AI in games - Summary Ai for Game Developers

Database foundation (Turan University)

UiT

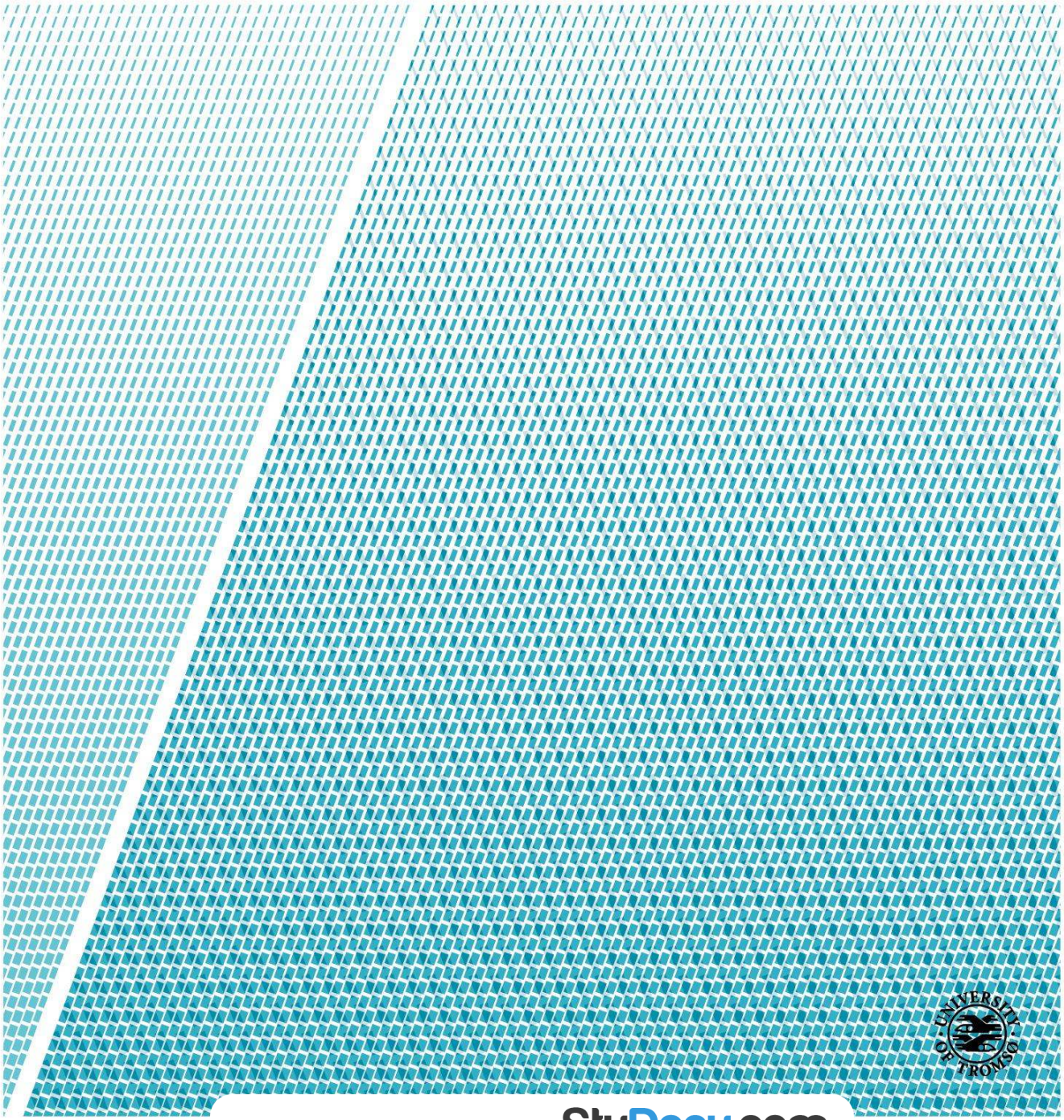
THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Engineering Science and Technology

Artificial Intelligence based narrative generation for games

Jørgen Elden Lindgren

Master's thesis in Computer Science June 2017



Title: Artificial Intelligence based narrative generation for games

Author: Jørgen Elden Lindgren

Date: June 13, 2017
Classification: Open
Pages: 45
Appendix Pages: 6
Attachments: Zip file

Department: Department of Computer Science and Computational Engineering

Study: Master of Science, Computer Science

Student number: 500047/jli042

Course code: SHO6264 Diploma Thesis - M-IT

Supervisor: Bernt A. Bremdal

Principal: UiT - The Arctic University of Norway (Campus Narvik)

Principal contact: Bernt A. Bremdal

Keywords: Augmented Transition Network, Ontology, Artificial Intelligence, Storytelling, Narrative

Abstract (English):This Master thesis presents an approach on using Artificial Intelligence to create stories for games. The Master thesis explores the posibility of using Augmented Transition Network and Ontology as a solution.

Abstract (Norwegian): Denne Masteroppgaven presenterer en måte å bruke kunstig intelligens til ålage historier for spill. Masteroppgaven utforsker muligheten til åbenytte Augmented Transition Network og Ontology som løsning.

Acknowledgment

I want to thank my supervisor Bernt A. Bremdal for giving me ideas, advice, help and valuable insight during this project, and for giving me the opportunity to work on a project about an a problem area that I wanted to work on. I also want to thank Tanita Fossli Brustad for helping me during the programming aspects of the thesis. And I want to thank my family and friends for supporting and helping me, in any way possible, throughout this Master's degree and Master thesis.

Jørgen Elden Lindgren
June 2017, Narvik

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The use of Artificial Intelligence (AI) inside computer games have been primarily limited to just allow for non-playable characters to select what types of movement to use, and to navigate the virtual environment. And although computer games may give the illusion of choices that have impact, they are in the end a predetermined choice with a predetermined end. Meaning that storytelling and narrative in computer games are predetermined, and that no amount of choices one has, the game will not feel alive. This master thesis aims to explore the potential of using AI for automated storytelling and automated generation of narrative in game development, and to conceptualize this through the development of a simple prototype.

1.1 Objectives of the Master thesis

The objective of this thesis is to explore the state of the art and the potential of automated storytelling and automated generation of narrative for game development. And to prove the potential of this through development of a simple prototype using text, images, or both. And use Machine learning where existing stories, narratives, photo sequences, etc. may constitute references for structures relevant for new narratives and provide the building blocks for such.

To support this task an investigating into different methods, that has been produced by researchers on the subject of storytelling and narrative generation, is required and a description of how said methods works. An important reference of for the proposed work will be the recent publications of Mark O. Riedl [1, 2, 3].

This investigation also includes literature research into the various AIs that may be considered a basis for this effort. This includes Neural Networks, Ontologies (Semantic Network), Augmented Transition Network, Case-based Reasoning and Analogical Reasoning (Analogy-based), Genetic Algorithms and others. Each of these AIs is to be explored and briefly described.

Based on the research a selection will be made about which method to go with, and the prototype will be created around this.

If possible, the recommendations that are put forward in the literature should be followed. It is preferable to start with an existing solution presented in the literature, as it will allow existing solution to be tested to see if it will work and modifications, with new ideas and preferences, can be made to the existing solution. If using an existing solution is not possible, then a prototype should

be built step-by-step using a scrum related approach, so as to control the use of time and resources while making sure that the prototype can always demonstrate functional capacities. This means that more functionality will be added when former functions with a higher priority have successfully been implemented.

1.2 Interpretation of objectives.

An analysis of the project text, reveals that, to start with the project is to first study the various papers that are researching applicable AIs, algorithms, and etc. that may be used in conjecture with storytelling and narrative generation for game development, and after that go through the various AIs and/or algorithms utilized in said methods.

Storytelling and narrative are two separate parts that will be described more later, but in short, storytelling is essentially the medium in which the story is told, for this thesis it is the prototype, narrative is a plot point which the player can interact with, which in turn can alter the course of the story in progress.

With regards to the sentence, explore the potential of automated storytelling and automated generation of narrative, part of the project text, the project indirectly tells that there should be minimal involvement by an author manually authoring new content. This does not necessarily mean that the author should not be responsible for establishing an initial state for the world, from which the AI used for generating stories/narratives and from this the world is able to be expanded upon by an AI.

1.3 Structure of thesis

Chapter 1: Introduction
Chapter 1 presents the introduction of the thesis, along with a description and interpretation of the objectives of the thesis.

Chapter 2: Previous work
Chapter 2 presents previous work, that are related or similar to the objectives of this thesis.

Chapter 3: Method of approach
Chapter 3 presents a description of various AIs, discussion of programming tools chosen, and creating a method to approach a solution to the objectives of this thesis.

Chapter 4: Results of approach
Chapter 4 presents the result gained from implementing the method from chapter 3, this includes the structure of the prototype, the parameters used to generate a sentence, and images of the output of the prototype.

Chapter 5: Discussion
Chapter 5 presents a discussion of the previous chapters, essentially summing up each chapter and then discussing the overall aspects of this thesis.

Chapter 6: Concluding remarks
Chapter 5 finishes up the thesis, by discussing future developments that could be applied to the prototype and giving a final conclusion to the thesis.

