Creating an Agent-Based Framework for Don't Starve Together

Fábio Almeida
Instituto Superior Técnico
Lisboa, Portugal
fabio.vieira.almeida@tecnico.ulisboa.pt

Rui Prada
Instituto Superior Técnico
Lisboa, Portugal
rui.prada@tecnico.ulisboa.pt

Samuel Mascarenhas
Instituto Superior Técnico
Lisboa, Portugal
samuel.mascarenhas@tecnico.ulisboa.pt

ABSTRACT

Today's video games are striving to maintain high levels of fidelity. The realistic graphical representation of virtual worlds is only betrayed by the lack of believability that in-game characters present. To maintain the immersion created by exquisite graphics, characters must be able to create the illusion of life, which requires them to possess basic human traits like social awareness, reactivity, and active goal pursuit. Some game genres, like role playing games, have seen this problem being addressed by using agency based characters. However, survival games have not been subject to the same attention. In this work, we address this issue by proposing a framework that allows developers to create characters based on agency models for survival games. By making use of FAtiMA Toolkit, a fully fledged model for agency, and Don't Starve Together, a popular survival game, we've implemented and published such a framework with an example character, Walter. Walter has been run and tested against a behaviour tree based agent.

KEYWORDS

Artificial Intelligence, Non Playable Characters , Believable Characters, Survival Games

1 INTRODUCTION

In recent years, the continuous improvement and development in computer graphics has pushed video games to new levels of graphical fidelity. Graphical representations of virtual worlds have become increasingly more realistic, allowing players to experience new levels of immersion [5]. Additionally, with the recent boom in virtual reality systems, players have never been so physically immersed in a game's virtual world.

The increases on fidelity and lifelikeness of the virtual world cause players to create expectations on the interactions they can have on the world. This expectation applies not only to interactions with the virtual world itself, but is also extended to the interactions with the characters that compose the world, typically called Non Playable Characters (NPCs).

Computer and player controlled characters need to act in a believable manner so that the illusion of reality created by exquisite graphics and physics, the player's immersion, is not broken [28]. The lack of believability in NPCs can break the player's immersion and can provide a bad gameplay experience, given that the game's flow and player's immersion are key elements to the gameplay experience [16]. Reactivity, goals, emotions, and social competence are necessary traits NPCs need in order to be believable [4].

Some game genres, like Role Playing Games (RPGs), which are heavily dependent on player to NPC interaction, have received special attention on the creation of believable characters. However, other game genres, like survival games, have not received such attention.

Survival games, a genre that has seen a rise in popularity recently, are often based in open world scenarios that can be procedurally generated. Either single or multi-player, some survival games revolve around resource collecting, crafting, and management of the character's needs (hunger and thirst, mostly), while others are set in supernatural worlds with some elements of horror: monster infested worlds where players have to survive by defeating them.

More often than not, survival games lack a specific objective other than surviving in a harsh world for as long as possible. However, survival games are trending among players and game developers alike, seeing great attention in the indie game scene.

Don't Starve [9], for example, is a single player survival game set on a procedurally generated world where players have to collect, craft, explore, and fight to survive. With a day and night cycle, Don't Starve provides an interesting gameplay by combining mechanics for temperature, wetness, hunger, health, and sanity. The introduction of sanity to the game's character provides a novelty factor among its peers, giving the character a human-like trait appropriate for a survival game, representing the strain put on the character by having to survive the harshness of a world with nothing to start with.

Following the trend for multiplayer survival games, Don't Starve has also been released as a multiplayer game under the name of Don't Starve Together (DST) [10]. The game is

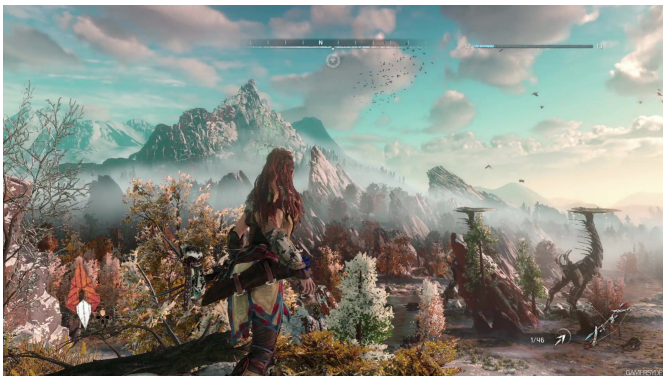

Figure 1: Horizon Zero Dawn exquisite graphics.

identical to its predecessor, though it allows several players to join the same world. The multiplayer nature of the game brings new challenges to players that can either cooperate or compete for survival. In extreme cases, players can even attack and kill each other giving the game an extra challenge to overcome.

