A Framework for Defining Intelligent Agents with Different Orientations that take Decisions based on a Planning Graph in a Game Environment

カン, ウメー アズファ

https://doi.org/10.15017/1500749

出版情報：九州大学, 2014, 博士（情報科学）, 課程博士
バージョン：

A Framework for Defining Intelligent Agents with Different Orientations that take Decisions based on a Planning Graph in a Game Environment

February 2015

Khan Umair Azfar

Abstract

This dissertation proposes a framework for the development of intelligent agents in Role-playing games where planning graphs are used to allocate actions to the agents and they choose to perform these actions based on their orientation. The abilities contained in them help in defining their orientation as well as their ability to perform actions during decision making activity.

The overall goal of this research is to generate a different animated story every time the system is run. By distributing a game environment into a collection of characters and items and using actions as the mode of interaction between the two, this random story generation is achievable. The physicality of the characters is dependent on the choices made by the users in previous runs of the system. The system adapts to these choices and generates a character based on them. This ensures that if this system is run at different areas around the world, it will start to create characters that are more according to the likes of the people in that part of the world. This required a system that could remember the users' choices and evolve with increased user input. For this reason two separate genetic algorithms were used to define the orientation of the characters through the allocation of abilities and for defining the physical characteristics. For allocating the right abilities in a character's chromosome a genetic algorithm is used to solve this knapsack problem. For deciding the physical characteristics of characters, genetic algorithm with sigma selection is used to narrow down the choice for the most likable physical appearance based on previous user choices.

A story is dependent on the actions done by the agents; if we string these actions together, we can generate a meaningful story. Actions are dependent on the goals that need to be achieved by the characters. These actions need to be in a certain order to allow efficient completion of goals, which means that every character must formulate a

plan before taking an action. This system analyzes the goals that need to be achieved by the characters and then creates a plan which elaborates all the actions and the order in which they should be performed to achieve those goals. The plan is composed of several pathways that can be followed to reach the agent's goals irrespective of whether all the actions can be performed or not. The agent first marks out all the actions that it can perform and then decides whether it is willing to do those actions at the given time. This decision making is based on the character's orientation as the character is most likely to select those actions which are according to its orientation; at the same time the character might choose the action which goes against its orientation if that is the best possible course of action at the given time. As a result, if a game has the same characters with the same goals they might achieve them differently during various playthroughs thus satisfying the overall outcome of the research to generate a different animated story every time the system was run.

Acknowledgments

I am using this opportunity to express my gratitude to my supervisor, Professor Yoshihiro Okada who supported me throughout my PhD. I am thankful for his aspiring guidance, invaluably constructive criticism and friendly advice during the course of my research. I am sincerely grateful to him for sharing his illuminating views that guided me in completing my PhD on time.

I would also like to thank the members of the Okada Laboratory who provided me with precious opinions and discussed various methods that helped me in overcoming the various challenges that I faced during my three years.

Special thanks to my wife, Humaira for standing by me through thick and thin and for helping me in providing motivation and support. She helped me in staying on track and keeping focus whenever I went astray.

Finally, I would like to thank my brothers and most importantly, my father for his renewed support and encouragement to pursue my studies abroad even when he needed me the most. This thesis is dedicated to him for believing in me.

Contents

Chapter 1

Introduction

Role-Playing Games (RPGs) are one of the most popular genres of games, which has been designed for or ported between every existing hardware platform produced for games. These games have their origins in tabletop and live-action variations which have been successfully translated to gaming devices. RPGs formed one of the primary sources of inspiration for the early evolution of computer games, and still retain a strong influence towards research and evolution of computer games [31]. RPGs have been the subject of interest from a variety of viewpoints in the game studies community, including design, development, culture, sociology, psychology and interactive storytelling [12], [16], [34], [36], [9]. Similarly, the operation of Non-Playable Characters (NPCs) or agents in these games is of interest in agent-based systems [35], [37]; and the unique nature of RPGs make them an obvious source of study for investigating their interaction with the player and the game itself, with the results applicable across the spectrum of game studies [50].

1.1 Background

Normally, all RPGs are story driven where different characters provide different tasks for the user to accomplish. The main plot of the story and the tasks that are to be accomplished are fixed which reduces the replay value of the game. A role-playing game

is an immensely popular genre as it gives the user the ability to interact with different characters and improve their own characters with the passage of time. The interaction between the characters is bound to create new stories and since the emphasis of the research has been digital story-telling, it makes sense to explore the various forms of RPGs that are present out there. The important categories are [23]:

- Pen-and-Paper/Table-top
- Systemless
- Live-Action Role-Playing
- Single Player Digital
- Massively Multi-Player Online
- Freeform
- Pervasive

Out of these categories, the one that has the closest relationship with digitized story-telling is the Pen and Paper RPG. Pen and Paper RPGs (PnPs) are complex games that involve multiple participants engaged in an activity that takes place in the real-life gaming situation and in the minds of the participants, simultaneously. One of the key aspects of these games is that they form very pure examples of interactive storytelling systems [4], which makes them great sources of inspiration for creating digital storytelling systems. PnPs generally contain an administrative function often referred to as Game Masters (GMs) [7], [53]. The GM is associated with a range of functions, with however substantial variance in the specific responsibilities. In PnPs, the GM normally plays a central role as story facilitator [4], [10] by managing the overall plot of the game story,

and controlling the behavior of any game world entities and objects not controlled by the player characters [49]. There is a degree of administration required for a game to make sense and to produce a digital story. But how do we decide if this is the only way an RPG can be defined. This requires us to evaluate the above categories and come at a definition that best details the different aspects that an RPG can have. This can only be done by discussing the different features that these RPGs contain.

1.2 Features of an RPG

There is significant variation amongst RPGs, including the mechanisms supporting game play and the varied play styles that typify them. There are however considerable areas of similarity in these games that bring them under the umbrella of role playing.

1.2.1 Character

Characters are the primary (in most cases the sole) means by which the players can interact with the game world. The methods by which the characters are defined can be quantitative, qualitative and maybe a mixture of both, but in all cases the characters are regarded as individuals, with their own unique place in the game world.

The players are able to effect, and influence, the changes in the game world through actions expressed through their characters. The development of a character varies vary widely between games but is always subject to some player control. There might be skill points which the player chooses to allocate, specializations to select or decisions made about emotional changes. This separates RPGs from other games with character development, where that development is fixed and pre-decided by the game designers, with some limited choices given to the player within a defined framework.

1.2.2 Game Master

RPGs sometimes have participants who control the game world beyond the players characters. These participants are typically referred to as game masters as they have the overall control on the rules for the game. The exact duties of the game master vary, The viewpoint of a game master is very different than that of the normal players. While players are mainly concerned with their particular character, game is responsible for presenting the world to the players, elaborating story elements and actions that can be taken throughout the game. This is even more apparent in a digital role-playing game where the automated game engine carries over these tasks and acts as a game master. The game masters are responsible for the selection and positioning of props/items within the game as they play an important role in defining the game.

1.2.3 Game Space

Role-playing games consistently make use of a fictional game world but they are not unique in this. Many, if not most, games are set many RPGs are set in a fictional world but RPGs do it in a manner that is consistent and distinctive. Players in an RPG are generally free to choose their path through the world and also revisit areas. The Characters have choice as where they can visit, and in what order should they visit these areas.

1.2.4 Game World

Role-playing games differ in the scope of the configuration available to the player from other games. Digital games, even though limited by current technology, offer a wider range of possibilities of interaction which usually including combat, dialogue, object interaction, etc. These interactions are generally limited and different genres try to specialize in only one type of interaction. RPGs however provide more interaction opportunities, but they tend to be generalist rather than specialist in how they allow players to perform the

interaction. The aim of the RPG is to give the user as much liberty as possible and make every one of the player's interaction matter in the grand scheme of things.

1.2.5 Narrative

Role-Playing Games demonstrate more story-like elements than other genres. The narrative elements in role-playing game are a result of other elements such as actions done by the characters and behaves as a corollary and not a necessary element in itself. In essence,RPGs cause narratives to emerge on through character interaction and do not contain them as such. Players experience a sequence of related events which can be interpreted as a narrative, in much the same way that narratives are formed from real life experience. As a result, the emphasis in RPGs remains on the actions and the character interaction to generate narrative. There however have been RPGs with deep story elements as well, but they are generally added as cut scenes and do not influence the actual game play.

1.3 Defining the Framework

The goal of this framework is to provide the administrator the tools that would enable him/her to populate the game environment with characters and items and allocate their initial and goal states. The main idea is that, the administrator adds the characters/agents and items to the game and based on the initial states of the items and the goals that the characters want to achieve, the characters formulate a plan and start doing actions based on that plan. As a result, as soon as the game starts, all agents starts to perform their actions persistently, till the final objectives are achieved.

1.3.1 Problems to Solve

RPGs normally have NPCs that have very basic AI and only respond to interactions that the players have with them. This is done to keep the game focused on the players and avoids any extra processing that is required to perform the tasks done by the agents. This however causes the agents to not behave naturally as they would in real life. When creating the system there were many problems that cropped up, the most important of which are explained here:

Environmental Effects

The first problem was to determine how important a role should an environment play in the overall story. It has been seen that a story normally focuses on the characters and the same story can be told in different settings if the characters and their goals are kept the same. Therefore, the dependency of characters and items on the environment was removed as their impact on the story was deemed negligible. However this can be an interesting area of research in the long run where environmental effects will have an impact

Character Goals

The second problem to solve was to provide every NPC with goals and a framework by which they will try to achieve their goals. The goals of an NPC are related to items, which means that for every item, an NPC can have a set of goals associated. As a result, increasing the number of items in a game can effectively provide the NPC with different goals. The NPC should then formulate a plan by following which it can achieve the goals. As the number of states increases, using a Finite State Machine for such a task becomes overly complicated,therefore this has been solved by using a planning graph. A planning graph provides the NPC with an action order, by following which goals can be achieved.

This graph is created in real-time thus ensuring that if the state of an item is changed, the framework takes into account that change and creates a new graph. At the same time, the graph allows the character with different paths to follow in order to achieve the objectives. If there is a problem in following one of the given paths, the character chooses an alternate path to achieve its objectives. This is achieved by using the backtracking quality of the A* algorithm which helps in traversing all the given paths. A* is a path finding algorithm that is used here to find the validity of the path being taken in a planning graph. It finds the best cost effective solution to a given problem provided that the heuristic is admissible. At the same time the character just focuses on the actions that are relevant as according to the plan and does not need to concern itself with unnecessary actions that it might otherwise concern itself with in an FSM.

Character Orientation

While creating a character we needed a system that helps in defining the orientation of the character. The orientation was supposed to play a major role in deciding the character's behavior and in turn should also reflect on the decisions that the character will make in the long run. As a result, the actions that were performed by the character were used to define the orientation of the character. Since actions speak louder than words, therefore the number of good or bad actions contained in a character define its orientation towards good or bad. But all the available actions cannot be part of a character and at the same time an action can not be repeated. Therefore a special kind of Knapsack problem was solved by using a Genetic Algorithm. Every action was given an ID number and by adding the ID numbers of all the actions together, the sum must reach a specific value that defines the kind of character that the administrator wants to create. Since we did not know before hand how many actions are going to be part of the character, GA were used to find the solution of this particular Knapsack problem where an action cannot be

repeated. The solution to acquire was a specific sum as defined by the administrator by combining together various collection of actions that can be either good or bad. Genetic Algorithm was used as its speed is not much affected by the number of abilities that it needs to go through and at the same time it provides solutions by randomly joining abilities together, thus creating different actors.

Decision Making Process

When tasked with two different kind of actions that produce the same result, we wanted the character to perform according to its orientation. If two actions of the same orientation were encountered, then the lower costing action is selected among the two. But if two actions cost the same yet have opposite orientation, then the action that is according to the orientation of the character is more likely to be selected. This is because the number of like actions only increases the chance of that action to be selected for performance. The character can however choose not to perform that action and resort to doing action that is not according to its orientation. There is always a possibility of the character to select the action that is against the orientation of the character thus making the outcome of the story unpredictable. The possibility of a character to select an action as according to a certain orientation is dependent on the number of actions contained in the character of that same orientation. The more those actions are the better the chance it has to perform an action belonging to the same orientation. This however does not rule out the possibility for the character to behave unpredictably as there is always a chance for the character to pick an action that goes against its orientation.

Character's Physicality

We also wanted to impress upon the idea that the created character should reflect the choices made by the administrator when a character was previously being defined. The

system takes into account the kind of characters that were created previously and uses that information to breed a new character that closely resembles the previously created characters. This is achieved by using a genetic algorithm with sigma selection to approximate the physical characteristics of the character to its parents. Thus using the previously selected attributes, an attribute within the standard deviation range is picked for every part of the physical body of the character, thus ensuring that created character is physically closer to the parents. As a result, the system tries to give its users a feeling of belonging to the created characters so that they can relate to them better. Therefore when this system is used at different places throughout the world, it will most likely start to create characters that are according to the choices that were made in those areas.

1.4 Related Work

Upon analysis of the different features of RPGs, we come to the conclusion that the most important aspect of an RPG is the characters that interact with their environment in the game world. In a digital game, these characters can be referred to as agents and their behavior takes the center stage in terms of research. There has been quite a lot of research in agent behavior and also about using this behavior in games. We start off with the most related work in this direction which comes close to this research.

Lankoski and Bjork [32] present method where design choices regarding gameplay are identified. These choices are described as gameplay design patterns and related to how specific features in a character design can support gameplay. Along with the design patterns, the concepts of recognition, alliance, and alignment are used to introduced to the characters to affect the overall gameplay. The difference of this technique with the current framework is the creation of the design patterns that adhere to a certain situation. The administrator needs to decide before hand what situations will the NPC need to encounter

during the course of the game and provide the NPC with specific design patterns before hand. If the situation is unknown to the administrator, he would resort to providing all possible design patterns to NPCs to get a favorable outcome. Our system does not dabble in this uncertainty. An NPC is given a list of actions that it can perform and based on those actions, the NPC tries to achieve the objective. If the plan cannot be formulated then it does not attempt to follow any plan. As a result the following of any action is supported by achievable objectives and does not blindly push the character in a certain direction.

MacNamee [44] has proposed the use of a paradigm of virtual humans to show how a complex society of believable social agents can improve immersion in computer game environments. Using a model of situational intelligence all the agents adapt to the dynamic situation at a certain point in a story. However, this approach centers on creating a reactive environment for the characters and forces the system to create new paths at the spur of the moment. Such an approach might be useful when user interaction comes into play. If the user chooses to change the attributes of a character, the reaction might end up changing the entire story. This approach is dependent on the definition of the reactive mechanism of the agents. A change is brought into the immediate vicinity of the agent and based on the defined artificial intelligence of the agent, the agent responds to that change. Normally in games, the agents only react to the immediate vicinity of the user and do not otherwise perform actively. In this approach there is a persistent simulation of agents that keep performing their actions even when the user is not in their immediate vicinity.

This approach is by far the closest to our system, but there are differences that set it apart. Like our system, there is a persistent simulation of characters but they do not generate any plans to react to the changes in the environment. Their artificial intelligence

is predefined which might however reach the goals eventually but it does not try to achieve the best order of actions that can lead it to its goals.