Chapter 2

Previous work

This chapter presents the research done on papers that is related to, or in some way similar to, the objectives in this thesis, as presented in 1.1 Objectives of the Master thesis.The papers are described in accordance to the authors description. However, the papers described here will not be a part of the conceptualized method during this thesis, the reason for this will be explained later. Although, some of the papers discussed in this chapter have the potential to be utilized in future developments.

The first part describes methods dealing with storytelling and the second part explains methods that deal with narratives. During the analysis, 1.2 Interpretation of objectives, storytelling and narrative has been described for the purpose of this thesis, although that same description may not be the same for the papers.

2.1 Storytelling

In a paper by M. O. Riedl[1], he claims that "Storytelling is an important part of how we, as humans, communicate, entertain, and teach each other.". The simple description of storytelling is that it is the telling of a story, with a structured narrative, to an audience. Stories has a beginning part, a middle part and an end part, that includes contents such as characters (protagonist, antagonist, supporting character, etc.), and an overall story that may be divided into several acts, arcs, or story plots. A story, except for experiencing it emotionally, can not be physically participated in or interacted with, by the audience, as it is told. The following descriptions deal with the various definitions of storytelling, story, stories or story generations presented in their respective paragraphs.

Paul Bailey describes in his paper, Searching for storiness: Story-generation from a readers perspective[4], an approach to automatic story-generation based on an intuitive model of responses from an imagined reader of the story. Generation assumes that there is access to reader's response, with which it then proceeds to heuristically search for story elements which can create story effects for the reader.

In the paper Story and Text Generation through Computational Analogy in the Riu system, by S. Ontañón and J. Zhu, they focuses on using story-generation with analogy-based process to generate both story and text using analogy. And they implement their technique in a system called Riu, which is a text-based interactive narrative system. Their paper has a more informative explanation of Riu [5]

Rafael Pérez y Pérez and Mike Sharples talks about, in their paper MEXICA: A computer model

of cognitive account of creative writing[6], that AI programs is being created for either the sole intention of completing an objective as a human would, or creating a program that will solve an objective unlike how a human would. They present that there is a middle ground between those positions.

In their paper, Generating story analogues[7], M. O. Riedl and Carlos Leon describes a computational system that, based on previous stories, generates story analogues. They do this taking existing stories or parts of stories and adapting them to new stories. By using an analogical reasoning algorithm to find if there are any similarities between the stories and they can create new contexts out of old stories

Theune et al.[8], introduces a multi-agent system framework for story creation in a program called The Virtual Storyteller. The agents in The Virtual Storyteller are responsible for creating three different parts of a story, one part deals with developments that happens in the fictional world, second part handles the portrayal of the plot of the story through a unique point of view, the third and last part deals with presenting the implementation of the story. There is also a main agent that controls the flow of the simulated virtual world, by choosing to combine sequence of events in the world that results in a story.

In his Ph.D. dissertation, Minstrel: A computer model of creativity and storytelling[9], Scott R. Turner introduces MINSTREL a storytelling system that creates stories by using large collections of word contexts already written by a human author. As a result of this the system is able to distinguish between a sword and a dragon, although it ends up with a dependence on a human author, which makes it so that the system can not really generate new information like a human is able to do. However this makes the system able to create a detailed story that is consistent, and able to use past events that has happened in story.

UNIVERSE is a program created by Michael Lebowitz with the purpose of running endlessly, like a soap-opera which Lebowitz writes was the inspiration for UNIVERSE. Using human authored plot points, where each plot point has a goal and the combined plot itself has a long-term goal, the different plots can be mixed and combined to create a plot continuously.

In the paper, Novice-friendly authoring of plan-based interactive storyboards[10], James Skorupski and Michael Mateas introduce a program called Story Canvas that generates interactive stories and uses similar techniques as UNIVERSE. The author is responsible setting the rules for plot generation as well the goals of the plots.

Porteous et al. in their paper[11] uses a technique to dynamically modify the story when a player takes an action that should change the story. In other words they present a technique that takes into account the possible detours a player might make. The system works by compartmentalize the original plot into smaller parts with constraints on the story and then build up the plot again. By doing this the story is more easily adapted to a player that takes a detour from the main plot.

EmoEmma is a prototype described by Cavazza et al.[12] with an interactive story, based on the novel Madam Bovary, where the player can verbally interact with characters that has the personalities has the same personalities from the book. The system allows the player to not only experience the original plot but also allows the player to engage multiple possible plots in the virtual world created by the author of the book.

M. O. Riedl and R. Michael Young presents in their paper, Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space[13], the Fabulist system. Fabulist merges both the author's goal and the characters goals to ensure that actions done by characters is understandable by readers of the story.

The paper by James R. Meehan, Tale-spin, an interactive program that writes story[14], presents a storytelling system called TALE-SPIN. The system gives the characters in the story goals and individuality, to create simple stories surrounding a character's aim to complete a goal.

GAME FORGE is different from the other story generation systems, so far, as it generate a entire fictional world based on a story that has been added. Presented by Harsook et al.[15] in their paper, GAME FORGE utilize a genetic algorithm to generate potential worlds which uses a story to set constraints that must be actualized in the finished world.

In the paper, Managing Interaction Between Users and Agents in a Multi-agent Storytelling Environment, Riedl et al.[16] describes an interactive story system called MIMESIS that uses techniques that adapts the the plot in order to balance a player's action against the predetermined plot. MIMESIS uses two methods to achieve that balance, one method happens the instance a player takes an action that might be a discourse and invalidates the current plot, the second method tries to envision what actions the player might do and tries to adapt the plot to make sure that the player can not do those actions.

SLANT is a system that integrates the programs, MEXICA, CURVESHIP and GRIOT, by Montfort et al.[17], in a blackboard system. where each program has access to the story and is allowed to make modifications to the story.

2.2 Narrative

Narrative is a sequence of events that has no standard form or structure, which means it does not truly have a beginning, middle or an end part, it just is. Yet in a narrative the listeners are allowed to become participants who can physically participate in the narrative. This means that it is a interactive narrative. The following descriptions deal with the various definitions of narratives presented in their respective paragraphs.

M. O. Riedl explains in his paper, Narrative Intelligence: A Human-Centered Goal for Artificial Intelligence, that "Narrative intelligence is the ability to craft, tell, understand and respond effectively to stories"[1]. He further explains that research into computational narrative intelligence could give AIs the ability to be able to act and or understand humans. Even mimic human behaviors like telling a narrative. Although there are many hurdles to overcome before computational narrative intelligence could become a reality.

In his paper, Interactive Narrative: A Novel Application of Artificial Intelligence for Computer Games, M. O. Riedl writes about Game AI. "Game Artificial Intelligence (Game AI) is a subdiscipline of AI and Machine Learning (ML) that explores the ways in which AI and ML can augment player experiences in computer games"[2]. To augment a player's experience, Game AI uses algorithmic techniques to give the appearance of intelligent behavior in non-player characters.

Riedl et al. presents a framework in their paper, Game AI as storytelling[3], that create interactive narrative. The framework uses an agent called experience manager that manipulates the virtual world to adapt dynamically the narrative content based on the player's actions or interferences on the narrative.

In M. O. Rield's Ph.D. Dissertation[18] examines the use of AI planning as a search-based technique to produce stories that exhibits strong plot coherence and strong character belivability. Riedl makes three central contributions to achieve this, first and extension to the AI planning search-based technique that identifies a character's intention, second implement a personality model for characters that can be integrated into AI planing, third an open-world AI planning algorithm that expands the ability of the normal AI planning algorithms to incorporate the same development, human authors do when dramatizing new work, in story creation.

SCHEHERAZADE, presented by Riedl et al. in their paper[19], is a system that tries to produce interactive narratives by use of crowd-sourcing. In their paper, Riedl describes how they got ananymous workers to submit linear stories about a bank robbery, which would then be combined to a single story.

Machado et al. uses Teatrix, an interactive narrative system[20] , to incorporate a template that is based on folktale functions. Teatrix has that both the author and player is responsible for creating content, the author manually creates the templates, and the player can select what will appear in the story.

In the paper Towards consistency in interactive storytelling: Tension arcs and dead-ends, by Leandro. M. Barros and Soraia. R. Musse, a prototype called Fabulator[21] is presented. Fabulator is designed for keeping tension arcs in interactive narratives and to stop any possible detour from the plot a player can do. To do this they use AI planning to alter the plot in a way that either prevents the deviation, or outright warn the player that they are moving to much outside the plot structure and that the player risks breaking the immersion by not being able to reach the end of the plot.

TeongJoo Ong and John J. Leggett explains the design of HEFTI[22] , a program that uses genetic algorithm to generate a story from a set of story templates that that is used to specify combinations that consists of assorted story elements at precis moment in the story. The story templates are further divided into sequences that characterize the relationships between the story elements, and by giving each sequence a specific time that they should occur, the sequences can then be combined dynamically to create the story as it is being told.

2.3 Miscellaneous

This category are for methods that are interesting but is not closely related as the examples above. In M. O. Riedl and B. Li paper, A phone that cures your flue: Generating imaginary gadgets in fictions with planning and analogies, researched using algorithm that can create fictional gadgets that is added in a story generator for plot generation. As they explain in their paper, most story generation lacks the capability to create new gadgets without the intervention of an author.