However, the game counts with no NPCs to accompany players in their journey for survival, making players dependent on each other for gameplay. The few autonomous non-aggressive characters of Don't Starve Together present limited behaviour and are not subject to the same rules as the players (hunger, health, sanity, etc.).

The nonexistence of interesting and robust Artificial Intelligence (AI) for characters is not exclusive to Don't Starve Together. Other survival games suffer from the same problem: the lack of collaborative AI for NPCs capable of creating believable characters.

As said before, on other game genres like RPGs, tools exist and have been used for the creation of AI controlled NPCs that take into account the character's believability [14][1][12]. Usually empowered by social models, these tools have not yet been applied to survival games.

Many modern day video games are dependent on player to NPC interaction, and while some degree of independent decision making is implemented through the use of AI techniques, it's mostly based on combat and has no social concern whatsoever. This lack of social ability in NPC can badly impact their believability and in turn affect the player's gaming experience.

While some work has been done to tackle the problem of believability in NPC, survival games however, have not been subject of this effort.

2 GAMES, IMMERSION AND BELIEVABILITY

The increasing lifelikeness in video games gives players increasing expectations of better, meaningful experiences and interactions. The world around the player is presented with exquisite graphics and increasing levels of fidelity, as proved by the recent title from Guerrilla Games, Horizon Zero Dawn [13] (Figure 1).

However, NPCs still have a lot of room for improvement. Scripted behaviour and repetitive dialog, products of the lack of socially deep NPCs, can easily break the gaming experience of the player, due to reduced believability in the NPCs actions [28]. This contrast between the world and it's inhabitants may break the suspension of disbelief, one of the major factors needed to achieve a successful gaming experience [16].

The concept of believable characters has long been studied and explored in many different forms of art and media [4]. From the original Disney animators of the 1930's and the book later published on Disney's animations [27], we can understand that in order to have believable characters we need to create the illusion of life.

NPCs in games

Most Triple-A video games make use of simple techniques to express NPC behaviour. Finite State Machines (FSM), hierarchical FSM and behaviors trees are the most commonly used techniques in nowadays industry [29]. However simple, they can achieve believable and expressive behaviours in the hands of skilled game designers. These techniques can express logic, basic planning and reactive behaviours.

Another advantage is that these techniques are deterministic, in other words, we can predict what can happen and then try to avoid problems. Their major weakness, however, is their limited expressiveness, which can cause the realization of complex behaviors unmanageable.

In games like Assassin's Creed Unity [20] and Dragon Age Inquisition [8], where there is a large number of NPCs, there is a tendency to make them reactive to the players instead of actively following their own goals. In Dragon Age Inquisiton, many NPCs are present throughout the world doing absolutely nothing. Sometimes, when the player approaches, she can overhear a conversation that is taking place between NPCs, usually directly related to the player, but these are always simple one or two sentences conversations and the player cannot interact with the NPC that is talking.

One can argue that, using current technology, it is impossible to make every single NPC socially aware and able to interact with the player. However, even if we only look at the NPCs with which we can interact, their behaviour is repetitive and lacks depth.

Other games, such as The Elder Scrolls V: Skyrim [25], have NPCs with occupations which they carry out during the day (the blacksmith will spend her day at the forge working), and houses where they return to during the night (almost every character will go to his home to have a meal and sleep). These simple behaviours highly contribute to the character's

believability as it provides the sense of having wants and needs. However, they still lack social ability.

If, for example, the player breaks into the house of a random NPCs, wakes her up and has a talk with her, she will ignore the fact that you just woke her up in the middle of the night in her locked house and proceed to have a completely normal conversation with the player. The weirder part still, is that the player will be branded as a criminal and may face the city's guards upon exiting the house. This lack of social ability can easily break the player's suspension of disbelief.

To address this problem we can make use of research based agency models for NPC control, as has been done in other works [15][12][11][19][26][18].

3 FATIMA

Fearnot AffecTIve Mind Architecture (FAtiMA) is an agent architecture with planning capabilities designed to use emotions and personality to influence the agent's behaviour [7], and was developed to endow the characters in the serious game *FearNot!* [3] with social intelligence. It has been used successfully in different social scenarios ([22],[23], [2], and [6]) and it is based on appraisal theory [1].

FAtiMA has a modular architecture where functionalities and processes are divided into modular independent assets. This enables developers to use a lighter and simpler version of FAtiMA by adding the required assets, according to their necessities. Therefore, behaviour and functionality are added by including assets in an agent's definition. These assets will then implement the desired behaviours and functionalities.