The OPIATE system dynamically casts characters into eight of the nine available roles, with the ninth role taken up by the user [14]. The nine roles fall into the following categories as defined by Propp [45]: Hero, Villain, Mediator, Donor, Helper, False Hero, Princess, King and Family. Every plot is broken into several subplots and all of the roles are dynamically recast for the enactment of every subplot. This casting is done based on a certain set of criteria. This system aims to create a different story by dynamically casting the characters but it is limited to the number of characters int he story. The chances of expanding this system to facilitate more characters is limited and will not meet the demands of a game where hundreds of characters interact together and try to achieve their goals at the same time. Our system does not limit the number of characters, items or actions that can be defined in it. With a vast range of actions and items given to the characters, every character formulates a plan as to how it should achieve the objectives.

The current system was inspired by the dynamic nature of these researches and aims to produce characters and items dynamically. This dynamic nature also extends to the actions performed by the agents to produce an unpredictable and natural environment and a user's choices have an impact on the overall character development through the use of evolutionary algorithms.

1.5 Overview

This dissertation consists of seven chapters. Chapter 2 explains the initial idea and leads up to the research that was done for the eventual creation of the current system. Chapter 3 focuses on the character creation process where evolutionary algorithms are used do define the physical and mental characteristics of an agent. Chapter 4 explains

the concept of using planning graphs for devising a plan of action for the characters to reach their objectives. Chapter 5 gives details about the decision making process that enables the agents to exhibit a more natural response to the actions they are supposed to perform Chapter 6 explains the entire system in detail and combines all the research explained in the previous chapters. Finally, we conclude this dissertation in Chapter 7 by giving a summary of the research and propose a few applications where this research can be very useful.

Chapter 2

Conceptualizing Story Generation

Computer graphics have been used for creating real and imaginative worlds by artists and programmers in various mediums like games and commercial movies. Even though the impact of these computer generated worlds has been immense, but the techniques for generating the virtual content have never been easy and require artists who are adept at complicated professional tools. As a result, the common user and to some extent the programmers have remained detached from the area of creating 3D content. The basis for the idea discussed in this thesis lies in the initial research done for 3D content generation and manipulation[26]. The idea was built on the concept of using voice to generate 3D landscape and objects. The most common way of telling a story is to narrate it with your voice and the listener creates the picture in his mind about how the story came to pass. We wanted to generate the same picture on a computer screen thus allowing the listener a visual representation of the scene being created by the narrator. This enables the narrator to create 3D scenes easily without requiring any special skills[19]. We used the Microsoft's Kinect camera, which also has a very good microphone array for this purpose. Our system setup can best be explained by figure 2.1.

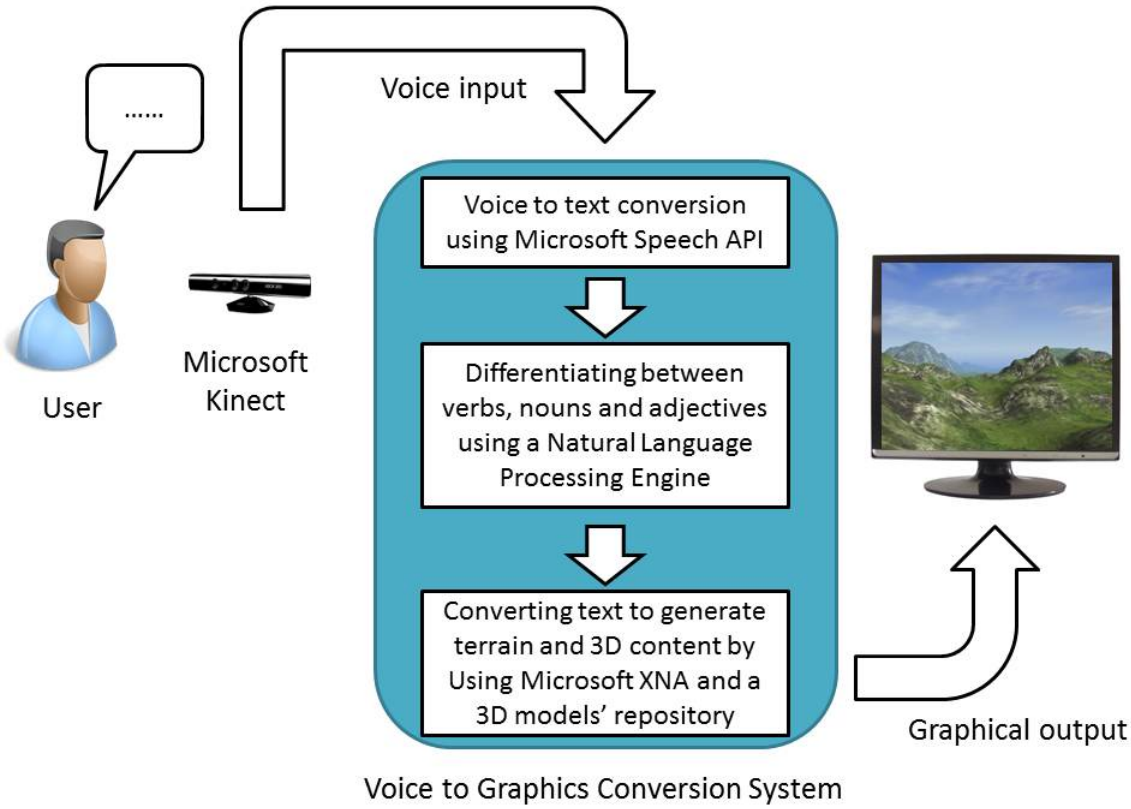

Figure 2.1: Terrain Generation System.

In the beginning, the user was given a plane terrain and a default skybox that can be manipulated by voice input as shown in figure 2.2. When the user spoke into the microphone, various terrain elements were generated using various mathematical algorithms in order to mimic the type of scene that the user was narrating as shown in figure 2.3.

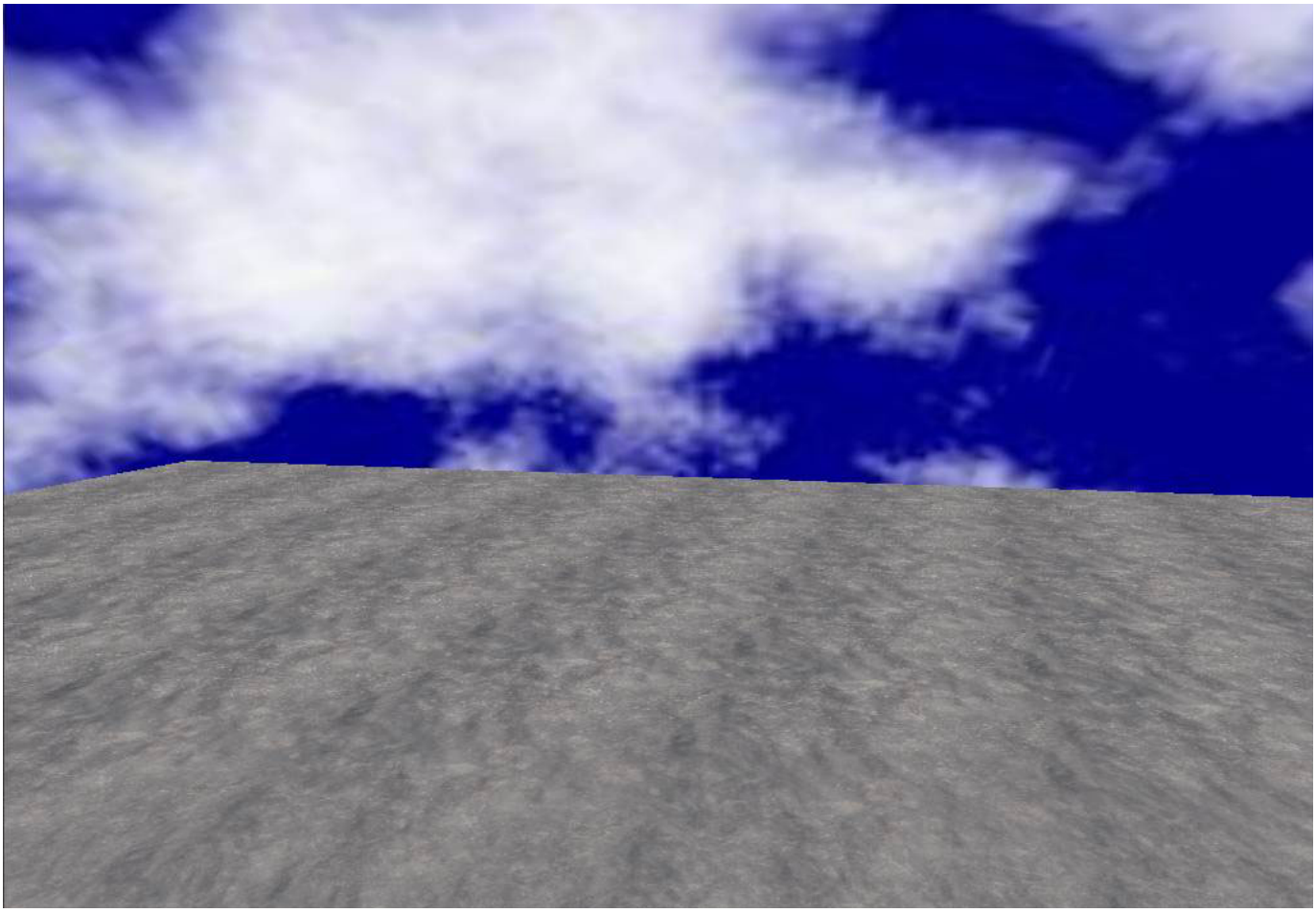

Figure 2.2: Terrain Before Manipulation.

The main purpose of this research was the easy creation of story objects in such a way that those objects would interact with each other in a meaningful way as defined by the story teller. The scope of this research was too ambitious and it resulted in encountering many problems early on. One of the main problems was the speech recognition software in use. Speech recognition does not always work properly due to the different accents being used the world over. If there is an error in understanding of the word, then the wrong object is created resulting in an error. To fix this error, the system required constant verification by the user to generate appropriate content.

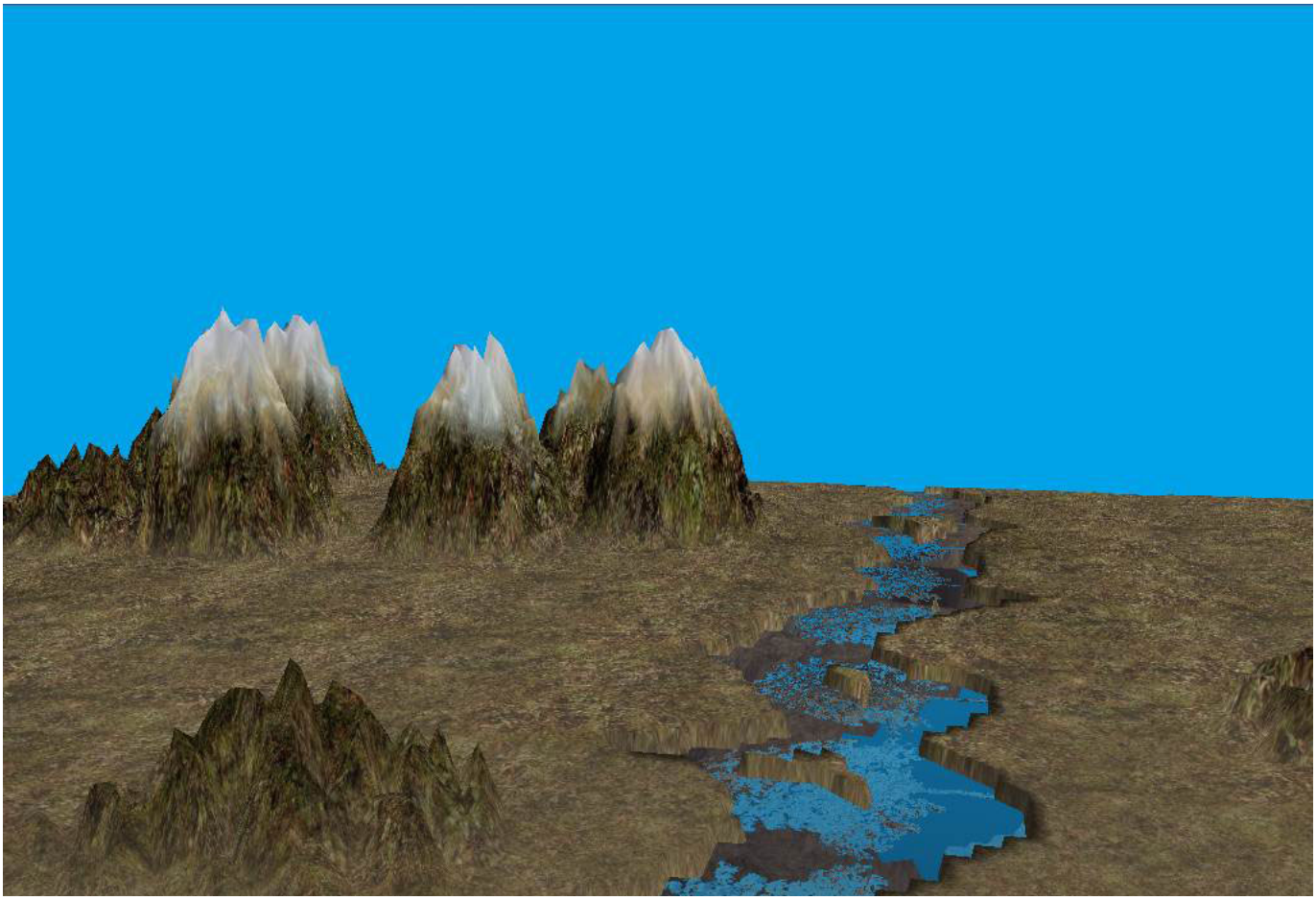

Figure 2.3: Terrain After Manipulation.

The second important problem was the creation of proper grammar. Accurate Natural Language Processing was required so that the computer knows what the user means and then it should create the tasks automatically. This in itself is an extremely large research topic and made it very hard to define the scope of the topic.

The final most important problem was the definition of the data library. The research aimed at generating objects based on the character's narration. This mean that either we needed to limit the number of objects that a character can use or create a library that constantly adds objects into it. If we limit the objects, then the research fails to produce

the desired results as the user will be greatly limited in storytelling. If we keep an ever expanding library, then we run into storage issues and the chance of error increases as the system does not always recognize the spoken word. The research however was useful in deducing the following points for taking it forward.

- We cannot keep the story generation system open, but we need to add a lot of constraints.
- Rather than defining our own stories, it was felt more appropriate to let the characters generate their own stories based on the objectives assigned to them.
- A story is a collection of tasks. By combining tasks together we can generate meaningful stories.
- User interaction is still required so that the user is not kept detached from the system.

With the addition of constraints in the form of limited user actions, the variability in story generation became less. Characters were defined on the basis of the actions that they contained and the stories were defined in the form of the sequence of actions that the characters should undertake in order to produce a meaningful outcome. It became evident that in order to produce a meaningful story a proper story structure needed to be defined.

2.1 Structuring a Story

A story is composed of characters and the actions that the characters perform. The sequence in which the characters perform these actions dictates how the story is going to progress. Not every character can perform every action, therefore the capabilities of the

characters in a story need to be limited. While creating a story, four rules need to be followed as proposed by Vladimir Propp [45].

- Actions of characters serve as constant elements in a tale, independent of who performs them.
- The number of actions needs to be limited.
- The sequence of functions needs to be identical.
- There is one structure for all stories.