Assisted news reading with automated illustration, by Delgado et al., presents a text-to-image sys-

tem that take news stories as input then process it and select images that are most fitting for each line of text within the news story. The system can only find an image if said image is already in the system.

Schwarz et al. presents in their paper a text-to-video system. The system was tested with using news stories where it did its function well, but when used for selecting images for fictional stories it did not work. This is on account of news stories being differently structured than fictional stories.

Eduard Hovy writes in his paper, Generating Natural Language Under Pragmatic Constraints[23], that although much work remains in natural language generation concerning syntax, the primary hurdle that more or less blocks existing generators from effortlessly producing coherent paragraphs is a result of not understanding text planning completely. He says to make up for this lack of understanding, a new view on how text generation should operate by seeing it as a planning task.

CAB, by Levi B. Larkey and Bradley C. Love, is a system that imitates the way humans make informative comparisons through a simple iterative computation that compares elements from different representation and matches them.

LAS is a language learning program, by John R. Anderson[24], that uses ATN at it's core. LAS has three functions, UNDERSTAND, SPEAK and LEARN. The function UNDERSTAND takes strings of words as input and treats it as a sentence, LAS then uses information it has in it's network to have the sentence parsed and a representation of the sentence's meaning is found. SPEAK generates a sentence from the information in LAS's network as output. And the function LEARN is as it sounds, it takes an input that is made up of sentences and the representations of the sentence's meaning.

In his paper, Force dynamics in language and cognition[25], Leonard Talmy talks about a semantic category called Force dynamics, which is about how the interaction between items happen with regards to force. This is about exertion of force, resistance to force and overcome said force, simply meaning a representation of force would be for example when a person tries to open a door that can not be open because it is locked.

Chapter 3

Method of approach

This chapter is presented in two parts. The first part is a mostly brief description of the different AIs mentioned in the 1.1 Objectives of the Master thesis. This also includes AIs examined during the research, which are the Blackboard System, Intelligent Agents, Multi-Agent System and Automated Planning and Scheduling. The second part is the combination of the AI chosen and the method that will be used.

3.1 Artificial Intelligence Methods

3.1.1 Artificial Neural Network

Artificial Neural Network, or Neural Network (NN) as it is most known by, is an attempt to mimic how the biological brain works. NN is network of one or more artificial neurons that can be trained to solve different problems.

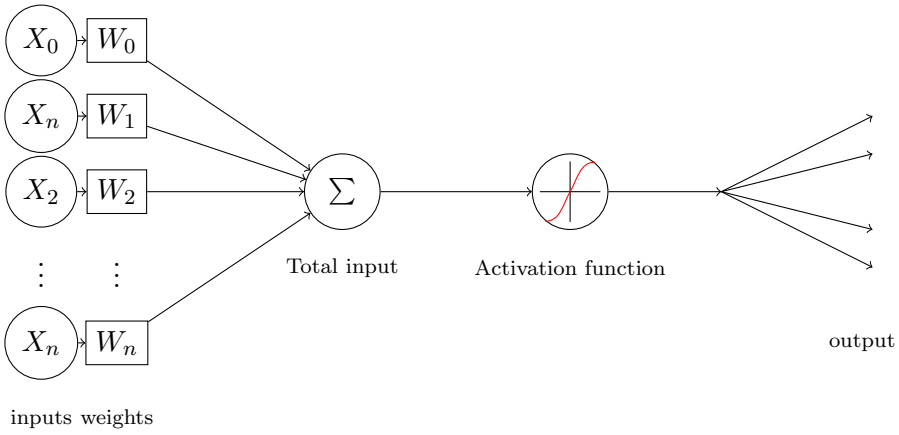


Figure 3.1: Artificial Neuron

Figure 3.1 illustrates an artificial neuron which is just a small part of a neural network. The neuron receives input from other neurons, and each input is given a weight

3.1.2 Ontologies (Semantic network)

Ontology and Semantic Network are two different concepts in Knowledge Representation. Knowledge Representation is a field in AI that is deals with the representation of information for computer systems.
An Ontology is vocabulary representation of information which are a set of concepts within a domain and the relationships between those concepts.
A Semantic Network is a graphic notation representation of information. Through the use of nodes and arcs the Semantic Network can describe the relationships between a number of objects.
In the end, Ontology and Semantic Network is equivalent in some sense, as every semantic network could be presented by an Ontology and every Ontology could be presented by a Semantic Network. The main difference between them is that one is a graphical presentation and the other one is a verbal presentation. Ontology and Semantic Network will be used interchangeably during this report.

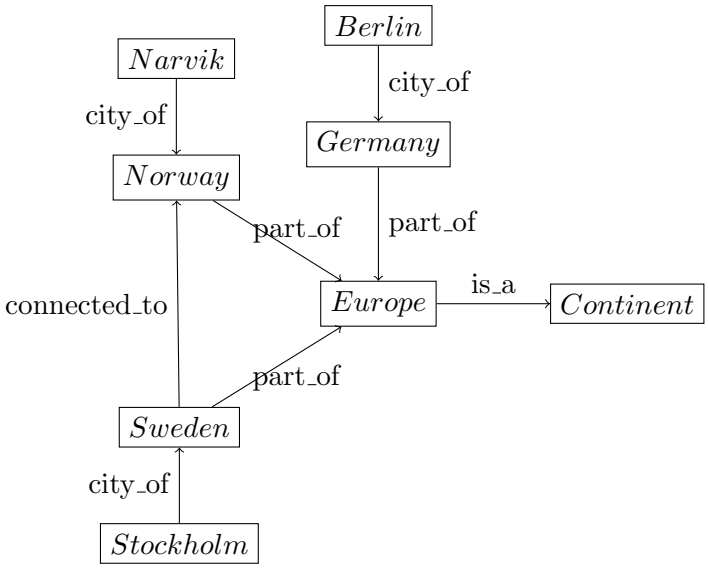


Figure 3.2: Simple Semantic Network

Figure 3.2 is a representation of a simple semantic network. The rectangles in the network represents objects and the arcs between the rectangles represents relationships. The Semantic Network shows that Narvik is a city of Norway which is a part of Europe that is a continent.

3.1.3 Augmented Transition Network

First described by W. A. Woods in his paper[26], Augmented Transition Network (ATN) is a method used for parsing and analyzing text of various length. The method is built upon the method of finite state machines (FSM), which is a directed graph, and was developed using the LISP language[27]. A directed graph is a mathematical structure consisting of a set of vertices which are connected by edges with an associated direction. A FSM uses nodes and arcs. Each node in the graph has an assigned arc, and each arc has a function. When a FSM starts it can go on until it has been stopped. ATN is just a advanced form of FSM.

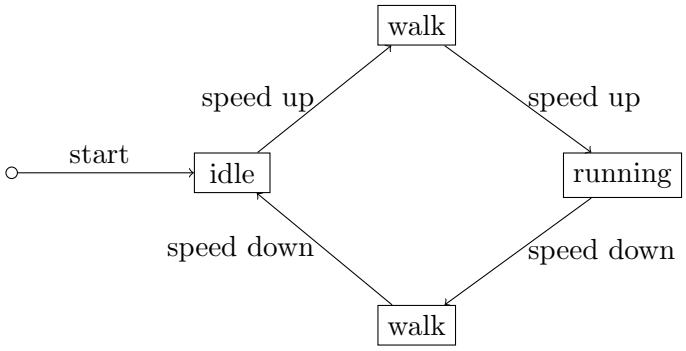


Figure 3.3: Simple Finite State Machine

Figure 3.3 shows a simple example of a FSM most common in computer games, a player movement. When the FSM begin it stays in the idle state until a player starts moving, then it continues from the idle state to the walking state. After which if the speed is increased it continues from the walking state to the running state, and if the player decreases the speed it goes from the running state to the walking state to the idle state. Each rectangle in FSM is a node which symbolizes its current state, and each arc is the transition from current state to next state. In ATN this translates to:

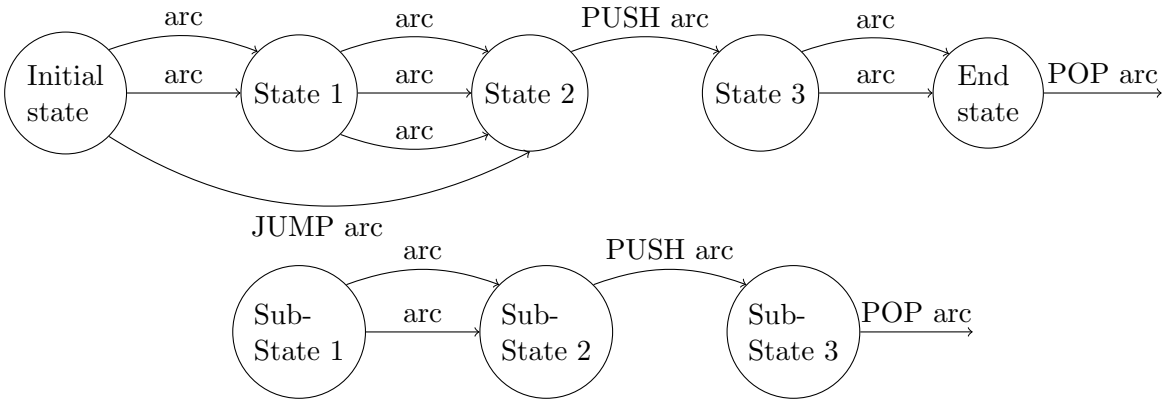


Figure 3.4: Augmented Transition Network

In Figure 3.4, the ATN starts at the Initial state and ends at the End state, there can be many states between the start and the end. A state can only be an End state if it is directly followed by a POP arc. An arc in ATN is defined by several elements that give the arc its common structure. The elements are as followed

- arc name
- arc classification
- arc test
- arc action
- arc register
- terminal action

arc classification, or type, is based upon the arcs name, for example the JUMP arc allows for bypassing states. arc test are arbitrarily created, they are mostly defined with regards to the type of arc. arc action is only done when the test is successful, and the action performed is specific to the arc type. arc register stores the actions done during the run time. and terminal action is where the arc stops its traversal.