FAtiMA comes with a set of predefined assets that can be used to define agents and their interactions[2]. Additionally, the developer can create its own assets that extend the toolkit's capabilities. The available assets are:

Role Play Character Asset
The Role Playing Character (RPC) Asset is an aggregation asset responsible for managing all the other assets that compose an agent. It is also responsible for maintaining the agent's Emotional State and the agent's memory, which is divided in two components: a Knowledge Base and an Autobiographic Memory. The Emotional State stores the current emotional state of the agent, defined by the Emotional Appraisal Asset, the Knowledge base stores the agent's beliefs, and the Autobiographic Memory stores the agent's recollections of past events and the emotions associated with such events.

Emotional Appraisal Asset
This asset appraises each event according to the OCC Theory of Emotions [21]. Developer defined appraisal rules, determine how an agent evaluates certain events, determining the agent's Emotional State (which is kept by the RPC).

Emotional Decision Making Asset
The Emotional Decision Making Asset provides RPCs with decision making capabilities based on rules with logical conditions. Allowing the agent to make multiple decisions at the same time, this asset makes use of the Knowledge Base and the Emotional State to decide which actions it decides to do. FAtiMA handles this by making use of what it calls layers of decisions. When the decision process is triggered, if a layer is provided, only actions for that layer are considered.

Integrated Authoring Tool Asset
This asset adds a Dialogue Manager that combines an hybrid solution between dialogue trees (a popular solution among game developers) and state machines to construct dialogues for agents. The Dialogue Manager keeps the current state of dialogue, giving an agent (or player) every available option to choose from.

Social Importance Asset
The Social Importance Asset implements the *SI* Dynamic Property, which represents the social relationship among two social entities A and B. This representation, based on the status-power theory [17], gives information about how willing an agent is of complying with another agent's wishes.

Comme il Faut Asset
Based in the work described below, this asset allows developers to define Social Exchanges which, unlike single dialogue acts, are composed by sequential steps: initiate, answer, and finalize. These steps represent an interaction between two agents.

World Model Asset
The World Model Asset gives developers a way to test the created agent in a simulated world where the consequences of each action is defined by the developer. This asset is especially useful to test particular aspects of an agent.

It is important to note that the Comme il Faut (CiF) Asset can be used to complement the Social Importance Asset by giving an agent knowledge about social practices. For example, while a boyfriend and father might have equally high social importance values, flirting is not an appropriate social practice when the target is the agent's father. By defining flirting as a social exchange, the developer can define an additional condition to only activate this social exchange if the agent is attracted to the target.

[1]The theory that emotions are elicited by evaluations (appraisals) of events and situations [24].

[2]The toolkit is available at https://github.com/GAIPS-INESC-ID/FAtiMA-Toolkit.

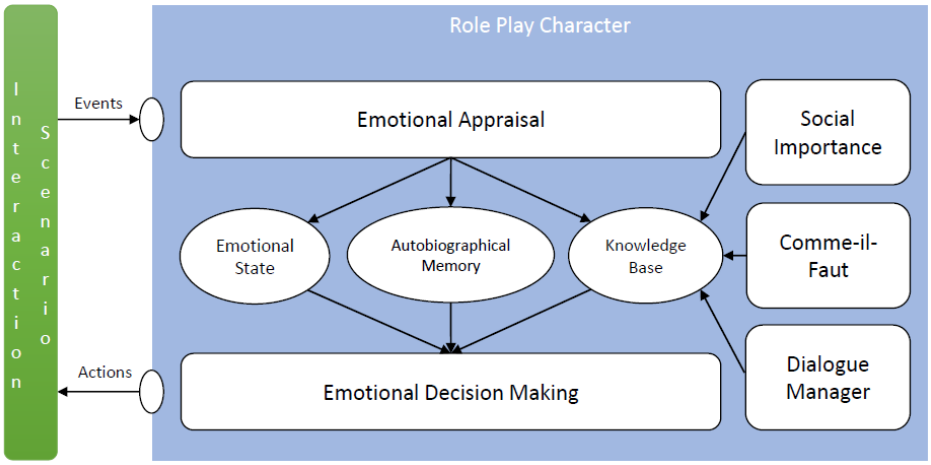

Figure 2: The Role Play Character Asset aggregating all other assets to make an autonomous agent.

