions
- [7]: Angel X. Chang, Manolis Savva and Christopher D. Manning. Semantic Parsing for Text to 3D Scene Generation
- [8]: O. Akerberg, H. Svensson, B. Schulz, and P. Nugues. Carsim: an automatic 3d text-to-scene conversion system applied to road accident reports. In EACL, 2003

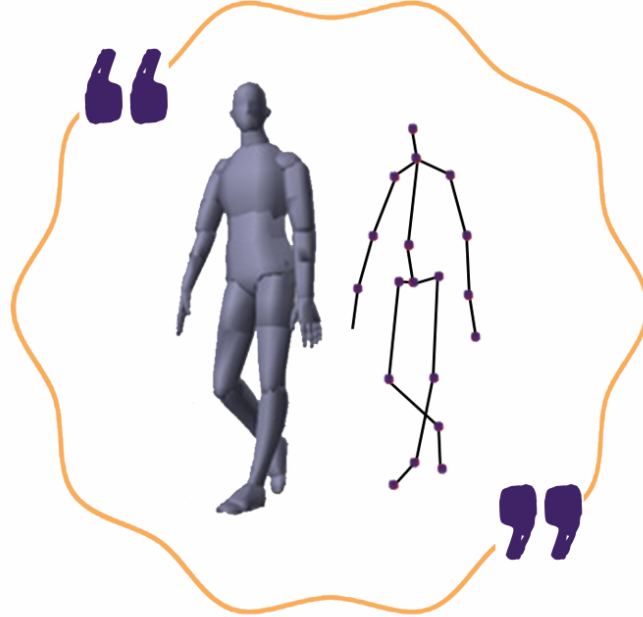
الملخص العربي

أصبحت مقاطع الفيديو المتحركة موضوع اهتمام لأنها أثبتت قدرتها الكبيرة على تبسيط الأفكار لأنها تقدم رسالتك بوضوح وتقدم متعة عظيمة وتوفر حرية إبداعية كاملة ومثالية لجميع قنوات التسويق وتساعد المتعلمين على الفهم بسهولة. انيمر يعطي الحياة لأفكارك ، ويأخذ وصفاً نصياً وينشئ فيديو رسوم متحركة ثلاثي الأبعاد للمشهد الموصوف. بعد ذلك ، تختار النماذج ثلاثية الأبعاد المناسبة وتطبق الإجراءات المذكورة على الأوصاف وتجعلها في مخرج فيديو.

يستهدف منتجنا المتعلمين والمؤسسات التعليمية ومنشئي محتوى الفيديو وصانعي الإعلانات. انيمر هو تطبيق لسطح المكتب يصور وصفاً مكتوباً. يحلل النظام النص d على وجه التحديد باستخدام تقنيات معالجة اللغة الطبيعية ، ويختار النماذج المناسبة من مجموعة قياسية ، ويولد مشهداً بصرياً ثابتاً ويطبق أخيراً الحركة اللازمة باستخدام تقنيات التخطيط والفيزياء لإعداده للعرض على المستخدم.



جامعة القاهرة
كلية الهندسة
قسم هندسة الحاسبات



ANIMARE

انيمر

كتاب مشروع تخرج مقدم الى كلية الهندسة جامعة القاهرة
كجزء من متطلبات الحصول على درجة البكالوريوس في هندسة الحاسبات

تقديم

برلنتي كيرلس
ياسمين علاء

باهي علي
جهاد محسن

اشراف
أ.د. نيفين درويش

10 يوليؤ 2019