First, what type of arc it is, second, and based on that type what it's main function is. Third, an arbitrary test that is created based on what that arc needs to have tested, fourth, an action that is done only when the test is true and if it is false a new arc is selected. Fifth, after the action has been completed it is stored in a register for later access, and sixth, a terminal action which indicates the next state.

As a result of these elements, there can be a wide variety of created arcs, and not all elements need to be inn an arc such as the POP and JUMP arcs, however there are three primary arcs that are in use. In Figure 3.4 they are, PUSH, POP and JUMP. When a PUSH arc is taken the ATN goes down to a lower level in the ATN structure, as shown in the figure by the sub-states, and will not continue onward on the upper level before the lower level is done. JUMP bypasses a state and ignores the passed state's arcs. The POP arc marks the end of the ATN parsing, if a POP arc is called on a lower level in the ATN structure, as displayed in Figure 3.4, then the POP arc signifies the end of the lower level and goes back up to the upper level to continue.

3.1.4 Analogy-based and Case-based Reasoning

In the paper, by Michael Gr. Voskoglou and Abdel-Badeeh M. Salem[28], they describe analogy-based(ABR) and case-based reasoning(CBR) to be two sides of the same coin.
ABR is described as the process of solving problems based on the solutions of similar problems from the past. However, this can prove to be difficult to implement for problem solving, as a result of the requirement of the the solver to obtain information, that could be solutions for other problems instead of the given problem. In short this means, that ABR tries to invent the wheel again by using information from other places because it does not remember that it exist from before. On the other hand, CBR tries to solve a given problem based on the information it has previously acquired. This means that, when comparing ABR and CBR, CBR can access it's own data banks for the information need to solve the given problem. This means that CBR does not try to invent the wheel again, as it knows it already exist from it's memory and can use it again.

3.1.5 Genetic Algorithms

Genetic Algorithm (GA) is a collection of algorithms and techniques that can be used to solve a variety of problems. GA is an adaptive heuristic search algorithm based on the evolutionary ideas

of natural selection and genetics, as such, this means that when presented with a problem GA use it's algorithms and techniques create multiple solutions to the given problem, and then the best solution is the one chosen.

To do this GA has a simple process, which is in four parts. First, a pool of random potential solutions is created that serves as the first generation. For the best results, this pool should have adequate diversity (filled with members that differ more than they are similar). Second, the fitness of each member is computed. The fitness here is a measure of how well each potential solution solves the problem at hand. the higher the fitness, the better the solution in relation to others. Third, members of the population goes through a selection process based on the fitness of each individual. The most fit members are chosen for crossover. The fourth and last step, features of the parents are recombined to form a new individual. Mutation may also be performed to avoid local optimum.

After all the steps have been performed, the termination process is next. If the population convergence to a single solution than the algorithm is terminated.
The following figure, 3.5, illustrates these steps.

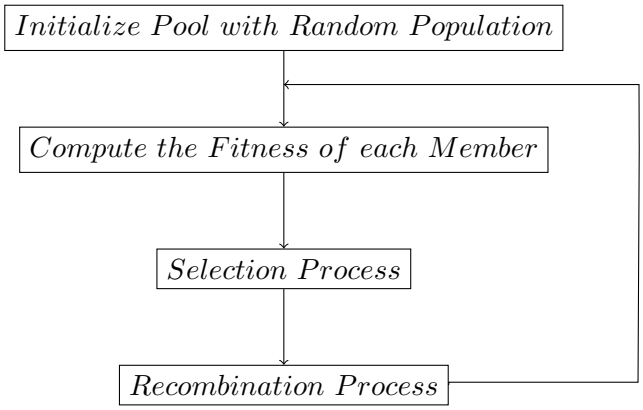


Figure 3.5: Genetic Algorithm

3.1.6 Blackboard System

The blackboard architecture, or blackboard system, is an approach to combine several different working systems together to solve a given problem, that they could not do on their own. The SLANT system mentioned in 2.1 Storytelling is an example of the blackboard system in use.

3.1.7 Intelligent Agent

An Intelligent Agent, from the perspective of an AI, is an autonomous entity that exists in an environment and acts in a rational way. what is rational is dependent on the environment. For example, is the agent attempting to solve a problem, or protect itself against other entities?
An Agent is generally made up of a number of different elements. these include one or more sensors that are used to perceive the environment, one or more effectors that manipulate the environment, and a control system. the control system provides a mapping from sensors to effectors, and provides intelligent (or rational) behavior. This anatomy can be applied to humans, as a first example. Sensors is essentially just sensations, like seeing, feeling, hearing etc., Effectors deals with motor control, like moving legs, arms, hands, etc., and the Control System is basically the brain and central nervous system.

What makes a program an agent is if it takes initiative when it is appropriate, when it maintains an agenda of goals that it pursues until the goal is either achieved or believed to be completed, and when the agent can delegate an action to another agent. Agents are also aware of its surrounding environment, and able to cooperate with other agents, software or human, to complete its task. The ability to communicate with other agents, software or human is also present, likewise being able to adapt its behavior based on experience.

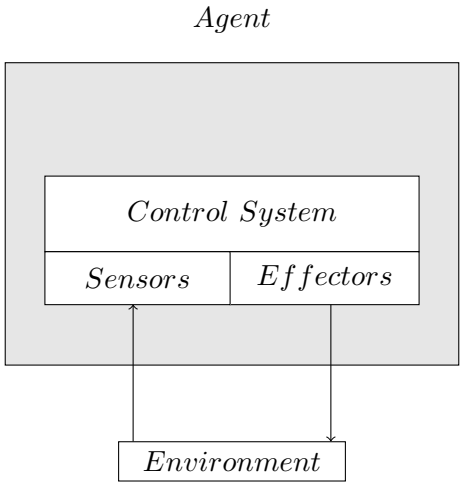


Figure 3.6: Intelligent Agent

3.1.8 Multi-Agent System

Multi-Agent System was briefly mentioned in section 3.1.7, but here it will be described. A Multi-Agent System are systems composed of multiple interacting agents. The agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action - of deciding for themselves what they need to do in order to satisfy their design

objectives. Second, they are capable of interacting with other agents - not simply exchanging data, but by engaging in analogues of the kind social activity that all humans engage in every day of human lives: cooperation, coordination, negotiation, and the like. Multi-Agent Systems (MAS) seem to be a natural metaphor for understanding and building a wide range of what humans might crudely call artificial social systems. the ideas of MAS are not tied to a single application domain, but, like objects before them, seem to find currency in a host of different application domains.

3.1.9 Automated Planning and Scheduling

Automated Planning and Scheduling, or AI Planning as it is most known by, is branch of AI that deals with the realization of strategies or action sequences, typically executed by intelligent agents, autonomous robots and unmanned vehicle. The AI planing system is charged with generating a plan that is one possible solution to a specified problem, the plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

This means that AI planning is simply, given a description of the possible initial state of the world, which could be one or more states, a description of the desired goals of each state, a goal state, and a description of a set of possible actions to, from any initial state, find a plan that is guaranteed to generate a sequence of actions that leads to one of the goal states. An action in AI planning may be given a score, a time limit, you could choose multiple actions or be limited to one action, some of the actions could be unlocked as a result of previous action or they could remain locked for the reason that another action was chosen.

A Simple example of this, with one initial state and one goal state. "You are hungry", is the initial state, and "You are not hungry", is the goal state. To achieve the goal state, a set of actions, from the initial state, is required to be taken so as to proceed from the initial state to the goal state. Some of the possible actions could be, "Go to fridge", "Choose food", "Choose healthy food", "Choose unhealthy food", "Prepare food" and "Eat food", etc., there can be more than just these a few actions, however for the purposes of this example these actions is all that is required.