Almost all of the stories follow the same principle as explained by Nigel Watts in his book, Teach Yourself Writing A Novel [52]. A role-playing game is normally based on a story and doing tasks in a certain order propels the story forward. As a result in order to produce a compelling game, it was found important to look into the story writing principles. A story's structure contains the following eight principles:

1. Stasis
2. Trigger
3. The Quest
4. Surprise
5. Critical Choice
6. Climax
7. Reversal
8. Resolution

These eight points are in accordance with the Freytags triangle [17] which is the normally used dramatic structure for telling a story.The real question that comes up with these principles is how to convert them into a role-playing game. The most important points are *The Quest*, *Critical Choice* and *Resolution*. These are the points that are necessary for a task to be performed. A character should be given a quest; there are many ways of performing the task; one path is followed to achieve the task. But what should be the factor that decides how a character makes a decision? How shall a character be defined so that every character appears to be different from every other character in the game? Lebowitz explains that for creation of believable characters in a story-telling program like the UNIVERSE, a Person Frame which is the collection of stereotypes, past events, traits and goals needs to be defined [33]. A character should have a history as that defines the probability of the character to perform a certain action. Goals define the ultimate aim of every character and through the combination of the tasks that the character is capable of doing.

2.1.1 Character Definition

Defining the stereotypes of a character seemed like a compelling way to defining a character. But a stereotype is nothing but a collection of actions that a character can perform. Also, many stereotypes overlap with each other and it becomes difficult to differentiate between two similar stereotypes. As a result a collection of actions seemed like a better way of defining the characters. In real life, the propensity of a character to do good or bad actions defines whether a character is good or bad. The same approach was taken here as the number of good or bad actions contained in a character defines the alignment of the character towards good or bad. But there can be many number of good or bad actions, so the correct combination of these actions holds importance. We find this combination by using a new approach where we combine the genetic algorithm with the Knapsack

problem to arrive at the character we want. This will be further explained in Chapter 3.

2.1.2 Plot Generation

There has been much research in the development of Dynamic Plot Generation Systems. Mostly, Propps Morphology [45] and Bayesian Networks have been used for the generation of such stories. The Story Engine [47] and the Open Ended Proppian Interactive Tale Engine [13] are based on Propps Morphology. In the non-linear interactive storytelling game engine (NOLIST) [5] uses Bayesian Network for finding the culprit in a murder mystery. The Dynamic Plot Generation Engine [3] also uses Bayesian Network uses the characters history for the development of a murder mystery.

Based on the observation done by Vladimir Propp all the stories follow the same structure. The initially proposed idea made the assumption that since all the stories follow the same structure switching items and characters will have little impact to the overall story. Genetic Algorithm was supposed to play a significant role in the generation of such a story.

Genetic Algorithm learns from the previously generated solutions and improves them by mixing similar solution together in every generation. Through user interaction the fitness of every step in a story will be analyzed and a new better story will be generated that is more in accordance with the choice of the user.The first step in generating a story by using GA will be to understand the structure of a few popular stories and form a structure of a general story after comparing them. The analyzed stories were:

1. Beauty and the Beast
2. Cinderella
3. Shoemaker and the Elf

4. The Emperors New Clothes
5. The Frog Prince
6. Hansel and Gretel
7. Jack the Giant Killer
8. Repunzel
9. Rumplestiltskin
10. Snow White and the Seven Dwarfs

Similarities

After reading the most popular fairytales, there were a few similarities that were seen among all the stories. The stories generally revolve around items of interest. All the characters take actions which are linked to the items or other characters which results in storys progression. Some action performed on an item normally triggers an event. The event can also be triggered by certain timing in the story which results in moving the story forward. After looking into the stories mentioned above, it was found out that almost all the stories play out like a Role Playing Game where different uncontrolled events or actions done by characters generate new tasks for the characters. The story progresses with the completion of these tasks and new tasks are generated to take the story further. Almost all the stories contain elaborate schemes and plots of the characters which can only be best explained by narrative. Here the aim was to replace the narrative with animations and let the user decipher what the story is about. In short, a story can be broken down into a series of steps which involve tasks being done by the characters.

2.2 Proposed Idea

The proposed idea at this point was the creation of a story by joining the tasks together randomly. The user will select which random order of tasks is most suitable and then that order will be selected as the parent of the next generation of tasks. Since the number of tasks, items and characters can vary greatly, this was thought suitable for creating new stories. This however provided an important problem and that was to maintain coherence in the story. For making a proper story, the tasks needed to be in a certain order which went against the logic of using Genetic Algorithm for story generation. As a result, the use of Genetic Algorithm was limited to the creation of physical and mental attributes of a character and a new technique was employed for story generation which fell under the domain of plan generation. The main advantages of this technique were that, the tasks were arranged in a proper order to maintain coherence and at the same time it offered several pathways of achieving an objective thus providing the characters with many different choices to take as was desired by opting to choose Genetic Algorithm for this problem. the subsequent chapters will explain this methodology in greater detail.

Chapter 3

Character Generation using Interactive Genetic Algorithm

A story is a collection of characters where they interact with each other and perform different actions. The characters can belong to different race, color or social structure and this in turn dictates their behavior under changing circumstances. A sure way of determining the role of the character as being good or bad depends on the actions performed by the character. It is important that the correct character is placed in the proper role to make the story believable and at the same time, the character should look the part or at least exhibit the thought process of the users who are going to interact with the story. Since the focus of this research had shifted from story narration towards animation, Role-Playing Games fit the correct genre to pursue the research. In a general Role-Playing Game, the story and characters are set by the game developer which remain the same through repeated plays of the game. This reduces the replay-ability of the game as the characters of the story behave the same way every time and they look the same too. A system was needed that allowed the users to create characters that are according to their liking. With that system, the users will be able to define the mental orientation of the characters, which will in turn allow the characters to behave a certain way under different conditions.

Many games allow the players to construct an avatar through a complicated menu system examples of which can be shown in Figure 3.1, where different types of avatars have been created by using the menu driven system of the popular game Saints Row: The Third.

Figure 3.1: Character Physical Customization in Saint's Row: The Third.

Characters in a story must always show something similar to the user's imagination to be believable. A good character performing evil actions and similarly a physically weak character cannot perform acts that require immense physical strength. Egri [11] describes a character as a sum of physiological, sociological, and psychological qualities. Based upon this he describes a checklist of various aspects that will influence behavior of the character which are as follows:

- Physiology (e.g., gender, age, height, weight, appearance, distinct, and physique);
- Psychology (e.g., moral standards, goals, obsessions, intelligence);
- Sociology (e.g., occupation, education, enemies, family life, friends, and hobbies).

There are many physical characteristics that can constitute the look of a character and there can be many number of combinations of those characteristics that define the final form of the character. Since this number of combinations can be immense, it was decided to use Interactive Genetic Algorithm for this purpose. There has been research in creating characters in different applications such as done by Zavala [54] where he has used Genetic Algorithms for creating a heterogeneous humanoid crowd population by using the Distance Function. In the current research, the Sigma Selection function is used to choose a character's physical attributes. Fujiwara and Sawai [18] demonstrated a method in which a 3D surface with simple facial features is created from a 2D surface by breeding geometry through Genetic Algorithm. Albin and Howard [2] use the power of Evolutionary Algorithms (EAs) on surface-based model geometry in order to create a complete avatar. Ventrella [51] who is also the chief author of the patent: Method and apparatus for creating and customizing avatars using genetic paradigm refers to a genetic approach to avatar design and customization and how it is possible to breed using sexual reproduction. In short, Genetic Algorithms have been used in the past for creating

characters in the past. We should now explore what Genetic Algorithms are and then use that information to explain the character creation process used in this research.

3.1 Genetic Algorithm

The Genetic Algorithm is a technique which provides optimal or near-optimal solutions under a wide range of selection pressure [22]. The convergence of the Genetic Algorithm is dependent on the amount of selection pressure which is the degree to which the better individuals are favored. If the selection pressure is too low, the convergence rate will be slow and the Genetic Algorithm will take unnecessarily long to reach an optimal solution [39]. But if the selection pressure is high, the Genetic Algorithm might get stuck at a local minima/maxima and might not be able to reach an overall optimal solution. The challenge is normally to find the sweet spot that allows us to get a good optimal solution in as little time as possible.

A Genetic Algorithm is a two-step process in which selection is applied on the original population to create a new population, in which some of the members are repeated, thus creating the intermediate population. Then crossover and mutation are applied to the intermediate population to create the next population. The process of going from the current population to the next population is considered to be one generation in the execution of a Genetic Algorithm. Goldberg [21] refers to this basic implementation as a Simple Genetic Algorithm (SGA).

For our character creation process, the acquired character after the completion of the Genetic Algorithm, must display some degree of randomness. Genetic Algorithm aims to optimize the solution by repeated iterations which reduces the chance that after every Genetic Algorithm run, the created character will be different. For this research, character that loosely fits the requirement of a role is good enough to be accepted. As a result, we are

searching for a solution that can be acquired by just looking at the local maxima/minima and we do not need to further optimize the results. For this reason, Elite selection is used for the two Genetic Algorithm implementations that are going to be discussed here. Elite selection quickly converges to a solution which reduces the number of iterations in a Genetic Algorithm. Since at every run, due to Elite selection, the Genetic Algorithm has a chance to get stuck at a separate local maxima/minima thus keeping the results fairly random.

3.2 Defining Physical Attributes

Nowadays many Role-Playing Games provide the users with the opportunity to create characters according to their liking. They make use of different tools at their disposal which are present within the game engine itself and help them in crafting the character the want. A few years ago, this crafting system was very basic but with time it has become ever more complex. The character creation can be either in 2D or 3D, depending on the game. If it is a 2D character creation, then the development team has to create many art assets that the user can use for creating the character. If however the creation is in 3D, then normally a single 3D model of a character is created and there are different sliders present that change the appearance of the character by changing the 3D mesh of the character. For both the cases there can be many number of physical attributes and using different combinations will create a different looking character. This however becomes a combinatorial problem because as we increase the number of combinations, the chance of the user to arrive at a character combination that was found pleasing the last time becomes slim. This was a major concern when creating the character creation system for this research because a system was needed that exhibited the past choices made by the user to have an impact. Genetic Algorithm has the property of storing the history in the

form of parents as the algorithm works. At the same time, it has the ability to handle large amounts of data and helps in converging at a solution quickly which fit in well with the requirements for character generation.

For this research the Unreal Development Toolkit (UDK) was initially used to help with character development. UDK is already being used for a large number of educational projects and since it is a game development tool, it seemed to be the best choice for developing the game/story generation system. UDK provides the possibility of creating a modular character objects, called modular pawn. The 3D character is created by joining parts together which are then governed by manipulating the underlying skeleton. Therefore we can place any model on top of the skeleton and it will move according to the bone structure of the skeleton. This can best be explained by Figure 3.2:

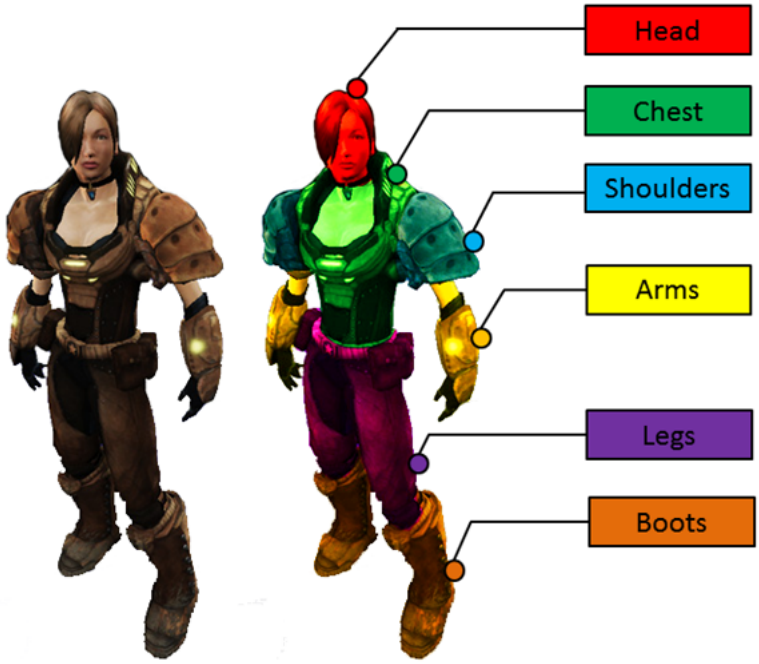

Figure 3.2: Modular Pawn in Unreal Engine 3.0 SDK.

The problem that came up with using UDK was that it required a large amount of 3D model assets to be created to show the proof of concept. Acquiring or making such assets is not easy and due to time constraints it was decided to shift to Microsoft's XNA Game Studio 4.0. XNA provides an easy way to develop both 2D and 3D games and hence the 2D game development route was selected as creating 2D assets is far easier. The underlying concept for character generation remains the same however and only the representation of the character changes. As a result, the search space in which the Genetic Algorithm was previously working on was enhanced quite significantly. The proof of this concept was given in [28] where a total of seven body parts with each containing 10 types as shown in Figure 3.3.

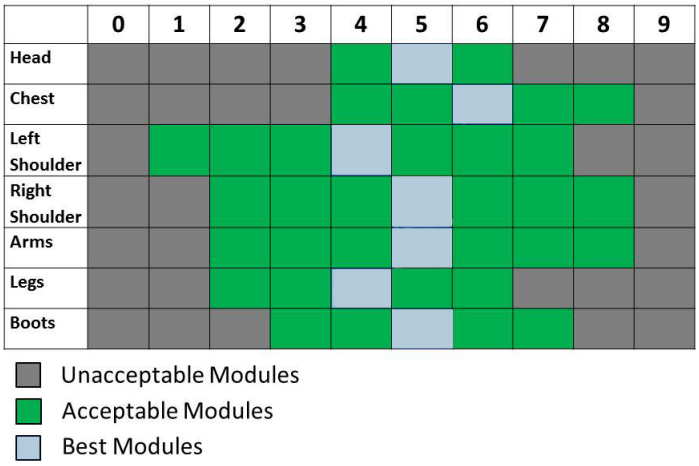

Figure 3.3: Acceptable Modules.

This made the total number of possible combinations to be 10^7, which is a modest total but far less than what can justify the use of a Genetic Algorithm. By using 2D assets, this total was changed to 50^{11} as now the character was composed of eleven parts and each part had a total of 50 types to choose from as shown in figure 3.4.

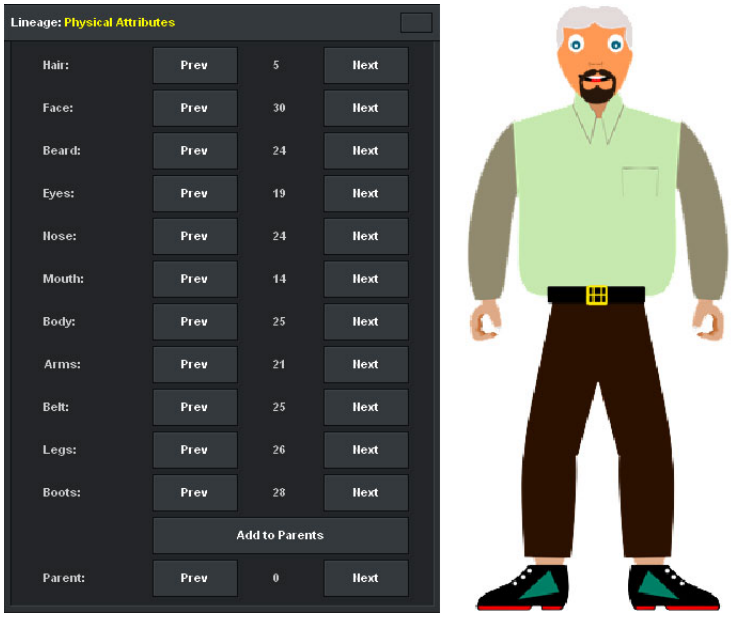

Figure 3.4: 2D Character.