4 DON'T STARVE TOGETHER

Don't Starve Together is a multiplayer wilderness survival game developed by Klei Entertainment, where the players must survive as long as they can. The players must face the harshnesses of a procedurally generated world that is actively trying to kill them, either by cooperating or competing with each other. Each game can last an indefinite amount of time which is only determined by the players' ability to survive: the better you play the game, the longer you can last.

The game presents mechanics for hunger, health, temperature, wetness, day and night cycles, seasons, and sanity. It also provides players with a monster infested world that try to kill the player's character. The addition of multiplayer brings the possibility for social interaction among players, which contributes positively for the gameplay experience.

Moreover, the sanity mechanic itself is a differentiating factor from other games in the genre. By providing a measurement of the characters' mental health, the game makes the characters more human and relatable. A normal human being under such harsh environments would suffer physically but also mentally, and the sanity mechanic depicts just that.

For these reasons, the lack of AI controlled NPCs with believable behaviour, the ability to control and personalize the generated game world, and the possibility of introducing modifications into the game, we've decided to use this game as the target of our project.

The Game

A player's main concern is to keep the character's stats balanced throughout the game and find ways to survive the dangers each season brings. To keep a steady supply of food and protection from the elements is as important as defending themselves from the monsters that will try to kill them.

Bellow is a list of the core game mechanics with a small description. This list represents a compilation of key aspects we have to take into account while making a NPC for DST.

Health
When it reaches zero a character dies. It can be replenished by certain items and by eating appropriate food and will be lost while fighting and when in extreme conditions of temperature.

Hunger
Will cause a character to start losing health when it reaches zero. It will gradually decrease over time and can be increased by eating food.

Sanity
Represents the mental state of the character. Certain conditions will cause it to decrease (e.g. fighting or standing in the dark) while others can cause it to increase (e.g. eating certain foods or being around befriended *pigmen*). When near zero will provoke hallucinations that can attack the character.

Temperature
In cases of extreme cold or heat the character will start to lose health over time. It can be kept in check with the use of craftable items like clothes and fires.

Wetness
As the character gets herself wet it can drop held items, food in the inventory will spoil faster, and her temperature will also decrease. The use of umbrellas and appropriate clothing will prevent the wetness level from rising.

Darkness
When in complete darkness a character will be attacked by the darkness creature. It is important to stay near a source of light during the night.

Fighting
Either as a means of defense or of gathering food, fighting is a core game mechanic without which survival is impossible.

Inner Workings

Everything that exists in the world of Don't Starve Together is represented as an Entity, from the character the player controls, to the sounds the player hears. These, in turn, are instances of *prefabs* that can be accessed and edited in the Lua scripts used to make the *mods*. It is important to note that each entity is uniquely identified by a six digit Globally Unique Identifier (GUID).

By specifying and configuring its *components*, *stategraphs*, and *brains*, the *prefabs* define every entity in the game. Each of these parts represent something different: *components* represent what an entity does; *stategraphs* represent how it looks; and *brains* represent how it behaves.

When a *prefab* is defined, it also configures its *components*, for example, in the definition of the *prefab* "wood", which contains the *component* "fuel", there is also the configuration of this *component* that, in comparison to the *prefab* "charcoal", has a lower level of "fuel". Effectively, *components* describe functionalities that a given entity can have, and how that functionality works. When an entity, either by a player's command or the *brain*'s command, does a chop action, it is the *components* of the held axe that determine how this action will be executed and trigger the appropriate animations and events.

While *components* can be reused throughout different *prefabs* (with different configurations), *brains* are defined on a per *prefab* basis. Both butterflies and spiders share the "locomotor" *component*, but have distinct moving behaviours.

When defining the *prefab*, if it is meant to have behaviour, an instance of a *brain* is also defined and attached to the *prefab*. However, not all *prefabs* have autonomous behaviour, and those that don't, don't have a *brain*. The *brain* itself is a Behaviour Tree that is attached to an entity, telling it what to do, leaving the part of how to do it up to the entity's *components*.

Using the Lua scripts, a *modder* can create *components*, *stategraphs*, and *brains*. These can then be attached to existing *prefabs* or new ones, also created by the *modders*.

5 PUTTING IT TOGETHER

Now that we have explored both the game used as a scenario and the technology used to control the characters, it is time to look at how we connected both. For the rest of this section, we'll present the implementation of the framework, explaining in full detail the decisions made.

The implementation itself consists in two modules: FAtiMA-DST, a Don't Starve Together *mod*; and FAtiMA-Server, a C# console application. Both these modules have been made publicly available on the Github repository https://github.com/hineios/FAtiMA-DST. In the repository there is also a guide to help anyone develop an NPC for Don't Starve Together. As shown in Figure 3, the communication between the two modules is made using Hypertext Transfer Protocol (HTTP), which transfers JSON objects back and forth.