3.2 Programming language of choice

There are many programming languages that can be used for programming and many programming tools to develop said program in. There are four programming languages that has been considered on the account of past experiences, C++, C#, Java and Python, and four programming tools has been considered as well, based on past experiences, Eclipse, Microsoft visual studio, Qt Creator, and NetBeans. After considering the various benefits each language and tool brings to the table, Microsoft Visual Studio and C++ was both chosen for a couple of reasons. First, Microsoft Visual Studio was chosen as a result of greater familiarity and experience with it. Second, C++ was chosen due to it being faster then the other programming language for the simple reason that it runs closer to the hardware. Advantages of this includes, running directly on the computer processor and having greater control over memory allocation. Another reason for choosing C++ was the more extensive familiarity and knowledge of the programming language.

3.3 AI method of choice

In theory, all methods discussed in chapter 2, have the capacity to be utilized for storytelling and generating narratives. And through, researching the various methods discussed and examining the various AIs, a choice was finally reached. The ATN and the ontology methods was chosen along with some of the ATN methods, one of which includes ontology. The reason for this choice is a result of considering the objectives of the thesis, which is storytelling and generation of narrative. And out of all the methods presented in Chapter 2 and the AIs presented in Chapter 3.1, ATN, including ontology, shows the most potential.

3.4 Proposing a method for automated storytelling and automated narrative generation

As mentioned earlier, ATN has been chosen as the AI method of choice. ATN is a method mainly used for parsing and analyzing text. However, through investigating the papers by S. C. Shapiro[29] and by P. L. Miller and G. D. Rennels[30], ATN is shown to demonstrate the possibility of using generating texts of various length.

3.4.1 ATN Methods

Shapiro presents a way to use ATN and SN for generating sentences and also parsing sentences. In his paper he writes that not much research has been done with regards as using ATN for syntax generation, and that the few research that has tried to use ATN for syntax generation but they did not have the same semantics as ATNs used for parsing or they had to use an external mechanism for determining the syntactic structure of the sentences to be generated. In his paper he shows an example of generalizing the ATN formalism that makes it so that a semantic network of words can be parsed and a sentence would be generated. This is done by combining ATN parsing and ATN generating to parse sentences, build and checks with if there are any previous knowledge of the sentence in the SN and then it creates a sentence in response.

In Miller and Rennels paper, Prose generation from expert systems: An applied computational linguistics approach, they explain the PROSENET/TEXTNET approach. PROSENET/TEXTNET uses ATN for generating the structure of prose with which its structure may vary depending on its content. The generated prose is put together of several prose fragments that can be all from a single word to a whole sentence or even a paragraph, this makes it so that the prose will have greater adaptability to variations or intricacy of the current material that is discussed. The approach is comprised of three components that is used to give prose output from an expert system. First, two ATNs called sentence-level and phrase-level is used to structure their respective prose expression, second is a expressive frame that is used to structure paragraph-level content, the third and last is paragraph-level ATNs that is used to structure paragraph-level prose expression. An expressive frame is the culmination of the different prose that has been generated.

The methods as presented by Shapiro and, Miller and Rennels, is developed by using the Lisp programming language. However, as previously discussed, C++ will be the programming language of choice during the development of the prototype.

3.4.2 A conceptual architecture for storytelling and narrative generation

Based upon the information acquired from Shapiro and, Miller and Rennels, a structural concept was visualized.

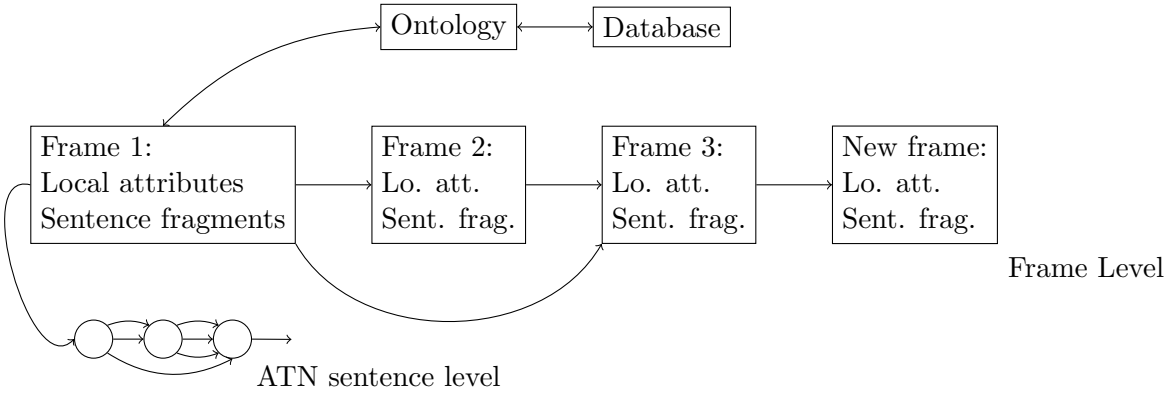


Figure 3.7: Simple example of the overall network for the proposed method

Figure 3.7 shows the structure of the prototype, this is called the frame level. The database consists of generated frames, the ontology keeps track of the relations between the various frames.

Figure 3.8: Simple ontology example

Figure 3.8 shows how the ontology could look like.

And when ATN, for the frame level, generates new frames it access its previous stored knowledge of each frame and their corresponding relation. This means, that at the start, an author will be necessary to create a set of story elements from which the ATN can draw its data from. Thus, frame level represents a type of narrative/story and the ontology explains the connection and content in the narrative/story.

In a frame, local attributes are essentially characterization and sentence fragments are basically sentence associated with that characterization. So, for example a local attribute could be that the frame is about a hero, and the sentence fragment could be the hero had said something or done something. In the end, these are just constructed sentences from the ATN sentence level, which will be described later.

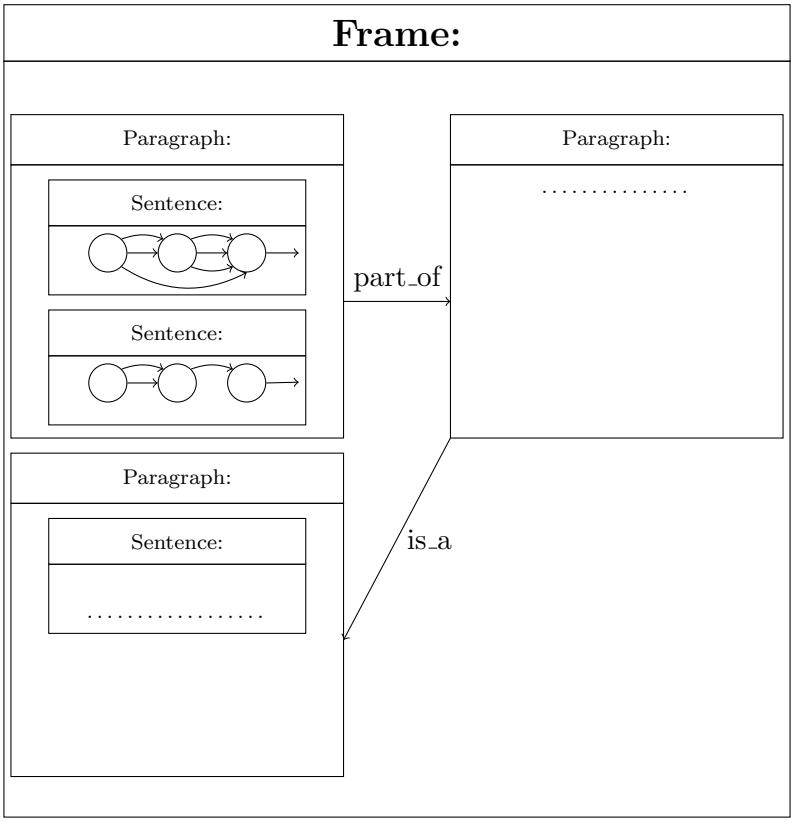
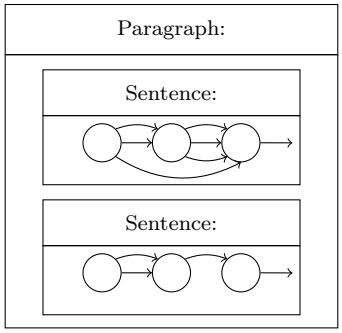


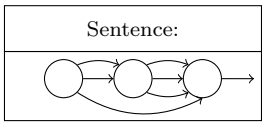
Figure 3.9: Frame structure

Figure 3.9 displays the contents of a frame. The frame can consist of one or more paragraphs and between each frame there is a standard relation, so that it is easier for the system to switch around on the relations between each paragraph. So that the system, is able to, change the overall structure of the frame to something else entirely.

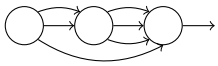
Figure 3.10a shows the paragraph which is inside a frame. The paragraph is populated by one or more sentences, and the sentences are dependent on what the contents of the previous sentence contains. This is to assure that the exact same sentence could never be recreated. Figure 3.10b displays sentence which is in a paragraph. The sentence is produced at the ATN sentence level.