3.2.1 Sigma Selection

User selection plays a major role in this selection. The users create characters and add them to the list of parents that the Genetic Algorithm uses to produce the next generation. As the user selections start to fall within a certain range after a while, the mean of their choices represents the optimal solution. Standard deviation represents the acceptable range from which the system will select the type for a certain part of the character. In short, to make a selection for every physical part of the character, the system selects a type that falls within the standard deviation range for that part. The mean exhibits the best possible type that can be selected for that part. Since the convergence needs to be for an acceptable candidate rather than the perfect candidate any type that lies within the standard deviation is acceptable. To make this possible, elite selection is used for the selection of the parents for every generation while sigma selection is used to retrieve the candidate.

3.3 Defining Mental Attributes

According to Propp [45] the number of actions that a character can perform needs to be limited. Based on the complexity of the story, this number can change and it is best left to the creator of the story to decide what this number should be. A character is a sum of its actions and that also defines its orientation towards good or bad. For this reason, actions were defined by associating numerical values to them [27]. A more detailed explanation of actions will be given in 4.1

For research purposes, the total number of actions that can be performed were kept to be 20 although this number can change based on the complexity of a story or depending on the system that needs to be designed. These 20 actions are distributed into 10 good actions and 10 bad actions. For our Genetic Algorithm a chromosome can contain a total of 10 actions, which is a combination of good and bad actions. The chromosomes in this implementation is dependent on the types of actions contained in it rather than the order in which the chromosomes are kept. The genes which represent actions are numbered from 0 to N-1, where N is the total number of genes, which in this case are 20. Genes from 0 to 9 are taken to be good actions and the genes from 10 to 19 are taken to be bad actions. The numbers reflect the intensity of the action. The lower numbers represent good actions and the higher numbers represent bad actions. The sum of these actions decides whether the character is a good or a bad. For example, a good character that has all the good actions will have a combined gene sum of 45. We combine the Genetic Algorithm with the Knapsack problem at this point. The Knapsack problem is used for combinatorial optimization where you need to determine the number of each item to include in a collection so that the total amount is less than or equal to a given limit. The difference here is that an item cannot be repeated and the number of items remains fixed as all the chromosomes must have the same amount of genes.

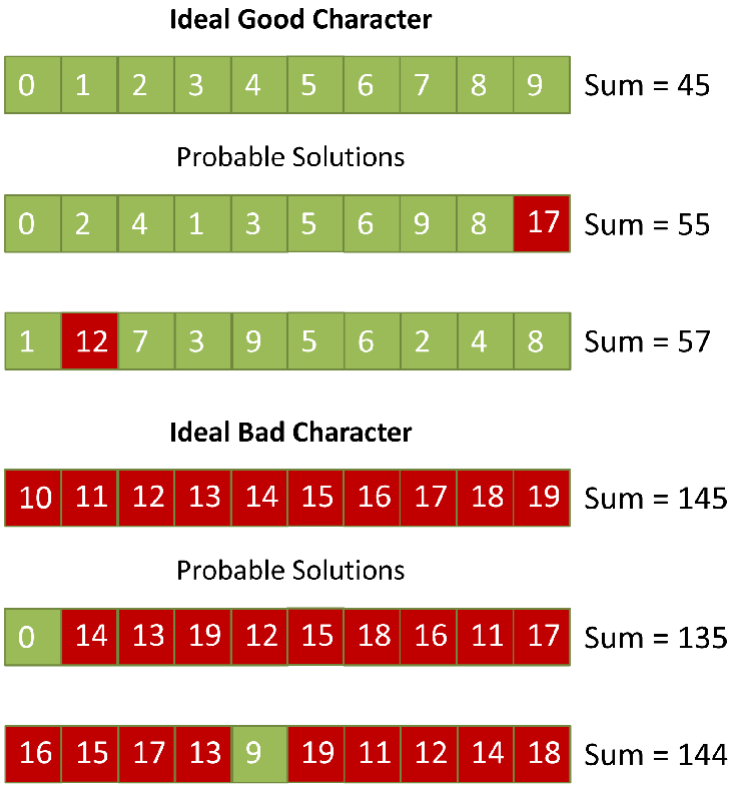

Figure 3.5: Probable Solutions.

Based on the requirement of characters within a story, the different characters will have different gene sum and thus have different orientations. As a result, the number of ways that sum can be achieved is increased with the number of genes present in the gene pool. As shown in Figure 3.5, the Genetic Algorithm has been used to find probable solutions for different kind of characters. Since the emphasis is on finding a close enough solution rather than a perfect solution, any one of the given solutions can be accepted.

Thus, from a gene pool of N actions, the number of genes in every created chromosome is $N - N/2$. According to the requirement of the characters, the Genetic Algorithm tries to find the chromosomes that are closer to the desired solution.

After every crossover, the repetition of a gene is checked. If there is a repetition, the

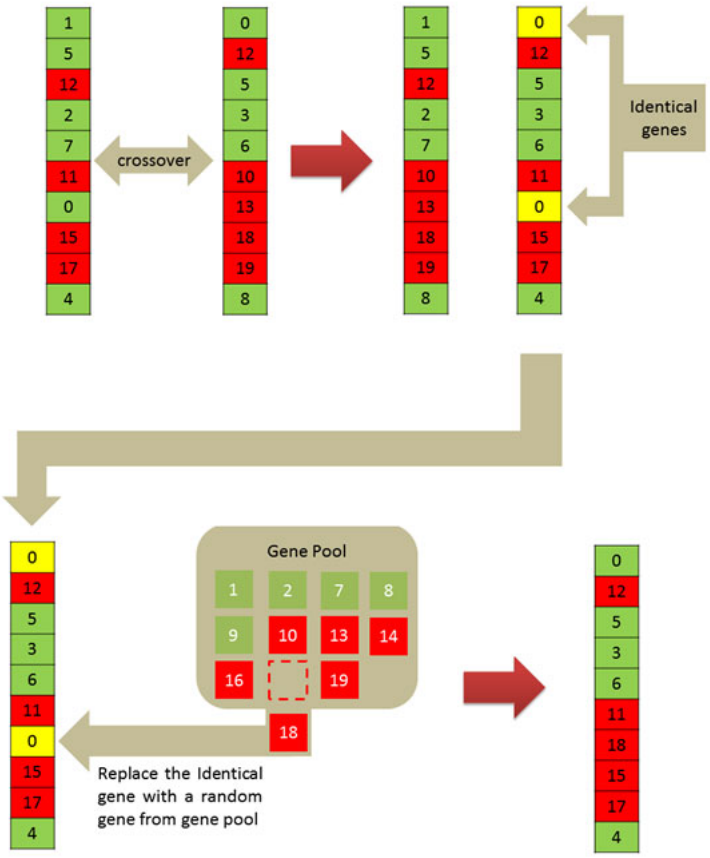

Figure 3.6: Complete Crossover Operation.

repeated gene is replaced by another gene from the gene pool. For mutation, a random gene is picked from the chromosome and it is replaced by another gene from the gene pool. After following this approach over a set number of generations, a chromosome is acquired whose sum of actions is closest to the desired character. This process is best described in Figure 3.6.

3.4 Complete Character

Character customization in Role-Playing Games is only limited to the character that is going to be used by the player only and all the other characters are predefined by the

game creators. This technique has a few advantages as the art team decides how a certain character should look like and it sometimes feels appropriate to make a certain character look a certain way based on the actions that character is supposed to perform. The disadvantage of this approach is the lack of variety that comes with it and reduces the replay-ability of the game itself and results in the user losing the interest in the game. This research hopes to provide a system that not only creates random characters every time the system is run, but also provides them with different set of actions. This system recommends the character based on the choices made by the user previously so that the character adheres to the general trend followed by the user. The user however is not stopped from making changes therefore it provides the flexibility to have a completely different looking character if the user so desires.

At the same time, the user can also provide the character with a different set of actions which can not only result in changing the orientation of the character, but also cause the character to achieve the same objective differently due to the change in actions that the character can perform. Therefore, the character acquired after the application of the two Genetic Algorithms can not only look different but can also perform differently which makes the entire system unpredictable resulting in a simulation of agents that can create different results under the same conditions.

Chapter 4

Formulating a Plan

After the creation of characters the next important aspect of the research was to define how the characters operated within a given environment. One of the first methodology that was analyzed was Finite-State Machine (FSM), which is the most common methodology used for programming Artificial Intelligence (AI) in games. This powerful organizational tool helps the programmer to break a problem into sub-problems and allows the implementation of intelligence systems with ease and flexibility [48]. The FSM is initialized by first declaring all the states, then declaring each state's transitions with its required conditions. The FSM is then put in one of the initial states and then it waits for a transition to occur. If some transition has occurred during the time the game was run, then the correct state is acquired as the current state of the FSM. FSM is a reactive mechanism which adheres to the changes in its surroundings to initiate transition to the next state; it does not give a suitable initiative that governs the states that it must achieve in order to complete its goals which is a more natural behavior. Orkin [42] realized this limitation and introduced planning techniques based on STRIPS [15] and goal-oriented action planning [41] in the game, F.E.A.R., which lead to a more natural behavior. The advantage of this technique is that, it provides several paths by which a single goal can be achieved. Since the goals of a character are separated from the plans generated to reach the goal, if

a plan fails it does not lead to giving up a goal but an alternate route is acquired which takes into account the new information about the world that led to the first plans failure.

This research includes user interaction, evolutionary algorithms and autonomous agents and a robust decision making process for creating animated stories in a game world. At the start of the game, the characters and their abilities are chosen and they are given different objectives. Based on their available states, the characters formulate a plan and perform actions to achieve that plan. Pollack and Horty [43] argue that in a dynamic and real-time planning the autonomous agents must be able to manage the plans that they generate. They must be able to decide when a plan should be kept inflexible and when they shall add more detail to an existing plan. The extra detail in this case comes from the actor's decisions taken on a certain route in a plan.

Dignum et al., [8] argue that coupling agents to games requires a design methodology where agents should be included from an early stage in the design process to profit from specific agent characteristics such as communication, cooperation, reasoning, proactive behavior, and adaptability. In the current system, the Agents are added from the beginning and it is their characteristics that define their behavior as they adapt to the changing conditions and apply reasoning based on their orientation to reach a decision.

In the current system the AI of the system uses a suitable plan in order to complete its objectives by employing a planning graph [29]. A planning graph defines a series of actions that when done in a proper order leads the AI towards its goals. If at any point performing an action becomes impossible, it should provide a backup plan for completing the objective. This planning graph can best be explained by explaining the various aspects of character, items, actions and states that are used for formulating a workable graph solution. The most important of these is an Action around which the entire graph is created and it will be explained first.

4.1 Action

Action is the most important component required for creating a planning graph. A character contains a list of actions that it can perform and the actions also define the character's orientation. At the same time, an item within the game contains a list of actions that can be performed on that item only. The thought process behind this allocation was that, every character cannot perform every action. That is the reason why different people in the real world have different jobs. For example, not everyone can become an astronaut. For that job you need the best pilots from the air force of a country, who are medically fit and smart enough to perform the tasks required of astronauts.

Similarly not every action can be performed on every item. For example, you can eat an apple but you cannot eat a hammer. You can however use a hammer to nail something, to loosen nails or even use it for defending yourself. In order to do that, we will need to add the actions for nailing, loosening and defending to the hammer. At the same time the character must have all of these actions if it wants to perform these actions. A carpenter might be willing to perform the first two actions, but when it comes to defending himself, he might throw away the hammer because he is incapable of defending himself. In such a scenario the created graph will rightly tell the carpenter to use the hammer, but his inability to do so will make the carpenter choose another course of action to defend himself.

4.1.1 State

The condition in which an item is in before and after the application of an action is called a *State*. An item can have many states which change due to an application of an action. New states can also be added after performing an action on the item. There are two kinds of states that an action can have. One is called *Preconditions* and the other is

called *Effects*.

Preconditions

For any action to be performed, some states are needed to be satisfied. These states are called Preconditions.

Effects

When an action is performed, it produces resulting states which are called Effects

There is also a third type of states known as the *Goal States*, which are a combination of effects acquired due to a series of actions. Goal States are objectives that a character needs to achieve with regards to a certain item. Based on these states, a plan is devised, acting on which a character should achieve the desired states.

How the system works is that for every item in the game, every character has some objective states that it wants to achieve. These objective states are given to the item which then takes into account the initial state in which the item is in and the actions that can be done on the item and produces a plan following which can help the character to achieve the desired states for that item. This plan creation process will be explained now.

4.2 Planning Graph

A Planning Graph encodes the planning problem in such a way that many useful constraints contained in the problem are found out to reduce the amount of search needed. Furthermore, Planning Graphs have polynomial size and can be created in polynomial time [6]. For this reason, we are using a technique called Graphplan to create a planning graph based on the defined rules. Theplanning graphis a layered acyclic graph in which the layers of vertices form an alternating sequence of literals and operators or states and actions. Its edges are only permitted to connect vertices between successive layers.

$$(S_1, A_1, S_2, A_2, S_3, A_3, , S_k, A_k, S_{k+1})$$

To each action $a_i \epsilon A_i$, a directed edge is made from each state $s_i \epsilon S_i$ that is a precondition of a_i. To each state $s_i \epsilon S_i$, an edge is made from each action $a_{i-1} \epsilon A_{i-1}$ that has s_i as an effect.

Graphplan was first introduced using the STRIPS language. Since then, more expressive languages like ADL [25] and UCPOP [20] have been used for plan representation. These languages allow the use of conditional effects, disjunctive preconditions, and universally quantified preconditions and effects for representing actions and goals.

The planning graph was constructed by following the guidelines given by Hong [24] for the construction of a Goal Graph and other literature for creating a Graphplan. Due to the nature of the research, relying on a language like STRIPS or ADL was not an option. Due to the nature of a game, the planning graph needed to be created in real-time as the actions done by the characters may change the initial state of the items which will require for the planning graph to be updated. The concept of a planning graph is to map out all the possible outcomes that can take place within a certain time period. If the goal states are reached within that time period, the planning graph is considered to be a success. Once the goal states are achieved, backtracking is done from them up to the initial state noting down all the actions that need to be performed along the way.

At any level within the graph, there can be one or more actions that lead to the same state. Since all these actions are stored during the back tracking, the character has a choice to pick any action that best suits its agenda. The suitability of the action is determined by a multi-step decision making process that takes the orientation of the character under account. This decision making process will be explained in Chapter 5.

4.2.1 Creating a Graph

This section will explain the graph creation and the impact of changing initial states and preconditions. We start off by creating a graph for a character whose initial state is *Hungry* and he wants to achieve the state, *Not Hungry*. As shown in Figure 4.1 (a), The possible actions that can be performed are Do Nothing and Eat. The character can perform the Eat action to reach the goal state.

In Figure 4.1 (b) the precondition for Eat action is changed; now the character must have food and it should also be hungry to perform then Eat action. The graph changes a little but the outcome remains the same and that being, in order to complete the objective, the character will need to perform the Eat action.