The choice of using HTTP communication arouse from a limitation we found while developing the initial proof of concept for this implementation. Initially, we considered importing the FAtiMA Toolkit directly, as the loading of C# Domain Specific Language (DLL) files into a Lua interpreter is possible. However, as a security measure, the Lua interpreter embedded in DST blocks any importation of external libraries. The inclusion of malicious libraries through the Steam Workshop would be a major security issue for both the company and the players.

As alternatives, we considered using textual files or a relational database that would act as a proxy between the modules, but we finally decided to use an HTTP connection as the game provided an easy way to make requests and register callback functions to handle the responses. Due to the client-server nature of HTTP however, the FAtiMA-DST module continuously makes HTTP requests with JSON encoded data to the FAtiMA-Server module, which handles these requests yielding an appropriate response (also JSON encoded). As a data-interchange format, JSON provides the necessary abstraction to exchange information between C# objects and Lua tables.

FAtiMA-Server

The FAtiMA Toolkit is written in C# and published as a set of DLL libraries. Taking this into account, we decided to implement a simple server in C#, capable of handling the HTTP requests made by FAtiMA-DST *mod*. This server contains a list of Role Play Character Assets that are uniquely linked with the agents running on the scenario. Each request the server handles has an entity's (NPC) GUID, which is used to also identify a Role Play Character Asset.

Additionally, requests are divided into three categories: perceptions, events, and decisions. Perception requests contain information on the NPC's state and field of vision, and happen periodically. Event requests contain information on relevant world events, and happen whenever an event is triggered. Decision requests trigger the decision process that tells the NPC what to do, and happen periodically.

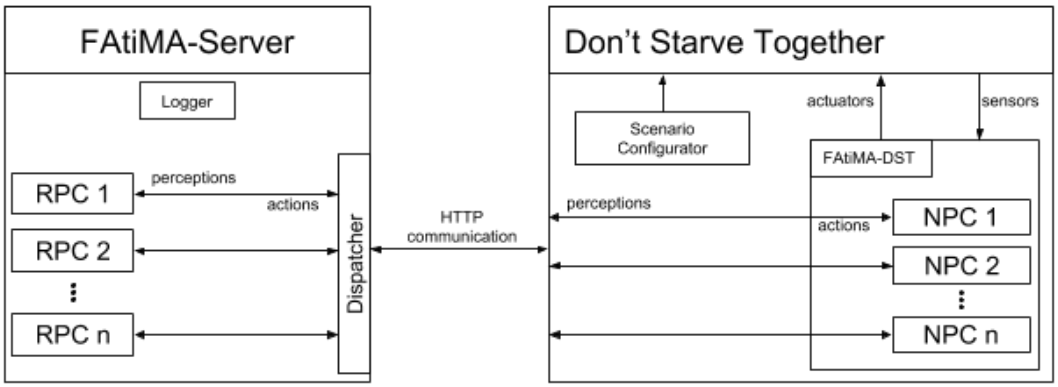

Figure 3: A graphical representation of the implementation.

If we consider the flow of information, the first two types (perception and event requests) represent a flow of information from the embodiment to the AI, while the third (decision requests) represents a flow of information from the AI to the embodiment.

When received, perception requests are translated into a series of Property-Changed events, that update the RPC's Knowledge Base. Event requests will be translated to Action-End events and decision requests will trigger the RPC's decision process for the appropriate layer.

It is important to notice the limitations of using HTTP as the method of communication. This form of communication makes it impossible for the AI to give direct actions to the embodiment, as we cannot send messages from the server to the client, only respond to client's requests. As such, we have to rely on periodic requests made by the embodiment that will cause the AI to make a decision and respond with an action.

FAtiMA-DST

his module was implemented as a *brain* for DST and is responsible to send perceptions to FAtiMA-Server and execute the actions it returns. To achieve this, it sends a perception request to FAtiMA-Server two times a second, a decision request for behaviour every one and a half second, and a decision request for dialogue every ten seconds. The values were tested and tweaked for the best compromise between behaviour and performance.

To send the perceptions to the FAtiMA-Server, it counts with a set of helper functions that extract the necessary information from surrounding entities. A set of listeners relay information about events to the FAtiMA-Server, and a Behaviour Tree executes actions received from the FAtiMA-Server.

The Behaviour Tree makes use of the built-in Buffered Actions to execute the actions. This provides a layer of abstraction over actions to handle all the locomotion details and appropriate verifications (e.g. it is only possible to chop a tree if an axe is held).