(a) Paragraph structure



(b) Sentence structure



(c) ATN sentence level

Figure 3.10: Paragraph and Sentence structure, and ATN sentence level

Figure 3.10c shows a simple example of the ATN structure. And as briefly explained in 3.1.3 Augmented Transition Network, the ATN can have a variety of handmade arcs that completes its task dependent on what you want it to do. The ATN starts at intial state and from it, it passes through a list of arcs associated with the current state and then randomly selects an arc. it does this by generating a random number, using the *rand()*-function from the standard libraries in C++. The *rand()*-function normally generates a number in the range $[0, 1)$. And to get this value to be in the range of the number of arcs, the %-operator has been used. By choosing an arc, $A \in [A_0, A_n]$, where n is the total number of arcs on the current selected state S, the algorithm starts to travers the ATN graph until a POP arc is called, which results in a completed sentence.

3.4.3 Algorithms

The following algorithms were reached by primarily studying the papers by Shapiro[29] and, Miller and Rennels[30]. The papers by Woods[26] and Bates[27] were used for the reason that they contain the most information the inner workings of ATN. These algorithms were adapted from LISP to C++, based on the information of the papers.

The major steps applied to the prototype, for generating text, are the following algorithms, ATN, String output, Randomly selected arc, Arc sorting and Arc test WRD. These algorithms were adapted for C++. As a result of this, when comparing the C++ algorithms to the algorithms in LISP, there is a clear difference between them, this is due to their widely different code structure

Algorithm 1 ATN

```
1: Initialize network and get the initial state from it.
2: From the initial state randomly select an arc.
3: Then add the initial state and the selected arc to the storage/path
4: while ( the pop arc has not been called
5:     if ( currently selected arc is pop)
6:         then the pop arc has been called
7:     end if
8:     if ( test the selected arc to see if its test succeed)
9:         if the test is successful, then the ATN traverses the network from the initial state
10:        to the next state. There it randomly selects a new arc, and adds it to storage/path.
11:    else
12:        if the test fails then a new arc on the current state has to be selected.
13:        for ( all arcs in the current state)
14:            if ( any arc that is not false)
15:                then state has remaining arcs that can be chosen.
16:            end if
17:        end for
18:        if ( state has remaining arcs then)
19:            a new arc is randomly selected.
20:        else
21:            if there no more remaining arcs on the current state, then that state is removed from
    storage
22:            the previous state is reselected, and then a new arc is randomly selected.
23:            if ( if the newly selected arc has the status true)
24:                then that arcs status will be set to false and
25:                a new arc is randomly selected.
26:            end if
27:        end if
28:    end if
29: end while
30: if ( the pop arc has been called)
31:    then the sentence is outputted.
32: end if
```

Algorithm 2 String output

```
initialize string result
for ( all states in storage)
    get the current state
    for ( all arcs in current state)
        if ( any arc has status true)
            add that arcs action to the string result so that
            it can be added to a vector of strings later
        end if
        clear the current arcs status
    end for
end for
then clear the storage/path
if ( vector of strings is empty)
    then add the result string from earlier to the vector of strings
end if
when the vector of strings is not empty then
a check must be done so as to make sure that there is no
sentences alike.
for ( all the sentences in vector of strings)
    if ( a sentence in the vector of strings is found to be equal to string result)
        then there is a sentence in the vector of strings that is equal to the string result the ATN
    just created
    end if
end for
if ( a sentence is not found to be equal to result)
    then result can be added to the vector of strings.
end if
```

Algorithm 3 Randomly selected arc

```
generate a random number by using the size of the vector and the rand function
then by using that number get an arc from the current state
if ( selected arc has no status)
    then arc is okay for use and is added to the path
else
    while ( the chosen arc has status false)
        pick a new arc until
        an arc with status on hold is picked
    end while
    if ( an arc with status on hold is picked)
        for ( all arcs on the current state)
            if ( if any arc has status true)
                then set that status to false, as the only reason an arc with status on hold is selected is
                when arc's test fails as the result of a previous arc's action
            end if
        end for
    else
        if an arc with status true is selected instead of an arc with status on hold
        then that is set to false in the first algorithm
    end if
end if
```

Algorithm 4 Arc sorting

```
after receiving input from the ATN Algorithm
check what type the arc is.
if ( current arc's type is POP)
    then POP algorithm is used
end if
if ( current arc's type is PUSH)
    then PUSH algorithm is used
end if
if ( current arc's type is JUMP)
    then JUMP algorithm is used
end if
if ( current arc's type is WRD)
    then WRD algorithm is used
end if
if ( algorithm has given current arc status true)
    then the arc's test is true
else
    then the arc's test is fails
end if
```

Algorithm 5 Arc Test: WRD part 1

```
after receiving input from arc sorting algorithm
the input is separated into several parts
if ( if the current state is the initial state and if arcs on the initial state has no status)
    for ( all arcs in the state)
        give them status on hold
    end for
    then give the current selected arc status true.
end if
if ( current selected arc has no status)
    initialize an int, i, that will be used to limit the first while loop
    also set a boolean value for arc status to true
    while ( storage is greater then i and boolean arc status is true)
        initialize an int, j, that will be used to limit the second while loop
        while ( storage is greater then j and boolean arc status is true)
            if ( any arc in the vector of arcs has the value true)
                for ( all states in storage)
                    get the arcs of the current state
                    for ( all arcs in state)
                        if ( any arcs in state has the status true)
                            if ( the current arc's action is equal to the arc in state's action)
                                then that means that current arc's status is false
                                as a result there is not supposed to be two words after each other
                            end if
                        end if
                    end for
                end for
            end if
            int j is increase by one
        end while
        int i is increase by one
    end while
    if ( if bool arc status is true)
        for ( all arcs in the current state)
            they are given status on hold
        end for
        the current selected arc is given the status true
    else
        if the bool arc status turns out to be false then
        for ( all arcs in the current state)
            is given the status on hold
        end for
        the current selected arc is given the status false
    end if
end if
```

Algorithm 6 Arc Test: WRD part 2

```
1: if ( current selected arc has status on hold)
2:     initialize an int, i, that will be used to limit the first while loop
3:     also set a boolean value for arc status to true
4:     while ( storage is greater then i and boolean arc status is true)
5:         initialize an int, j, that will be used to limit the second while loop
6:         while ( storage is greater then j and boolean arc status is true)
7:             if ( any arc in the vector of arcs has the value true)
8:                 for ( all states in storage)
9:                     get the arcs of the current state
10:                    for ( all arcs in state)
11:                        if ( any arcs in state has the status true)
12:                            if ( the current arc's action is equal to the arc in state's action)
13:                                then that means that current arc's status is false
14:                                as a result there is not supposed to be two words after each other
15:                            end if
16:                        end if
17:                    end for
18:                end for
19:            end if
20:            int j is increase by one
21:        end while
22:        int i is increase by one
23:    end while
24:    if ( if bool arc status is true)
25:        for ( all arcs in the current state)
26:            if ( any arc has status false then their status is not changed)
27:                if an arc do not have a status false then it is given the status on hold.
28:            end if
29:        end for
30:        the current selected arc is given the status true
31:    else
32:        if the bool arc status turns out to be false then
33:        for ( all arcs in the current state)
34:            if ( any arc has status false then their status is not changed)
35:                if an arc do not have a status false then it is given the status on hold.
36:            end if
37:        end for
38:        the current selected arc is given the status false
39:    end if
40: end if
41: if ( the current arc has the status true)
42:     then the test was successful
43: else
44:     the test was not successful
45: end if
```

Chapter 4

Results of approach

4.1 The structure of the prototype

The development of the proposed conceptual architecture produced the following structure...

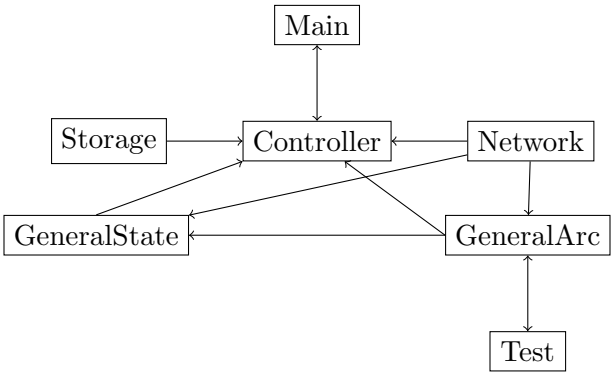


Figure 4.1: Simple UML diagram of the structure of the prototype

Figure 4.1 shows the resulting structure of implementing the conceptualized method...

4.1.1 Main

The Main class role is to is to run the ATN, the amount of times ATN can be run is decided in main. The reason for this is that, if the ATN is run only once then one sentence is created, however, if ATN is run more then once, more sentences are outputted. The output of the ATN is shown in a Console window.

4.1.2 Storage

The Storage class stores the ATNs traversed path through the network. The class has five methods, and only one of those five is important, the string output algorithm. The method passes through the path and check each state's arcs for the status true and the status false. It then prints out every action to an arc whose status is true. after every arc has had it's action added to the string it thereafter removes the status from said arc.

4.1.3 General state

The GeneralState recieves an argument that is used in order to delegate which of the states is the initial state. The class has four methods, first method sets a vector of arcs, essentially every arc added to a state is put in a vector, second method returns/allows the vector of arcs to be accessed from outside class, the third method allows other classes to aquire which state is the initial state, and fourth and last method is used to go through the vector of arcs and then randomly selects which arc to go through, algorithm 3 illustrates how the selection occurs.