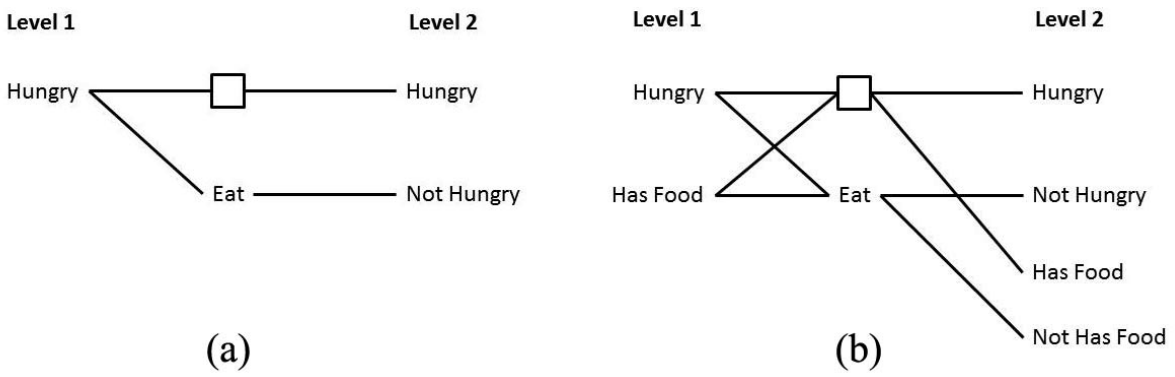

Figure 4.1: Creating a planning graph by changing the preconditions.

We change the initial state from Has Food to Not Has Food. Since the character does not have food, the Eat action cannot be performed. This calls for finding an action that makes the availability of food valid. The character will now first perform the BAKE action which will produce food and also create the effect Has Food. The graph that is acquired is shown in Figure 4.2.

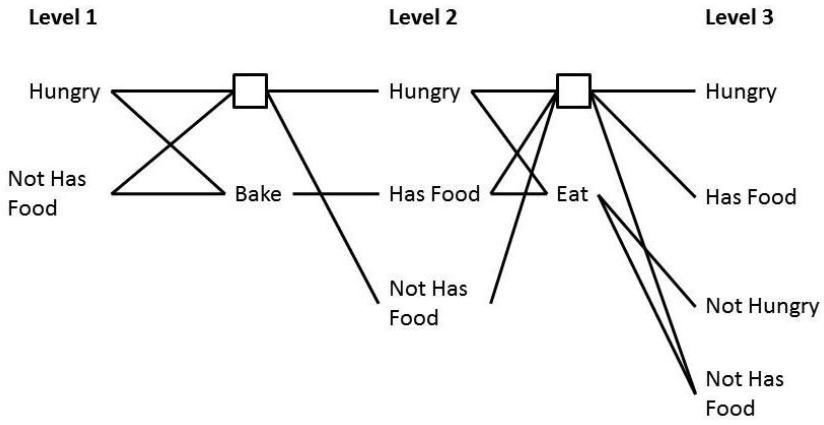

Figure 4.2: Planning graph that is acquired after different initial states.

4.2.2 Backtracking

The planning graph is considered to be complete when the goal states are achieved. This means that all the possible ways that the desired states can be achieved, have been achieved. There can be more than one path that lead to the final goal states, hence we need to check for the possible parents for every goal state. We do this by backtracking from the goal states to the parent states on every level, noting down all the actions done along the way. If we look at Figure 4.2, the goal states are achieved in Level 3 and when we backtrack from them to initial states, we find that the actions that the character will need to perform are Bake and Eat. As a result, this is the order in which the actions will be performed by the character and it will disregard all the other actions that are present. In this example, there is only one path that leads to the goal state, but it is a possibility that there can be multiple routes that lead to the same goal for another problem.

In the story, *Little Red Riding Hood*, the mother of Red Riding Hood initially does not have a cake that she wants to give it away, so her initial states with respect to the cake are ¬Have (Cake) and ¬Given(Cake). She needs to own a cake before she gives it away, so that her final states come to be ¬Have(Cake) and Given(Cake). Therefore, we

see in Figure 4.3, that she first bakes the cake to gain the state, Have (Cake) and then gives it away to reach the final states, ¬Have(Cake) and Given(Cake). This is achieved in Level 3, however, we expand the graph one more time to see if the states are going to be repeated. As the states in the next level are the same as in Level 3, we arrive at the conclusion that the graph cannot be further traversed and backtrack from the final states in Level 3 to arrive at the beginning of the graph, selecting the actions performed in between.

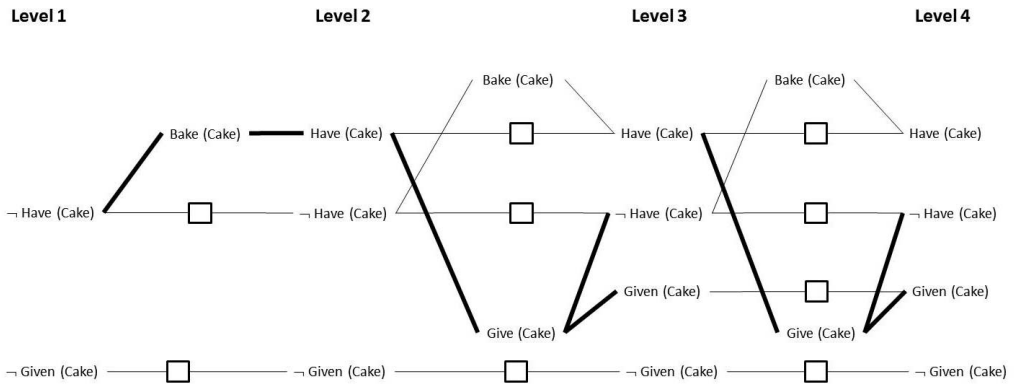

Figure 4.3: Planning graph that is acquired after different initial states.

Once all the actions at the different levels have been acquired, they will define the several paths that can be traversed to reach the goals states. This path will be traversed by using the A* algorithm, which is actually the most common path-finding algorithm used in games [42]. A* is a directed algorithm and does not search for a path blindly but at every step it evaluates the best possible route to take [38]. This path selection is further enhanced by the Decision Making Process of the character which further enhances the A* algorithm in deciding the best possible route at a given time.

4.2.3 Comparison with State Transition Diagrams

The same graph can also be reproduced as a State Transition Diagram as shown in Figure 4.4. There are however some advantages in using a planning graph over a state transition diagram. One of the big disadvantages for using a state transition diagram is that you have to define all the possible states of a system which becomes a problem for large systems as there is an exponential growth in the number of states. This explosion in the number of states leads to state transition diagrams becoming far too complex for much practical use.

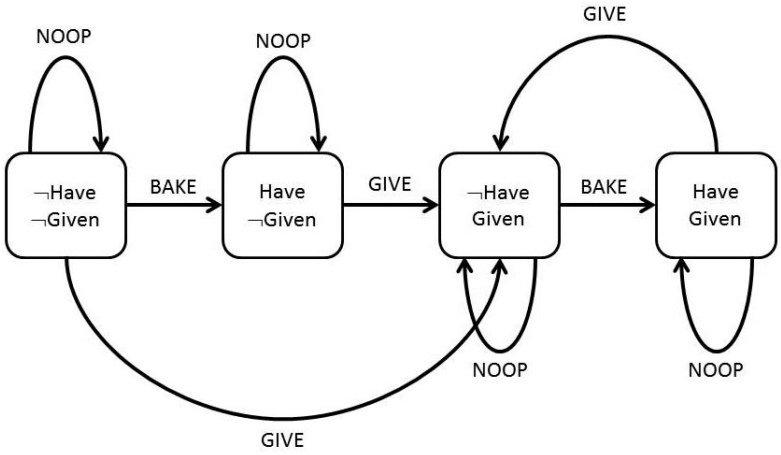

Figure 4.4: State Transition Diagram.

The current system does not put any limit on how many actions and states that are going to be managed by it and if the administrator decides to have a large number of actions and states, then using state transition diagrams becomes unfeasible. Using a planning graph appears to be a viable choice in this scenario as the number of states can be quite high if desired by the administrator.

4.2.4 Graph Structure

Like the other implementation of the graphs, linked lists were chosen for creating the planning graph. In this graph, the states represent the vertices and the actions represent the edges of the graph. The first level of the graph contains the initial states and final level of the graph contains the goal states, provided that the goal states are achievable. This is a weighted acyclic graph, which means that for traversing any edge, there is a cost involved and the graph can move only in one direction. For this implementation of the planning graph, a linked list of a specialized structure knowns as *Graph-Node* is used. Each Graph-Node contains the information about its parent nodes, its child nodes and the actions that can be performed to reach that state. This can best be explained by referring to Figure 4.5 which shows an example of a simple planning graph.

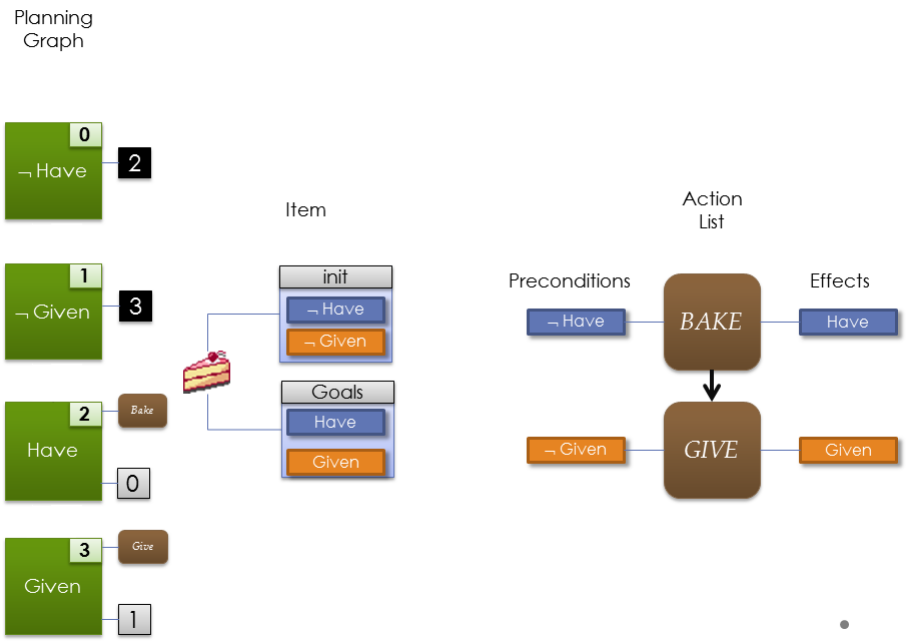

Figure 4.5: A Simple Planning Graph.

In Figure 4.5 we can see that the initial states of the item are ¬Have and ¬Given and the goal states are Have and Given. We can also see that there are only two actions in the Action List, namely Bake and Give. Based on this information, the planning graph has been created.

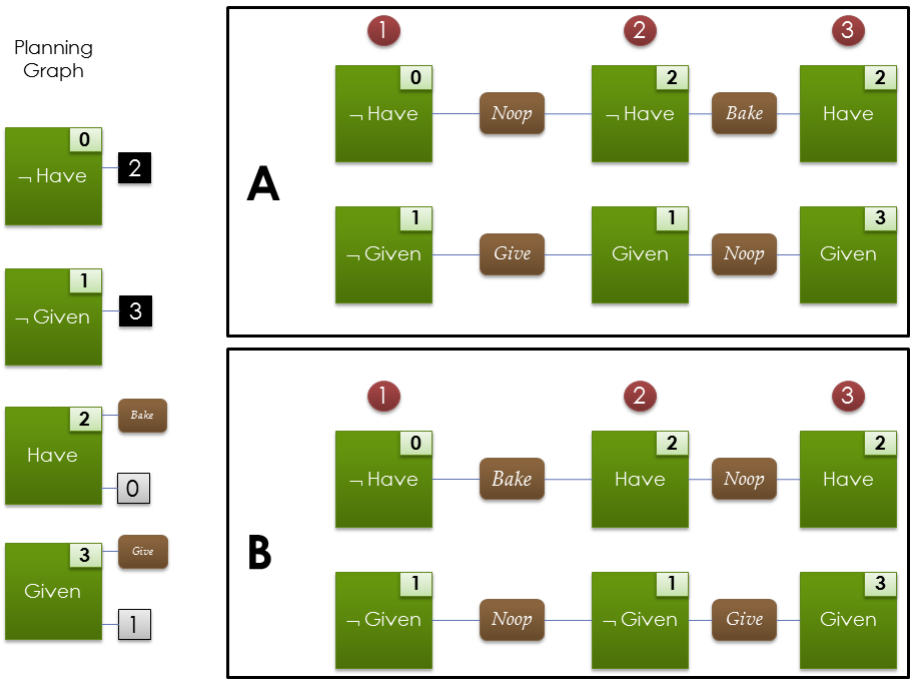

Figure 4.6: Two Interpretations of the Planning Graph.

This planning graph can be interpreted in two ways as shown in Figure 4.6, therefore we need to make sure that we get the accurate graph and this is achieved by giving cost to every action. As a result the action with the lower cost value will be selected first and the graph will be interpreted in only one way. Once the graph has been acquired, the order in which the actions should be performed are then sent to the character asking for the planning graph. The character then uses an elaborate decision making process to select which actions it should perform to reach the objectives.

Chapter 5

Decision Making Process

Decision making process plays an important role in describing the behavior of the characters. Creation of the planning graph is the first part of the process which is followed by the decisions taken by the characters to perform certain actions [30]. Character creation as explained in Section 3 comes into play as well as it defines the actions that can be performed by the character. Before we explain the decision making process, we must explain the concept behind its workings to have a complete grasp on its implementation. Unless that is defined in detail, this process has been known to cause confusion and creates misunderstandings.

5.1 Actions Done on Items

How this system works is that before the start of the game, there is an administrator who selects which items are going to be the part of the game, what will be their initial states and what are the actions that can be performed on the items themselves. We have to understand the concept that not every action can be performed on an item. For example, a wrench can be used for opening the bolts and a skilled mechanic has the skills to do so. So an appropriate character with the appropriate skills can use the wrench properly. A doctor on the other hand may not be suitable in performing the same job as he does

not have the appropriate skills. At the same time, a wrench cannot be used to administer any drugs which can only be done by using an injection. Therefore a proper correlation between the item and the actions is required in order to generate a suitable plan.

Every item has a one or more initial states and every character might have one of more objective states that it wants to achieve in relation to that item. As a result, the item is provided with these goal states by different characters to get different planning graphs for them to perform. The item has a list of initial states and it uses the many actions that can be performed on it to produce a graph that may help a character to reach the final goal states. A graph is generated for every character to reach its goal states and the list containing the order in which actions should be performed are passed on to the character. There can be many paths that can be taken within this graph, so an efficient algorithm is required to make this traversal. A* algorithm was deemed to be most suitable for this purpose as explained in Section 4.2.1. But the most important part of the A* algorithm is to find the most suitable Heuristic, which turned out to be slightly complicated in this case.

5.2 Selecting Heuristic

Defining a admissible heuristic is complicated for this implementation of A* algorithm. One of the easiest ways by which it could be defined was to have the action with the lowest cost repeated over and over again. The problem with that approach is that actions depend on states and whenever an action is performed, it ends up changing the state of the item. As the effects of one action become the prerequisite for the other, it makes the application of the same action repeatedly impossible as shown in Figure 5.1.

The second technique that can be used is to have two actions performed alternatively. The problem with this approach is that the two actions may not satisfy all the states that

Figure 5.1: Repeating same Action is not a Viable Heuristic.

need to be achieved in order to arrive at the desired goal states. This can be solved by breaking the Action selection into two parts. The first part being the selection of actions that satisfy the initial states.

- An item can have many initial states
- Based on those initial states, you find the actions that can be performed
- Then select least costing one from the doable actions

The second part being the selection of actions that satisfy the goal states.