To handle the FAtiMA-Server commands, the tree contains two nodes: a node that executes actions (via buffered actions), and a node that handles the special behaviour of wandering. The handling of dialogue is independent from the behaviour and is executed immediately without making use of the behaviour tree.

6 CREATING WALTER

As an example NPC, we've implemented a model based agent to demonstrate how one could use the set of available beliefs to create behaviour. This example only makes use of FAtiMA's Dialogue Manager and Emotional Decision Making assets.

This example has been published to the Steam Workshop in the form of an AI companion, an NPC named Walter [3]. On the *mod*'s public page, a set of instructions can be found on how to run the character. The page also contains links to the public repository where all the code for this framework can be found.

Currently, the framework counts with a total of forty eight beliefs and over fifty actions[4]. By making use of the beliefs defined in the framework, we created a set of seventeen rules that give Walter its behaviour.

[3]The *mod* public page can be found in this link: http://steamcommunity.com/sharedfiles/filedetails/?id=1339264854

[4]The lists of beliefs and actions can be consulted in https://github.com/hineios/FAtiMA-DST

Figure 4: Example of speak action.

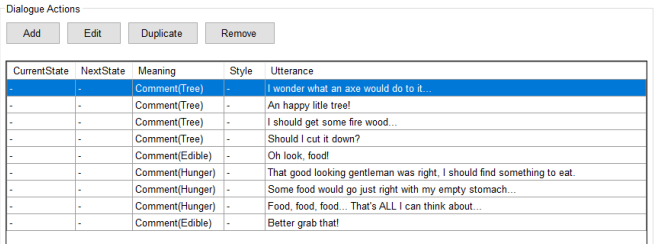

Figure 5: Walter's available dialogues.

Dialogue

As far as speech goes, the current framework has some limitations. Currently, there is no two-way communication, meaning that, although the agent can speak (through the use of text), the players will not be able to respond. In Figure 4 we can see the execution of a speak action by the agent while he is executing another action.

To make an NPC speak, the developer will need to create the available utterances using the Dialogue Editor as depicted in Figure 5. Multiple utterances can be created with the same meaning, allowing for variety in the NPC's dialogue. In Walter's example, several utterances have been added for each meaning.

To trigger these sentences, the developer also needs to add a rule with the appropriate layer to the decision making asset, as shown in Figure 6. Here we can see, that in order for Walter to say a comment about hunger, his `Hunger` belief has to have a value lower than seventy.

The addition of these sentences allows players to better understand the NPC's decision process. By stating some random utterances about its current state, we can transmit the characters state of mind to the players. These sentences can also be used to transmit the NPC emotional state, needs, and goals.

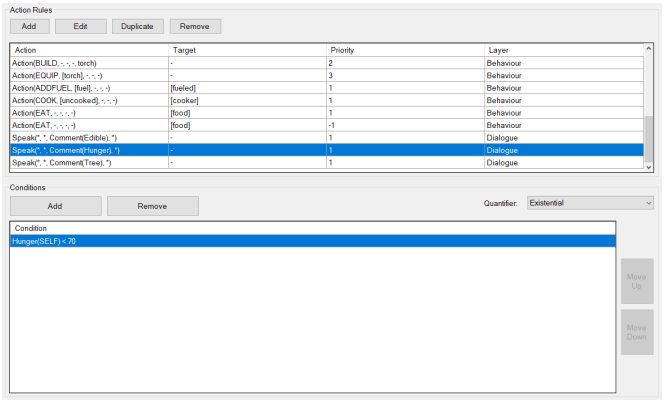

Figure 6: Walter's dialogue action rule.

Figure 7: Configuration screen for FAtiMA-DST.

Playing with Walter

In order to run our framework, players will have to run FAtiMA-Server by launching the console application available at the public repository. FAtiMA-Server will automatically listen to HTTP requests on port 8080. Then, before launching a game, the player will need to activate and configure the *mod* as shown in Figure 7. When the game is launched, the number of characters specified will spawn in the same place that players spawn.

In Figure 8, we can see an example of a player playing the game with two NPCs being controlled by FAtiMA.

7 EVALUATION

We've successfully implemented a framework that enables developers to use an agency model, FAtiMA, to create NPCs for Don't Starve Together. This was achieved through the implementation of a modification for DST and a companion console application. As contributions resultant from this work we also count with a tutorial for creating NPCs and an example NPC.

Figure 8: A player (center) playing the *mod* with two FAtiMA controlled characters (top and bottom).