4.1.4 General arc

The GeneralArc class takes takes in multiple arguments, an arcs current state, basically who owns that arc, what type of arc it is, what kind of action that arc has, and the state at which the arc ends at. If the arc is not between the initial state and another state, however, if it is between two intermediate states then it will get the argument previous state, and if the arc is at the end of the network it does not have the argument next state or an action as a result of this arc being the POP arc. The class contains eight methods which the other classes is able to access at any given time. Three of the methods relates to the states of which an arc has, previous, current and next. Two methods concerns what type of arc and what kind of action said arc has. Another two methods sets and gets the status of the arc. And the last method checks if the arc can be taken, by doing a test.

4.1.5 Network

The Network Class sets all the parameters in the network for the ATN to pass through. It sets a Boolean value, of true, on which state is the initial state, and the rest of the states get the Boolean value false. It sets how many states there are in the network and how many arc each state has. The class sets the parameters that is added to the vector of states. Each state might have an assigned arc with the following parameters, an arc's previous state, an arc's current state, an arc's type, an arc's action and an arc's next state.

4.1.6 Test

The Test Class verifies if a selected arc can be taken, by going through algorithms that check for all possibilities. The class has five methods, the first method is explained in the algorithm presented in Chapter 3, in short however, the method checks what the type of the selected arc is and then it is sorted into the algorithm that pertains said arc type. The second method sorts parts of the arc in to variables for easier access, it then checks if this selected arc belongs to the initial state, then it sets all other arcs status on the initial state to on hold and the picked arc's status to trues. If the arc selected does not belong to the initial state then it goes through another algorithm that checks if the selected arc has no status or has the status on hold, and dependent on that the part of the algorithm chosen might vary. The first part of the algorithm concerns if the selected arc has no status, where it then basically finds out if the selected arc is allowed to be passed through. If the selected arc do have a status then the second part of the algorithm is run, which is basically the same as the first part just with more restrictions.

4.1.7 Controller

The Controller Class is the start of everything, initialized in the Main Class, the Controller class initializes both the Network class and the Storage class in advance of starting the ATN. The class has two methods, the first method simply prints out the sentence constructed after going through

the network, the second method is essentially the ATN. The ATN starts by first procuring the initial state of the network and then receive a randomly selected arc from general state. It then adds the initial state to the storage for later use, afterwards it sends the selected arc and storage to the test class. It then receives input from the Test class when is done with the testing. After which if test is successful then it will select a new arc, if not then it will see if any of the arc on the current state has any available arc, if not then it returns to a previous state. If the current state do have an available arc it is chosen and the arc that had its test fail will be given the status false and the arc with the status on hold will be changed to true if its test is successful. This continues until a POP arc is called.

4.2 Parameters used in prototype

The tables that are presented her, has the following elements, previous state, current state, arc type, arc action, and next state. Each of these elements have been described earlier. The arc action parameters in each of the tables were arbitrarily selected, based on words and sentences usually found in a story setting.

Previous state:	Arc type:	Arc action	Next state
none	WRD	Once upon a time,	State 2
none	WRD	A long time ago,	State 2
none	WRD	There once was a time when	State 2
none	WRD	In a faraway place,	State 2

Table 4.1: State 1 parameters

Previous state:	Arc type:	Arc action	Next state
State 1	WRD	a great evil	State 3
State 1	WRD	a magical forest	State 3
State 1	WRD	a fantastical beast	State 3
State 1	WRD	a conflict	State 3
State 1	WRD	a magical artifact	State 3

Table 4.2: State 2 parameters

Previous state:	Arc type:	Arc action	Next state
State 2	WRD	appeared,	State 4
State 2	WRD	originated,	State 4
State 2	WRD	emerged,	State 4
State 2	WRD	transpired,	State 4

Table 4.3: State 3 parameters

Previous state:	Arc type:	Arc action	Next state
State 3	WRD	with great power to	State 5
State 3	WRD	arising out of	State 5
State 3	WRD	for the reason that	State 5
State 3	WRD	as a result of	State 5
State 3	WRD	a great evil	State 5

Table 4.4: State 4 parameters

Previous state:	Arc type:	Arc action	Next state
State 4	WRD	conquer the entire world	State 6
State 4	WRD	a magical artifact	State
State 4	WRD	monsters	State 6
State 4	WRD	as a result of	State 6
State 4	WRD	a conflict	State 6

Table 4.5: State 5 parameters

Previous state:	Arc type:	Arc action	Terminal action
State 5	WRD	with an iron fist,	State 7
State 5	WRD	hunted by	State 7
State 5	WRD	The creatures hunted	State 7
State 5	WRD	monsters	State 7

Table 4.6: State 6 parameters

Previous state:	Arc type:	Arc action	Terminal action
State 6	WRD	magical creatures	State 8
State 6	WRD	controlled by humans	State 8
State 6	WRD	everyone and everything	State 8
State 6	WRD	hunted by	State 8

Table 4.7: State 7 parameters

Previous state:	Arc type:	Arc action	Terminal action
State 7	WRD	was defeated	State 9
State 7	WRD	for food	State 9
State 7	WRD	to annihilation	State 9
State 7	WRD	heroes	State 9

Table 4.8: State 8 parameters

Previous state:	Arc type:
State 8	POP

Table 4.9: State 9 parameters

4.3 Prototype output

The results were produced using the parameters described earlier.The results obtained is gained through only the use of the WRD arc and the POP arc. The parameters on the arcs amount to a combination of 128 000 sentences the prototype can create. To see some of the different types of sentences generated, the ATN was run 20 times trying to create a sentence each time. As a result of this it produced different results every time.

Figure 4.2: Four parameters from state 1 included, first run

Figure 4.3: Four parameters from state 1 included, second run

Figures 4.2 and 4.3 uses four parameters from state 1, while from the other states only one parameter is used for each state. As a result of this the ATN can only create four unique sentence as shown in both the figures. Increasing the amount of parameters increase also the amount of sentences created.

Figure 4.4: Four parameters from state 1 and two parameters from state 2 included, first run

Figure 4.5: Four parameters from state 1 and two parameters from state 2 included, second run

Figures 4.4 and 4.5 uses four parameters from state 1 and two parameters from state 2, while from the the other states only one parameter is used. As a result of this the ATN can now create 8 unique sentences as shown in the figures. Increasing the amount of parameters in use of state 2 will create even more sentences.

Figure 4.6: Four parameters from state 1 and five parameters from state 2 included, first run

Figure 4.7: Four parameters from state 1 and five parameters from state 2 included, second run

Figures 4.6 and 4.7 uses four parameters from state 1 and five parameters from state 2. Now even more unique sentences are created. And now, when adding all the parameters on all the states the following is achieved.

Figure 4.8: all parameters from all states included, first run

Figure 4.9: all parameters from all states included, second run

Figures 4.8 and 4.9 shows when all the parameters on all the states are included. At this point Now even more unique sentences are created. And now, when adding all the parameters on all the states the following is achieved.

Chapter 5

Concluding remarks

As state in the introductory chapter, in 1.1 Objectives of the Master thesis and in 1.2 Interpretation of objectives, the main goal of the thesis is to explore the state of the art and the potential of automated storytelling and automated generation of narrative. And to prove the potential of this through the development of a simple prototype using text, images or both.

5.1 Discussion

As it stands now, the prototype is functional, it is able to able to generate sentences, based on words that have been manually added to the prototype. Through achieving this, the prototype is able to fulfill a small part of the objectives. As the objectives was to prove the potential through automated storytelling and automated generation of narrative the , through development of a simple prototype, using text, images or both. For now the prototype, fulfills the text requirement. The results from prototype is varied, the reason for this is that, each time the ATN is run, a random function is employed. And this allows the ATN to randomly select different arcs all the time. From testing, and changing the parameters, the prototype produces different types of results.

The prototype is based on two methods, as mentioned earlier, about ATN. The methods that have been used, are not modern to say the least. And the methods were developed in a LISP environment, which creates hurdles when trying to functions in a C++ environment. The hurdles give great difficulties, as a result of this, the potential to improve upon it great. This lead to a great deal of programming going towards adapting the old methods. So until new methods are created that deals with ATN, the only way to make it work is to designed it from scratch.

The prototype, as of now, have implemented sentence structure and ATN sentence level. And as it stands, the use of the prototype is narrow.

5.2 Recommendations for future work

As it stands now, there is room for a great deal of improvements that could eventually be added to the prototype. the code can be optimized to allow easier integration of new arcs. Semantic network can be added, frame can be added and paragraph can be add.
Some additions to the prototype in the future could be adding weights to arcs, so that when a character makes a choice they are more inclined to make the same choice again. Taking an example for The virtual Storyteller, agents could be used to create characters based on the resulting frames,

paragraphs, etc.

ATN could not only be used for creating stories and narratives, ATN could also be used to parse a sentence for a character and the generate a reply to the other character.

Creating a framework so that the game could be played.