- An item can have many objective states
- Select all the actions that can reach one of those states
- Then select the least costing actions that can reach one of the states

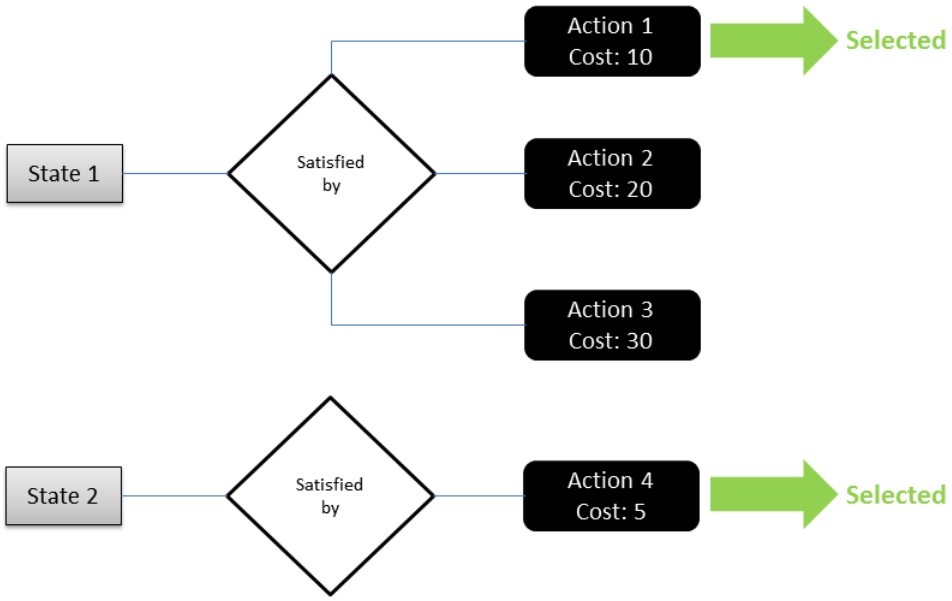

Figure 5.2: Alternate Actions as Heuristic.

As a result, we get a list of actions that must be used to get a viable heuristic. The system can then use the least costing actions from the list for every state to arrive at a heuristic for comparison as shown in Figure 5.2. All the states get covered and the problem that a single action can not be consecutively repeated gets solved. A small example is shown in Figure 5.3 where the two actions completely define the heuristic for the A* Algorithm to start from the initial states and end up at the final states.

5.3 Sorting Through Actions

Once the action order list is obtained, the character needs to decide which action should be performed through a series of checks. The actions contained in the list contain the information about the cost of the action, the level in which it should be performed and the orientation towards which it is aligned. The system does a series of passes to accept or reject an action based on this information.

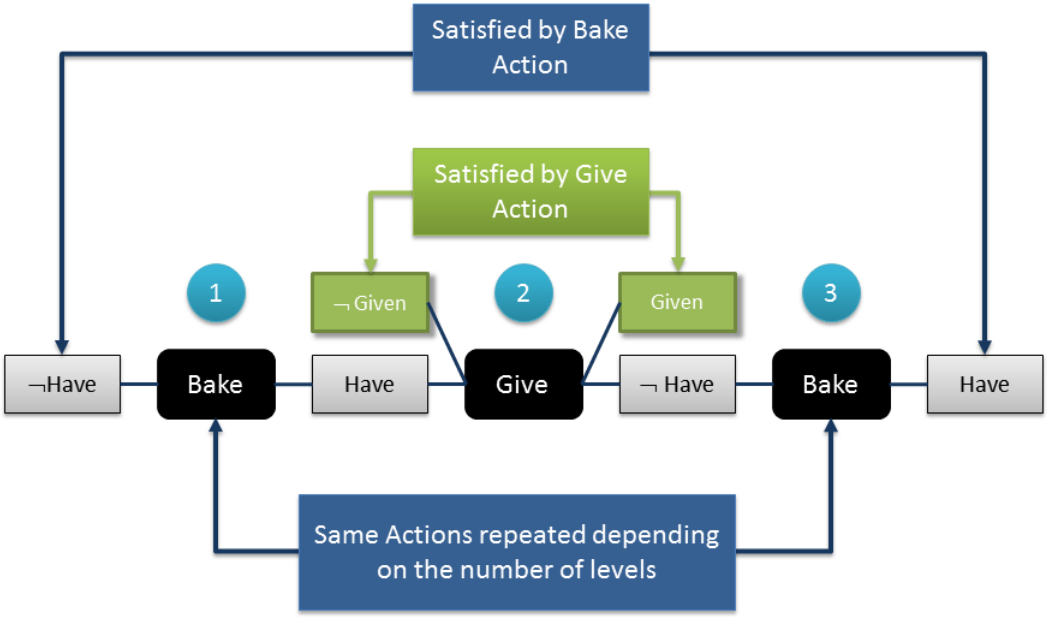

Figure 5.3: A Heuristic Example.

5.3.1 Availability of Action

The action order that is returned by the item is dependent on the list of actions that can be performed on the item. This means that if we want the item to be in a certain state after a passage of time, we need to perform the actions in a certain order. This however does not mean that those actions can be performed by the character as well. As explained in section 3.3, not every action can be contained in the list of mental attributes of a character. Therefore it is quite possible that the best possible action that can be done on the item is not available to the character. All such actions will be avoided, thus limiting the decision paths. This is shown in Figure 5.4 where all the unavailable actions are marked and all the paths that lead to/from those actions are represented by red lines showing that these paths will not be taken while making a decision.

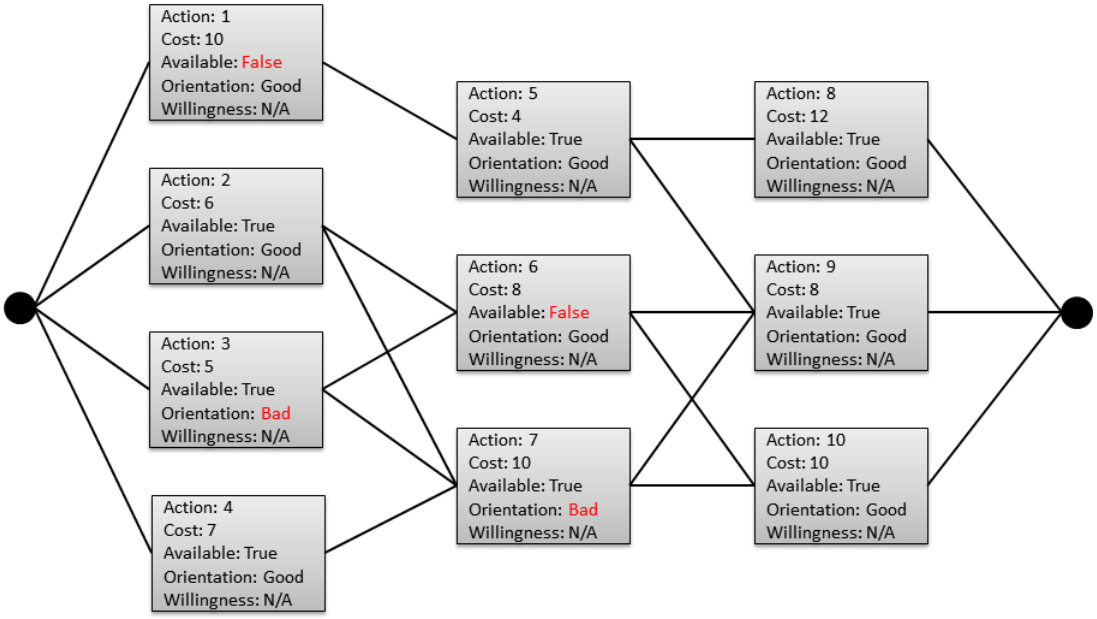

Figure 5.4: Marking Unavailable Actions.

5.3.2 Orientation of Action

Every action has a cost associated with it which separates it from other actions of the same orientation. There however can be an action with a lesser or the same cost but with an opposite orientation. The thing that matters in such a case is to select the right action for the job. The way of doing this is to add an additional cost to the action thus making its selection to be improbable but not impossible. This cost addition ensures that the character first opts to do the action that is more according to its orientation before choosing the action with the opposite orientation. Therefore, after the first pass where all the not doable actions are marked, the second pass adds extra cost to all the actions which have opposite orientation as shown in Figure 5.5. It should be kept in mind that this does not take place for characters who have neutral orientation.

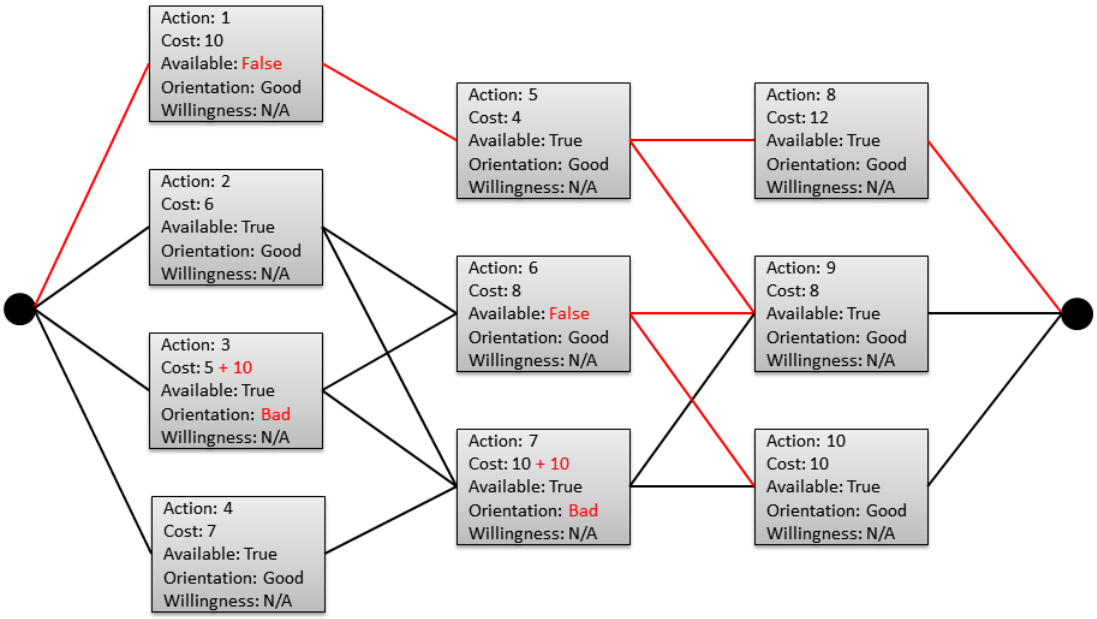

Figure 5.5: Marking Actions with Opposite Orientation.

5.3.3 Willingness for Action

The final pass finds out the willingness of the character to perform every action in the action order list. The willingness is dependent on the ratio between that actions with the same orientation to those that have opposite orientation. For example, if the total number of actions is 10 and the good actions are 8, then there is an 80 % chance that good action will be selected by the character. After this final pass we will get the complete picture about what actions can be selected by the character to achieve its objectives. If you look at Figure 5.6 you will see that Action 9 has lower cost than Action 10, but the character has chosen not to do Action 9. The only remaining option left is Action 10. Under the same conditions, it is quite possible that the character might opt to do both Action 9 and Action 10. In that case, due to the nature of the A* algorithm, Action 9 will be selected as it exhibits lower cost.

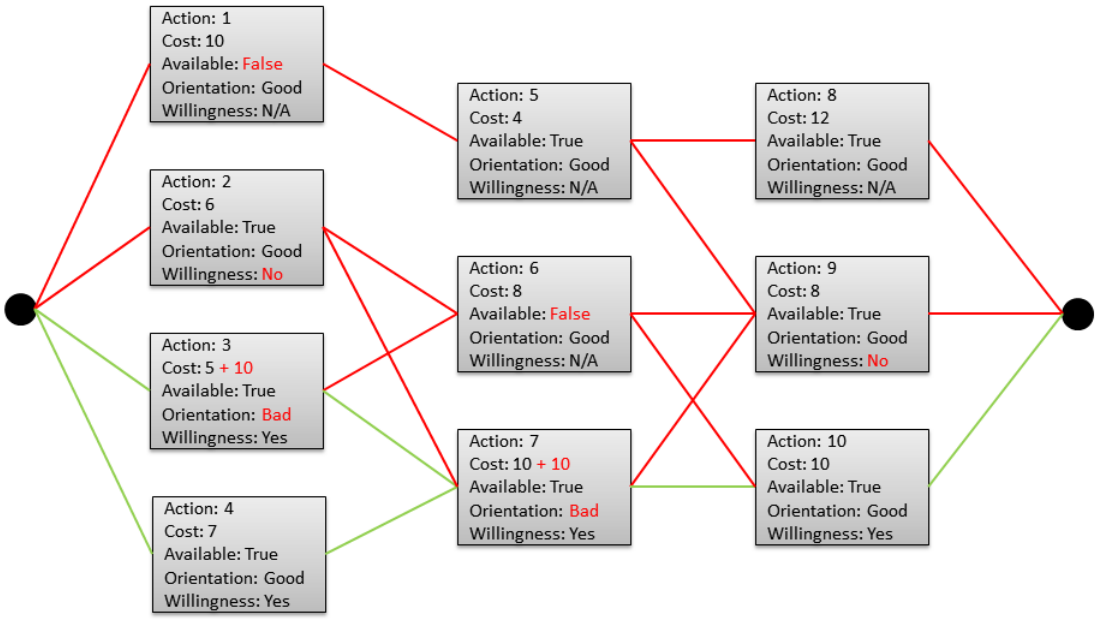

Figure 5.6: Marking Rejected Actions.

5.4 A* Traversal

If we look at Figure 5.6, we can see that the graph that is received has a wide variety of paths. The decision making process further reduces the paths that the algorithm will need to analyze. This is very much like how A* algorithm works when it encounters impassable terrain. Afterwards the A* uses a best-first search and finds a least-cost path from a given initial node to the one goal node. As A* traverses the graph, it follows a path of the lowest expected total cost or distance, keeping a sorted priority queue of alternate path segments as it proceeds.

In A* search the evaluation function (where n is the current node) is:

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

where

$f(n)$ = the evaluation function.

$g(n)$ = the cost from the start node to the current node

$h(n)$ = estimated cost from current node to goal.

$h(n)$ is calculated using the heuristic function that was defined in 5.2. With a non-admissible heuristic, the A* algorithm could overlook the optimal solution due to an overestimation in $f(n)$. the requirement for the admissible heuristic is:

$$\forall n, h(n) \leq C(n)$$

where

n is a node

h is a heuristic

h(n) is cost indicated by h to reach a goal from n

C(n) is the actual cost to reach a goal from n

Based on this principle, in Figure 5.6 the algorithm first has a choice between Action 3 and Action 4. It goes with Action 4 as it is the least costing one. Afterwards the choice is simple as there is only Action 7 which leads to Action 10, thus allowing the character to achieve the goals.

The Decision Making Process analyzes the Action Order List that was received after the generation of the planning graph and marks the decision path that the character should take to reach the goals. The character's attributes and the willingness of doing an action plays an important role in selecting what actions will be done by the character. Down the line, everything is dependent on the random number generation, but it is controlled

by managing the probability. Other factors can also be added to further enhance the decision making like character's motivation in achieving a goal, which slowly decreases with every failed attempt at reaching the goal states. Such factors will make the process more complicated, thus allowing the characters to exhibit a more natural response.