Walter - The AI Companion

The example character we created to demonstrate the framework has been published in the Steam Workshop. At the time of writing, with 48 days of existence, the *mod* counts with 738 unique viewers and 128 subscribers.

While subscribers represent the number of people who've added the *mod* to their game, it does not tells us how many of those have actually used the *mod*. The *mod* also counts with 25 comments on the public page, mostly related to running the mod.

Although it has received no direct negative feedback, we believe that the necessity of running a separate console application has greatly impacted the community's acceptance and use of the *mod*. Additionally, to the best of our knowledge, no member of the community has used our framework to develop their own NPCs.

Monte Carlo Tree Search Project

As part of the Artificial Intelligence for Games course taught at Instituto Superior Técnico, a group of three students used our framework for their final project. The project consisted in the implementation of the MCTS algorithm for the decision process of the agent.

Briefly, the Monte Carlo Tree Search (MCTS) algorithm is based in a random exploration of a state space. By randomly (and rapidly) collecting data of the space state, the algorithm will gradually improve its knowledge of what the best next move is.

Unlike many other algorithms, MCTS does not run until an answer is found. Instead, the usual approach consists in letting the algorithm run for some time (or a certain number of iterations) and then return the best possible action, in regard to the information collected so far. The algorithm has proven to be especially good in problems where the space state has an high branching factor and many possible combinations, such as the Go game.

In this case, the implementation of the MCTS algorithm was achieved through the use of FAtiMA's Dynamic Properties. The created Dynamic Property, called `MCTS`, will use current knowledge base and apply the MCTS algorithm. The MCTS algorithm was let run for two thousand iterations (about twenty seconds) before returning a plan to act upon.

One of the main difficulties found during this use case was related to the random exploration of the state space. As Don't Starve Together does not provide a way to simulate the resulting state of applying actions and FAtiMA did not have such functionality either (although it can now achieve this through the use of the World Model Asset), the group had to create their own simulator of the Don't Starve Together world.

This work, praises the versatility of our framework by demonstrating the ability to create a planning agent based on MCTS. Although the group had to struggle with the world simulation, the end result was an NPC for DST with planning capabilities.

Walter vs. Artificial Wilson

As a means of evaluating the created NPC, we've run it and recorded how long it can survive. Additionally, we compare it to an unpublished character for Don't Starve, Artificial Wilson.

Artificial Wilson, is the work of the anonymous *modder* KingofTown and can be found in the public repository https://github.com/KingofTown/DS-AI.

Testing Conditions. For each run of the characters, a new world was dynamically generated with equal sets of constraints for both characters. The characters were then let run for as long as possible until their death.

As we said, Artificial Wilson was created for the original game, Don't Starve. Therefore it is incompatible with Don't Starve Together because of the updates made to turn it into a multiplayer game. However, the game mechanics and configurations remain the same.

As such, we can use the scenario configurator to generate equivalent worlds. For this test, we considered the following restrains:

- **A world with no enemies**: all aggressive monsters have been removed from the game as well as the giants.
- **No random meteorological events**: apart from raining all other events have been disabled (like meteors, frog rains, lightning storms, and wildfires).
- **No resurrection stones**: if the NPC dies, it stays dead.
- **No matting season for *beefalos***: during this season these, normally pacific creatures, will attack any character on sight.
- **No modifications**: apart from the necessary ones for running both characters.

Figure 9: Days survived for Walter.

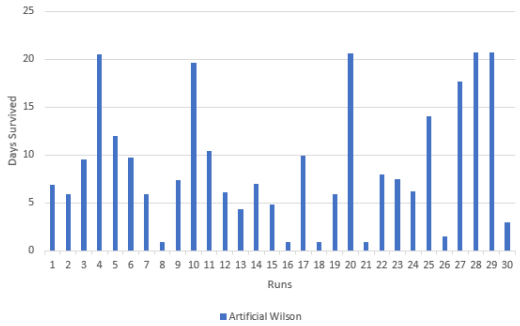

Figure 10: Days survived for Artificial Wilson.

- **Only one NPC and no players**: while in Don't Starve this does not apply, in Don't Starve Together the tests were run locally with no players present in the world.

These constrains will generate similar worlds for both Don't Starve and Don't Stave Together. For each play-through of an agent, a new world is generated with these sets of constraints. Each generated world is unique but complies to the specified set of constraints.

The Results. Each character has been run thirty times and the survival duration for each character is presented in Figures 9 and 10.

Walter survived, on average, three days while Artificial Wilson survives, on average, eight point nine days. The percentage difference of 99% between both values can be explained by the detail put in each character's decision process[5].