As it is now, the prototype is just generating text. However, using the method by Delgado et al. images could be added based on the generated text, although using their method involves going unto the Internet to search for a picture that fits the generated text. Although, if Delgado et al. method were to be used with a Convolutional neural network (CNN), then based on the text generate by the ATN, the CNN could then generate an image based on that text.

5.3 Conclusion

To sum up this thesis, the objectives to solve was, explore the state of the art and potential of automated storytelling and automated generation of narratives for game development, and to develop a simple prototype to prove this potential.

The results that was achieved during the testing of the prototype, satisfy a part of it objectives, those being automated storytelling and text based prototype

The solution that was arrived at with the conceptualized method, although very little, there is a potential for an automated storytelling and an automated generation of narrative by the use of an augmented transition network and ontology.

Bibliography

[1] M. O. Riedl, "Computational narrative intelligence: A human-centered goal for artificial intelligence," *CHI'16 Workshop on Human-Centered Machine Learning*, 2016.

[2] M. O. Riedl, "Interactive narrative: A novel application of artificial intelligence for computer games," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pp. 2160–2165, AAAI Press, 2012.

[3] M. Riedl, D. Thue, and V. Bulitko, *Game AI as Storytelling*, pp. 125–150. New York, NY: Springer New York, 2011.

[4] P. Bailey, "Searching for storiness: Story-generation from a reader's perspective," 1999.

[5] S. Ontañón and J. Zhu, "Story and text generation through computational analogy in the riu system," in *AIIDE*, 2010.

[6] R. P. y Pérez and M. Sharples, "Mexica: A computer model of a cognitive account of creative writing," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 13, pp. 119–139, 2001.

[7] M. O. Riedl and C. Leon, "Generating story analogues," in *AIIDE*, 2009.

[8] M. Theune, S. Faas, A. Nijholt, and D. Heylen, "The virtual storyteller: Story creation by intelligent agents," in *Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment (TIDSE) Conference*, pp. 204–215, 2003.

[9] S. R. Turner, *Minstrel: a computer model of creativity and storytelling*. Ph.d. dissertation, University of California at Los Angeles, 1993.

[10] J. Skorupski and M. Mateas, "Novice-friendly authoring of plan-based interactive storyboards," in *International Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[11] J. Porteous, M. Cavazza, and F. Charles, "Applying planning to interactive storytelling: Narrative control using state constraints," *ACM Transactions on Intelligent Systems and Technology*, vol. 1, pp. 10:1–10:21, Dec. 2010.

[12] M. Cavazza, D. Pizzi, F. Charles, T. Vogt, and E. André, "Emotional input for character-based interactive storytelling," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, (Richland, SC), pp. 313–320, International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[13] M. O. Riedl and R. M. Young, "Story planning as exploratory creativity: Techniques for expanding the narrative search space," *New Generation Computing*, vol. 24, no. 3, pp. 303–323, 2006.

[14] J. R. Meehan, "Tale-spin, an interactive program that writes stories," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, (San Francisco, CA, USA), pp. 91–98, Morgan Kaufmann Publishers Inc., 1977.

[15] K. Hartsook, A. Zook, S. Das, and M. O. Riedl, "Toward supporting stories with procedurally generated game worlds," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pp. 297–304, Aug 2011.

[16] M. Riedl, C. J. Saretto, and R. M. Young, "Managing interaction between users and agents in a multi-agent storytelling environment," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, (New York, NY, USA), pp. 741–748, ACM, 2003.

[17] N. Montfort, R. P. y Pérez, D. F. Harrell, and A. Campana, "Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories," in *Proceedings of the Fourth International Conference on Computational Creativity*, 2013.

[18] M. O. Riedl, *Narrative Generation: Balancing Plot and Character*. Ph.d. dissertation, North Carolina State University, 2004.

[19] B. Li, S. Lee-Urban, and M. O. Riedl, "Crowdsourcing interactive fiction games," in *Proceedings of the 8th International Conference on the Foundations of Digital Games*, 2013.

[20] I. Machado, A. Paiva, and P. Brna, *Real Characters in Virtual Stories: Promoting Interactive Story-Creation Activities*, pp. 127–134. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[21] L. M. Barros and S. R. Musse, "Towards consistency in interactive storytelling: Tension arcs and dead-ends," *Comput. Entertain.*, vol. 6, pp. 43:1–43:17, Nov. 2008.

[22] T. Ong and J. J. Leggett, "A genetic algorithm approach to interactive narrative generation," in *Proceedings of the Fifteenth ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '04, (New York, NY, USA), pp. 181–182, ACM, 2004.

[23] E. Hovy, "Generating natural language under pragmatic constraints," *Journal of Pragmatics*, vol. 11, pp. 689–719, 1987.

[24] J. R. Anderson, "Induction of augmented transition networks," *Cognitive Science*, vol. 1, pp. 125–157, 1977.

[25] L. Talmy, "Force dynamics in language and cognition," *Cognitive Science*, vol. 12, pp. 49–100, 1988.

[26] W. A. Woods, "Transition network grammars for natural language analysis," *Commun. ACM*, vol. 13, pp. 591–606, Oct. 1970.

[27] M. Bates, "The theory and practice of augmented transition network grammars," in *Natural Language Communication with Computers*, (London, UK, UK), pp. 191–259, Springer-Verlag, 1978.

[28] M. G. Voskoglou and A. M. Salem, "Analogy-based and case-based reasoning: Two sides of the same coin," *CoRR*, vol. abs/1405.7567, 2014.

[29] S. C. Shapiro, "Generalized augmented transition network grammars for generation from semantic networks," *Comput. Linguist.*, vol. 8, pp. 12–25, Jan. 1982.

[30] P. L. Miller and G. D. Rennels, "Prose generation from expert systems: An applied computational linguistics approach," *AI Magazine*, vol. 9, pp. 37–44, Sept. 1988.

Appendices

Appendix A

Project description

UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT The Arctic University of Norway

Artificial Intelligence based narrative generation for games

Jørgen Elden Lindgren

Thesis for Master of Science in Computer Science

Problem description

This project aims to explore the state of the art and the potential of automated storytelling and automated generation of narrative for game development. The objective is to prove the potential through development of a simple prototype using text or images, or both. Machine learning where existing stories, narratives, photo sequences etc. may constitute references for structures relevant for new narratives and provide the building blocks for such.

The student will investigate different methods for supporting such tasks and describe how they work. An important reference for the proposed work will be the recent publications of Mark O. Riedl [1], [2],[3].

In his literature research the student will also address methods that can be considered a basis for this effort This includes neural networks, ontologies (semantic network), Augmented Transition Network (ATN) generator, case-based and analogical reasoning, genetic algorithms and others. Each should be explored and briefly described.

Based on this effort the student should make a selection and create his prototype. The prototype could consist of text only, images only or a combination of both.

It is important that the student follow the recommendations put forward in the literature. It is preferable that the student start with an existing solution presented in the literature, test this and make modifications according to his own ideas and preferences. If this is not possible the student should build a prototype step-by-step using a scrum related approach to control use of resource and time while making sure that the prototype can always demonstrate functional capacities. This means that more functionality will be added when former ones with a higher priority have successfully been completed.

The effort will be documented according to standards of reporting that the student is already familiar with. This will be followed by an executable program that must be first tested beyond the development environment prior to submitting the final thesis. In addition, the student should add the source code and a camcord that demonstrates the main functionality of the program.

Produce a video

1. Mark O. Riedl, 2016: Computational Narrative Intelligence: A Human-Centered Goal for Artificial Intelligence. CHI'16 Workshop on Human-Centered Machine Learning, May 8, 2016, San Jose, California, USA.

2. Mark O. Riedl, 2016: Interative Narrative: A Novel Application of Artficial Intelligence for Computer Games.

3. Mark O. Riedl,David Thue, Vadim Bulitkov, 2016, Game AI as Storytelling, School of Interactive Computing, Georgia Institute of Technology, Atlanta, Georgia, USA, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada.

Dates

Date of distributing the task:	09.01.2017
Date for submission (deadline):	06.06.2017

Contact information

Candidate	Jørgen Elden Lindgren jli042@post.uit.no
Adviser at UiT-IVT	Bernt Bremdal bernt.bremdal@uit.no

General information

This master thesis should include:

- Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis
- Selection & investigation of actual materials
- Development (creating a model or model concept)
- Experimental work (planned in the preliminary work/literature study part)
- Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 4 weeks. It should include bullet pints 1 and 2 in "The work shall include", and a plan of the progress. The preliminary study may be submitted as a separate report or "natural" incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgements, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software/codes or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

Submission requirements

This thesis should result in a final report with an electronic copy (i.e. CD/DVD, memory stick) of the report included appendices and necessary software codes, simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered in one signed loose leaf copy, together with three bound. If there is an external company that needs a copy of the thesis, the candidate must arrange this. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the "General guidelines for thesis" and the subject description for master thesis.

The final report with its appendices should be submitted no later than the decided final date. The final report should be delivered to the adviser at the office of the IVT Faculty at the UiT.

Appendix B

Source code

The source code is located at two places. A readme file is included at each site.

https://source.uit.no/jli042/Prototype

https://bitbucket.org/Tverren/prototype

Appendix C

Zip file

Name of the zip file: demo.zip