Chapter 6

Complete System

When this research was started, the aim was to have a dynamic system that is influenced by user interaction. The initial concept was for the user to define all the objects by voice and for the system to identify every object and create it. The problems with such an approach were explained in Chapter 2 and the research direction was changed. The user involvement is still a critical component of the system, but its usage has changed. The emphasis of the research was to have a separate game story every time the system is run. The story is dependent on the tasks done by the characters in the story, so there needs to be a flexibility as to what tasks can be assigned and who is going to perform those tasks. The user can therefore select the orientation of the characters who are going to participate in the game, how they will look like, what items they will perform actions on and finally what objectives they will have. For this reason, a system was created that lets the users do just that one step at a time and it will be explained here in detail.

6.1 System Setup

As according to [8] the design methodology requires that the characters and the items be added to the game in an early stage; therefore a user who is managing the role of an administrator decides how many characters need to be in the game and what items

should be added to it. First of all, the system asks the administrator to add the items to the game and set up their states and actions that can be performed on them.

6.1.1 Item Creation

Every item in an RPG is required for creating new quests and moving the story forward. Every item in this system is composed of a list of initial states and a list of actions that can be performed on it. Since not every action can be done on an item in real world, therefore the item contains a list of all the doable actions. An item can be in one of many states which dictate what action can be done on the item at any given time. Based on the initial states of the item, actions can be performed on it, thus resulting in new states. A good example of this is an electrical switch. The initial state of th switch can be off and you can apply the flip action on it to turn it on. Apart from that the switch serves no other purpose, hence there is no other action that can be performed on it. If the user however wants to create a complicated story, all that needs to be done is to add actions to the item thus increasing the functionality of that item.

To create an item, the *Item Creation* window is used as shown in Figure 6.1. From the drop-down box the user selects the item that needs to be created and then adds the initial states to the item. Once the states have been added, the user selects the actions that can be done on the item one by one and finally presses *Create Item* to add the item to the system.

6.1.2 Character Creation

For character creation, the user needs to select the orientation of the character, which can be good, bad or random as shown in Figure 6.2. Good means that the majority of abilities contained by the character will be good. Bad means that the majority of the abilities contained by the character will be bad and random means that it will contain

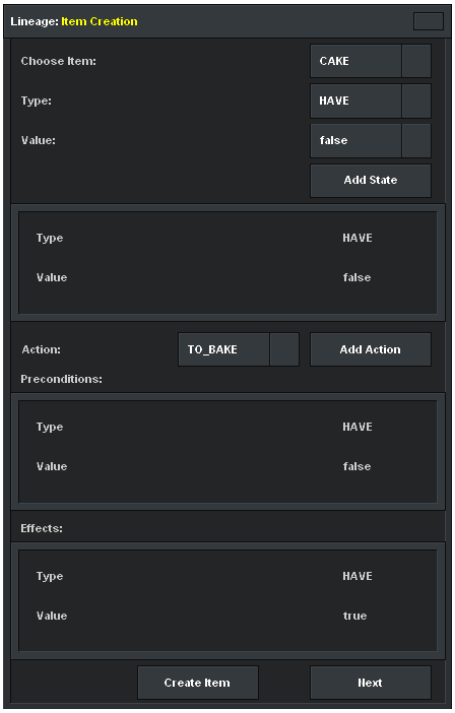

Figure 6.1: Item creation.

a random collection of abilities and it will not be aligned to any one orientation. the second step is to select the item of interest for the character and assign the goal states for that item. In this way, the user can assign the goal states that the character must acquire for every item of interest. Now the character will go through the list of items and request a separate planning graph from that item to reach the goal states. Finally the user can select if the created character is the main character, which means that it will be controlled by the user and then add the character to the system. This way more characters can be added to the system one after the other with different orientations to create different simulation of character behavior.

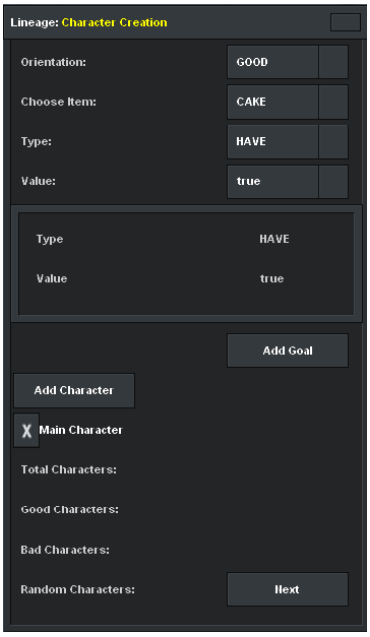

Figure 6.2: Item creation.

6.1.3 Character Finalization

The final part for setting up the system is that for finalizing all the characters. We can see in Figure 6.3 that there are 4 separate windows that perform very distinct operations. The *Abilities* window produces abilities as according to the orientation of the character, selected by the administrator. The administrator can click on *Good*, *Bad* or *Random* to change the characters orientation if needed. The *Physical Attributes* window allows the user to change the physical appearance of the character. The character is produced by the system by taking into account all the past selections made by the administrator. Once the selection has been made, the administrator can add the character as a parent for the next series of creations or can even use one of the parents as the selection for the current character. If however, the administrator wants to use the previous selections to get a new looking character, then the *Population* window can be used to generate the next character. The administrator can select the number of generations that the genetic algorithm should

use to create the next character. The GA uses sigma selection to arrive at the most favorable character. Once the administrator is happy with the character, the Finalize Character button can be pressed to finalize the selection. When all the characters have been finalized, the Simulate button can be pressed to start the game. Now the characters will communicate with the items and try to achieve their objectives.

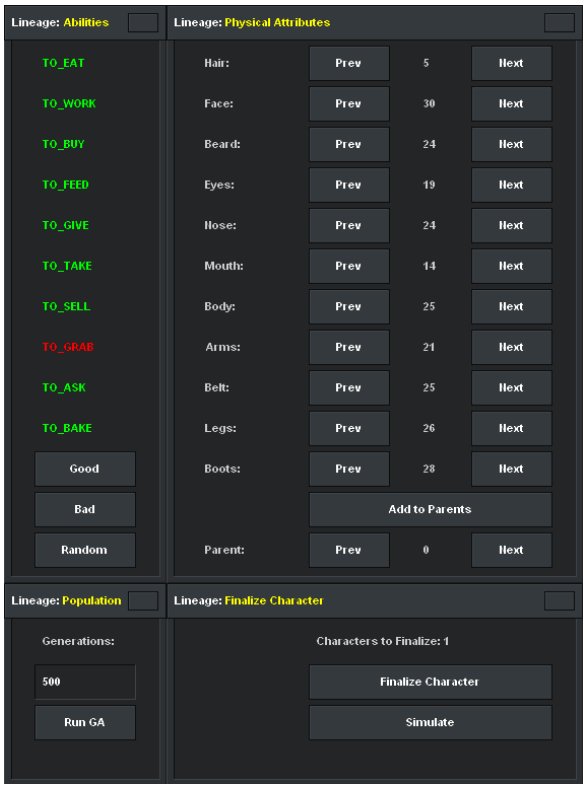

Figure 6.3: Item creation.

6.2 The Working System

Once the simulation starts, the user can pause it at any time by pressing the space bar. the user can click on any of the characters to see the abilities of that character. The user can bring the mouse on any one of the items present in the game and it will show the

initial states of the item, the goal states and the Actions that the item contains as shown in Figure 6.4.

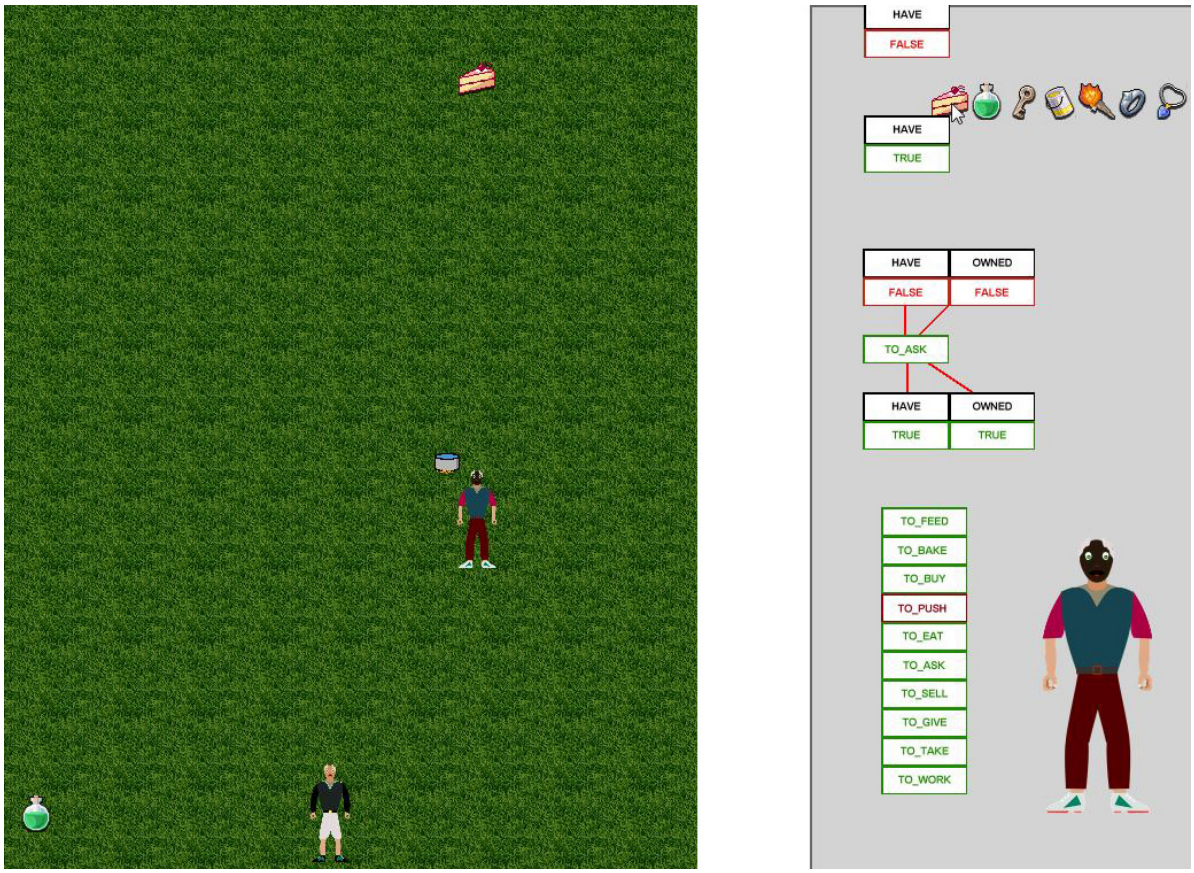

Figure 6.4: Item creation.

At the beginning of the simulation, all the characters and items are spread randomly over the game canvas. The characters then check from their list of items if they have any goal states available. If they have goal states available, they communicate with the item to get an action order list. They then apply the decision making process on the action order list to get the solution. At any time if the states of the item they were pursuing changes, then the characters request the item for a new action order list and the decision making process is applied again to find a new solution. The simulation comes to a close

when all goal states for all the characters have been achieved.

6.3 Evaluations and Results

The software evaluation was distributed into two parts. The first being the character creation aspect of the system and how the users react to it. A total of eight users in out research lab were allowed to play with the system to create their own characters and then they were given a questionnaire to fill out which contained specific questions which ranged from the utility of the system to options that they would like to have in the system.

The first two questions were about selection choices and whether the users would prefer a 2D character or a 3D character. As it turned out six users thought that the number of choices were adequate and they did not need more selection choices. This was mainly because of the huge number of choices that were available to them and it took them a large amount of time to go through all of them. The character creation became a chore rather than an enjoyable activity. This however is subjective to the kind of user that is using the system. If a person is meticulous about every character, then the number of available options will be appreciated to fine tune every character. From a game playing perspective, users want to start a game quickly and character creation might hinder their progress.

The users unanimously chose to create a character in 3D rather than 2D as was provided to them. The system was initially designed with keeping the 3D characters in mind but was later changed to 2D characters due to the difficulty in creating the large number of 3D assets required. The system can however use either 2D or 3D engine as the code is not dependent on it and since we wanted to show the utility of Genetic Algorithm in the creation of a character, making of the 2D assets was found easier. A better system however would be where we can change a single 3D mesh with different numerical values

rather than using a modular character.

For character creation, six users opted to allow the system to choose a predefined character for them rather than creating their own from scratch. They liked this idea because then they only need to do slight modifications to the character rather than creating one from scratch. In our research the system does indeed provide a character option based on the choices made by the users previously. Therefore it is quite likely that the users will select the predefined character rather than making their own. However, when the users were asked if they will be OK with using a character that has not been defined by them, then only half of the users liked this idea. The other half preferred that a character should be made by them.

For character creation, selection of the mental attributes had played a major role in our research. We were of the opinion that users would like to select the mental attributes of their characters. As a result, while creating the characters, the orientation played an important part. Users were asked if they would like to choose how their character behaved and six answered yes. However, users paid little interest in characters which they were not going to control and only two users preferred that they get to choose the mental attributes of the NPCs. All however liked the idea that other characters will compete with their character in completing their objectives. This brings two observations into the limelight. The first being that the users want the other characters to play an active role while playing the game. It also highlighted that users are more concerned about the appearance of their own character than that of the opponent when playing a game. Every user liked the idea that their character should change appearance based on the choices they make throughout the game. As a result, the change of appearance of a character based on his actions can be an interesting research topic in future. This however might also point towards the importance that the users give to their own characters and they

want their choices to leave an impact on the game world.

From this evaluation, it can be gathered that character actions and appearances are both important for the users in an RPG. The users relate to these characters and use them as a vessel to express themselves. A system that provides a high degree of customization will be welcomed by the normal users and a system like this will be appreciated in the world of Role Playing Games.

As explained in section 4.2 that the planning graph is created in polynomial time, therefore graph creation is done quickly enough. Every time a call for the graph creation is made, a data structure is created from scratch to take into account all the states that have changed since the last call for graph creation. Since the environment is dynamic in nature, this call for graph creation can be quite frequent. Once the graph has been created A* algorithm is used to find the shortest path. If we consider that the suggested heuristic by us is optimal, then the time complexity of A* will come out to be O(n).

In order to evaluate the overall benefit of this implementation over the usual method we need to compare the two systems with the same set of parameters. The scope of this evaluation is too immense to arrive at a conclusive result. For example, we will need to create a proper story that is first of all enacted by characters according to a predefined FSM and then we will need to create the same story and let it be performed by characters using a planning graph. Even though one implementation is going to be scripted while the other will be dynamic in nature, the benefit will only be apparent if the given problems are too complex to be handled with an FSM. At the same time, the number of characters present in the story will also be a major factor in deciding which approach is better.

Another point of evaluation is dependent on the type of simulation that is required. Creating a separate plan for every character does indeed create extra overhead which might not be suitable for a simple evaluation. On the other hand , if a complex simulation

is desired, then we need to define complex actions and the possible states that can be reached to produce a meaningful outcome. If the combination of actions and states do not produce the desired goal states then using this system will be meaningless. We have to make sure that effects of one action might lead to prerequisites for another action while defining all the actions.

There are however many factors that suggest that in the long run, as the simulations become more and more complex, this system will become beneficial over the usual implementation. The system works by finding the correlation between the actions that are suggested by the plan with the actions that are contained by a character. There are several steps that reduce this search space therefore the A* algorithm avoids calculating various decision paths which will not lead to a solution. First of all, the marking of all the actions that can not be performed by the character. Secondly, marking of the actions that the character is not willing to perform. Once these actions have been marked the A* algorithm only goes through the actions that are available to the character. Another important point to note is that, the action order list only contains those actions which are relevant to the task at hand. All the other unnecessary actions are not even considered which greatly reduces the number of actions that the A* algorithm has to go through. Finally, the character only starts to perform actions if a valid graph was created in the first place. Therefore the character only starts performing an action if it would lead it to its objective, otherwise it will continue doing one of the mundane actions that had been assigned to it. It will however create another graph because the initial states of the item might have changed and many actions that the character was not willing to do previously might become available the next time.

Chapter 7

Summary and Future Works

This dissertation has described a new methodology by which agents can take decisions in a game environment that is dependent on their mental attributes. This approach allows the agents to evaluate every action against their orientation and then applies their willingness to performing a certain action. As a result, the response that is received by the character is very close to how a real person would behave under similar circumstances.

As an application example, it was also shown how user interaction will influence the overall system and how a certain degree of randomness will be attained by the system. A Role-Playing Games' example was used to exhibit that generally the NPCs in such games do not have well defined objectives and their behavior is mostly scripted. In this system, every NPC is given a series of objective to accomplish and there is an elaborate decision making process that tries to mimic human behavior to help the NPCs exhibit a more natural reaction to the changes in their environment.

7.1 Compendium

Chapter 1 introduced research backgrounds and research purposes. Most of the technical terms and related works this researches were described in this chapter. Chapter 2 explained how the overall topic was conceptualized. From the initial concept of gener-

ating stories through a user's vocal input to using evolutionary algorithms to defining characters and their abilities which also plays an important role in the decision making process for an automated agent. Chapter 3 described the complete method for character creation for this system. The user's opinions play an important role during character development as the user can choose how the character should look and behave and some of these opinions are carried over to the next iteration whenever the game is run again, thus providing a sense of belonging to the user with the characters. In Chapter 4, The formulation of the planning graph is discussed. A planning graph is being used here to define the order in which the actions should be performed by the characters to achieve their objectives. The planning graph overrides the need to depend on FSM for the automation of agents. The graph is created in polynomial time thus making its utility for such a problem feasible. Chapter 5 defines the elaborate decision making process that takes into account the abilities, orientation and willingness of the character to promote a more natural response by an agent when it tries to achieve its objectives. It also explains with an elaborate example the traversal of the A* algorithm with in the generated graph and how the decision making process affects this traversal. Chapter 6 showed the complete working system by bringing all the concepts explained in the previous chapters together in one application. It showed the proof of concept with a working example that simulated the behavior of agent interaction with the items in the environment based on the Planning graph provided to them. The evaluation of the system and experimental results indicate the proposed framework is applicable to Role-Playing Games which can benefit from automated agent responses and interaction. The same research can also be used for individual and crown behavior simulations by exhibiting the emotional response of agents towards different goals.

7.2 Future Works and Prospective Applications

The main concept that initiated this research was the generation of animated stories. Since the scope of that project was too big, it needed to be limited and was done so by focusing on the most important components of the story namely characters and items. The original idea expected the characters to be added to the system but did not set any rules about the behavior of the character. In such a situation, the creator of the story cannot dictate each and every action that the character will take, therefore a certain degree of automation was required. Due to the open nature of the research, the actions that can be performed by the characters were not going to be fixed and defining an FSM was not feasible because the addition or removal of any action resulted in redesigning of the FSM. The planning graph solves this problem by dictating what actions needed to be done and in what order to have a meaningful outcome. However, in order to keep the system manageable, several limitations were put in place. One of these limitations was the amount of actions that were defined for the character. The number of these actions can be increased which will also greatly change the behavior of the characters.

Another area which can be improved is the definition of the orientation of characters. Currently, the number of actions pertaining to a certain orientation decides whether the character is good or bad. This is a workable solution but it needs to be more complex. One way that this can be improved is to change this definition with respect to the environment in which the character resides in. For example, in a war situation many good individuals do not have much inhibitions when it comes to killing other people, which is unacceptable in a normal situation. As a result, the overall behavior of the characters can change which will also impact the overall decision making process. If in future we base the collaboration between the agents based on their orientation, then with the change of environment might result in those agents working together who previously did not do so

because the orientations did not match.

To create the characters physically, Genetic Algorithm is used which takes into account the physical appearance of the parents to produce child population. If, during the running of the simulation we add the functionality that new characters will be created, then we can use the GA to create more characters that will adhere to the choices made by the users earlier. Although crossover and mutation are known as the main genetic operators, it is possible to use other operators such as migration in genetic algorithms which can be used here to show its effects on local population [1].

One of the prospective applications that can use this system is a simulation of every day human behavior and activities. The systems that are currently in place normally aim to solve one problem by studying the human psychological response as exhibited in the crowd walking simulation [46]. Even in this simulation, the emotional response of the agents is not taken into consideration which is an important component of this research. Adding the emotional response to the agent's decision making process will make the simulations more realistic. Since the characters will be providing an emotional response, it will be easier to calculate how the entire population will react to a change in its environment. Since the decision making process goes through a series of steps, a trend in population response can be simulated using this research.

Another very useful application that can be created is the complete breakdown of the objective that a user wants to achieve. For example, if A user plans on watching a movie at a theater, there is no proper way of suggesting how this objective will be achieved. In such an instance the system will evaluate the states for every action that the user will need to perform. For example, in order to watch a movie the user needs to have tickets. To have tickets, the user needs to have money. To have money the user needs to carry a purse. Therefore the system will outline 3 actions for the user. The first will be to carry

a purse, the second will be to buy tickets and the third one will be to watch movie. If at any point an action is not doable, it will also suggest an alternate course of action, making the entire process very convenient.

In short, this research can help immensely in creating planning and scheduling applications that can help in solving many every day problems and help in creating a life like simulation of individual behavior.

Bibliography

[1] Akbari, R., Ziarati, K., A multilevel evolutionary algorithm for optimizing numerical functions. *IJIEC 2: 419430*, 2011.

[2] Albin-Clark, A., and Howard, T., Automatically generating virtual humans using evolutionary algorithms. *In TPCG, Eurographics Association*, W. Tang and J. P. Collomosse, Eds., 6164, 2009.

[3] Arinbjarnar, M., Murder she Programmed: A Dynamic Plot Generating Engine for Murder Mystery Games., University of York, January, 2006.

[4] Aylett, R., Louchart, S., Tychsen A., Hitchens M., Figueiredo R., Managing emergent characterbased narrative. *In proceedings of the INTETAIN 2008 Conference (Playa de Laguna, Mexico)*., 2008.

[5] Bangs, O., Jensen, O. G., Jensen, F. V., Andersen, P. B., and Kocka, T. Non-Linear Interactive Storytelling Using Object-Oriented Bayesian Networks. *Proceedings of the International Conference on Computer Games: Articial Intelligence, Design and Education*, 2004.

[6] Blum, A.L. and Furst M.L., Fast planning through planning graph analysis, *Artificial Intelligence*, 90 (12) , pp. 279298., 1997.

[7] Combs, N., The Intelligence in the MMOG: From Scripts to Stories to Directorial AI. *In Proceedings of the Other Players conference.*, 2004.

[8] Dignum, F. Westra, J. W. van Doesburg A., and M. Harbers., Games and Agents: Designing Intelligent Gameplay, *International Journal of Computer Games Technology*, vol. 2009, Article ID 837095, 18 pages, 2009.

[9] Drachen, A., Hitchens, M., Eladhari, M. P., *Role-Playing Games: The State of Knowledge.*

[10] Drachen, A.; Hitchens, M.; Aylett, R., Louchart, S., Modeling Game Master-based story facilitation in multi-player Role-Playing Games. *In Proceedings of the 2009 AAAI Symposium on Intelligent Narrative Technologies II (Stanford, USA)*, pp. 24-32., 2009.

[11] Egri, L., The Art of Dramatic Writing. *Simon and Schuster.* New York., 1960.

[12] Eladhari, M. and Lindley, C. A., Narrative Structure in Trans-Reality Role-Playing Games: Integrating Story Construction from Live Action, Table Top and Computer-Based Role Playing Games. *In Proceedings of the 2005 DiGRA Conference.*, 2005.

[13] Fairclough, C. R., and Cunningham, P., AI Structuralist Storytelling in Computer Games. *Proceedings of the International Conference on Computer Games: Articial Intelligence, Design and Education*, 2004.

[14] Fairclough, C. R., Story Games and the OPIATE System.*Ph.D. dissertation*, University of Dublin - Trinity College, 2004.

[15] Fikes, R. E. and Nilsson, N. J., STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189208., 1971.

[16] Fine, G. A., Shared fantasy: Role Playing Games as Social Worlds. *University of Chicago Press.*, 2002.

[17] Freytag, G., The Technique of the Drama, 1863. *(Reprinted by Johnston Reprints in 1968.)*

[18] Fujiwara, Y. and Sawai, H., Evolutionary Computation Applied to Mesh Optimization for a 3-D Facial Image, *IEEE Transactions on Evolutionary Computation*, Vol. 3 No. 2, 1999.

[19] Fukutake, H., Akazawa, Y., Okada, Y. and Niijima, K., 3D Object Layout and Composition by Voice Commands Based on Contact Constraints, *International Journal of Computer Graphics and CAD/CAM (CGACC), International Scientific Publishers*, Vol. 1, No. 3, pp. 101-111, ISSN 1817-3489, 2006.

[20] Gazen, B., and Knoblock, C., Combining the ex-pressivity of UCPOP with the efficiency of graphplan. *In Proceedings 4th Euro. Conf. on Planning.* 1997.

[21] Goldberg, D., Genetic Algorithms in Search, Optimization and Machine learning. *Reading, M A: Addison-Wesley*, 1989.

[22] Goldberg, D. E., Deb, K., and Thierens, D., Toward a better understanding of missing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1), 10-16, 1993.

[23] Hitchens, M. and Drachen, A., The many faces of role-playing games. *International Journal of Role-Playing*, vol. 1, no. 1, 3-21, pp. 2008.

[24] Hong, J., Goal Graph Construction and Analysis as a Paradigm for Plan Recognition., *Proceedings of AAAI-2000*, pp. 774779., 2000.

[25] Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y., Extending planing graphs to an ADL subset. *Proceedings of 4th Euro. Conf. on Planning*, 273285. 1997.

[26] Khan, U. A., Okada, Y., 3D Terrain Generation and Texture Manipulation by Voice Input, *Proceedings of the 4th Asian GAME-ON Conference on Simulation and AI in Computer Games (GAME-ON ASIA2012)*, pp.71-75, 2012.

[27] Khan, U. A., Okada, Y., Evolving Story and Character Generation for Role-Playing Games, *Proc of the workshop at SIGGRAPH ASIA 2012*, pp 59-64, Nov, 26-27, 2012.

[28] Khan, U. A., Okada, Y., Character Generation using Interactive Genetic Algorithm, *Proc. of the GameOn 2013*, pp. 31-35, November 25-27, 2013.

[29] Khan, U. A., Okada, Y., Planning Graph with Character Orientation for Decision Making of Non-Playable Characters in a Role-Playing Game, *Proc. of the 7th Int. Conf. on Game and Entertainment Technologies (GET2014)*, pp. 165-172, July 15-17, 2014.

[30] Khan, U. A., Okada, Y., Emotional Decision Making response of Non-Playable Characters in a Role-Playing Game, *IADIS International Journal On Computer Science and Information Systems, ISSN: 1646-3692*, Vol.9 Issue 2, pp. 53-66, 2014.

[31] King, B. and Borland, J., Dungeons and Dreamers. The Rise of Computer Game Culture from Geek to Chic. *McGraw-Hill/Osborne*, California, 2003.

[32] Lankoski, P., Bjork, S., Character-Driven Game Design: Characters, Conflict, and Gameplay. *In GDTW 2008 Conference*, Liverpool, UK., 2008.

[33] Lebowitz, L., Creating characters in a story-telling universe, *Poetics*, Volume 13, Issue 3, Pages 171-194, June 1984.

[34] Louchart, S. and Aylett, R., Solving the narrative paradox in VEs lessons from RPGs. *In IVA 2003 Proceedings*, 244-2410., 2003.

[35] Louchart, S., Aylett, R., From synthetic characters to virtual actors. *In Proceedings, 3rd Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE 2007)*, 88-91. Marina del Ray, Mexico: AAAI Press., 2007.

[36] Mackay, D., The Fantasy Role-Playing Game: A New Performing Art. *McFarland and Company.*, 2001.

[37] Mateas, M. and Stern, A., Towards Integrating Plot and Character for Interactive Drama - Working notes of the Social Intelligent Agents: The Human in the loop. *In Proceedings of the AAAI Fall Symposia.*, 2000.

[38] Matthews, J., Basic A* Pathfinding Made Simple, *in Rabin S, AI Game Programming Wisdom*, Hingham, Massachusetts:Charles River Media, pp 105-113., 2002.

[39] Miller, B. L. and Goldberg, D. E., Genetic algorithms, tournament selection, and the effects of noise, *IlliGAL Report 95006*, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois, USA, July 1995.

[40] Okada, Y., Etou, H., Niijima, K., Intuitive Interfaces for Motion Generation and Search, *Intuitive Human Interfaces for Organizing and Accessing Intellectual Assets, Lecture Notes in Computer Science*, Volume 3359, pp. 49-67, 2005.

[41] Orkin, J., Applying goal-oriented action planning to games, *AI Game Programming Wisdom 2*, Charles River Media, Brookline, Mass, USA., 2003.

[42] Orkin, J., Three states and a plan: the AI of F.E.A.R., *Proceedings of the Game Developers Conference (GDC '06)*, San Jose, Calif, USA., 2006.

[43] Pollack, M. E. and Horty., There's more to life than making plans: plan management in dynamic, multiagent environments, *AI Magazine*, vol. 20, no. 4, pp. 7183., 1999.

[44] MacNamee, B., Proactive Persistent Agents: Using Situational Intelligence to Create Support Characters in Character-centric computer games. *PhD. Thesis*, Computer Science Sept., TCD 2004.

[45] Propp, V., Morphology of Folktale. *University of Texas Press*, 1968.

[46] Rymill, S. J., Dodgson, N. A., A psychologically-based simulation of human behavior. *In Theory and practice of computer graphics* (pp. 35-42)., June, 2005.

[47] Schneider, O., Braun, N., and Habinger, G., Story lining Suspense: An Authoring Environment for Structuring Non-Linear Interactive Narratives. *Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2003.

[48] Schwab, B., AI Game Engine Programming, *2e. Charles River Media, Inc.*, Rockland, MA, USA, 2009.

[49] Tychsen, A., Hitchens, M., Brolund, T. and Kavakli, M., The Game Master. *In Proceedings of the 2005 Interactive Entertainment Conference*, 2005.

[50] Tychsen, A., Tales for the Many: Process and Authorial Control in Multi-Player Role-Playing Games. *In Proceedings of the 2008 Interactive Storytelling Conference.* Springer LNCS 5334, 309-320., 2008 .

[51] Ventrella, J., Avatar Physics and Genetics. *In Proceedings of the Second International Conference on Virtual Worlds (VW '00).* Jean-Claude Heudin (Ed.). Springer-Verlag, London, UK, 107-118, 2000.

[52] Watts, N., Teach Yourself Writing A Novel. *Hodder headline Plc.*, London., 1996.

[53] Young, J .M., Theory 101: The Impossible Thing Before Breakfast.*Places to Go, People to Be*. 27., 2005.

[54] Zavala, M., Luga, H., Larios, V., Virtual Reality Advanced Editor For Crowded Worlds Simulations. *International Conference on Computer Graphics and Artificial Intelligence*, Athens Greece, 28-29 May 2010.