Walter is a model based agent with nineteen rules of decision. Each of these rules has no decision process besides the character's ability to perform it (e.g. does the NPC has an axe equipped to cut down a tree?, does it have the required ingredients to build a fire?).

[5]Percentage difference equals the absolute value of the change in value, divided by the average of the 2 numbers, all multiplied by 100. We then append the percent sign, %, to designate the % difference.

Artifical Wilson is based on the built-in behaviour trees. The final behaviour tree counts with over seventy nodes and dozens of helper functions. Embedded in the NPC's decision process are some planning capabilities and resource management (e.g. the NPC will ignore resources if she has already gathered enough of it), which Walter does not possess.

If we compare both characters to real human players, Walter has the ability of a newcomer who as never played the game, while Artificial Wilson could be compared to a returning player who played the game for a couple dozen of times[6].

For each character we've recorded not only their survival time but also their cause of death. In Table 1 you can see a summary for both characters.

Table 1: Artificial Wilson and Walter cause of death summary.

Artificial Wilson		Walter	
CoD	Times	CoD	Times
Cactus	2	Darkness	17
Cold	5	Hunger	13
Darkness	10		
Hunger	13		

8 CONCLUSIONS

The use of Behaviour Trees and Finite State Machines, the most commonly used techniques in today's industry, provide skilled game designers with the ability to create somewhat complex behaviour. However, these techniques can quickly become an authoring burden. Having to manage hundreds of behaviour nodes or states is no simple task. Furthermore, grasping the context of social interaction will often be left out of the NPC's behaviour.

Focusing on survival games, we've noticed how these games will often lack the presence of NPCs, and when they are present, they can break the player's immersion by not being subject to the same rules as the players. This, allied to the fact that many survival games rely on multiplayer interaction in order to keep interesting, has motivated us to create a platform that allows developers to create believable NPCs with interesting behaviour for survival games.

In order to make believable NPCs we need them to possess certain human-like traits: social awareness, active goal pursuit, and reactivity, for example. By incorporating FAtiMA's capabilities into a commercial survival game we've been able to realize a platform that allows developers to create NPCs with social ability and planning capabilities.

[6]This comparison has been based on the writers' experience with the game

We've chosen Don't Starve Together for its multiplayer nature, its ability to configure the world we play in, the possibility of including our own modifications to the game natively, and for its sanity mechanic. A novelty among its peers, the sanity mechanic represents the character's mental health, another component that enhances the player's immersion by giving the character a human trait.

Then we implemented the *mod*, FAtiMA-DST, and the console application, FAtiMA-Server, that allow us to run AI powered NPCs in DST. As an example and proof of concept, we've also created Walter, a model based NPC that was published in the Steam Workshop.

Additionally, the platform has been successfully used to implement a planning agent using the MCTS algorithm. By creating a new Dynamic Property, the developers were able to create an agent using FAtiMA and run it in DST.

In order to evaluate our work, we've compared our example agent, Walter, with Artificial Wilson, an agent created using the in-game's behaviour trees by an anonymous *modder*. Although Walter's performance is poor when compared to Artificial Wilson, the discrepancy can be justified due to the difference in the size of both implementations. While Walter counts with nineteen rules of decision, Artificial Wilson counts over seventy nodes in its behaviour tree.

This dissertation provides the ground work for future developments in the creation of NPCs for survival games, in particular for Don't Starve Together. It is our belief that with an equal effort, Walter's behaviour would at least match the performance of Artificial Wilson with the added value of social awareness, emotional responses, and active goal pursuit.

Future Work

As this work stands, it allows the creation of NPCs for Don't Starve Together. However, the platform has great room for improvement.

- Two-way communication. Currently, the platform allows the NPC to talk and make remarks on the world. However, the player is not able to respond and talk back to the NPC. Either by allowing chat responses or by adding an additional component to the game's interface, the interaction with the NPC could be greatly improved.
- Add more beliefs. Expanding the character's available beliefs will enable us to create more complex behaviour. However, only by adding behaviour to NPCs, will we understand which beliefs are effectively necessary.
- Improve Walter. As a proof of concept Walter plays its role as expected. Nonetheless, Walter is a simple model based agent that does not make full use of FAtiMA's capabilities. Although the framework is fully prepared to support FAtiMA, the character does not benefit from all the capabilities FAtiMA might grant to a virtual agent. By incorporating additional assets into the RPC's definition, like the Emotional Appraisal Asset or the Social Importance Asst, Walter could be further improved.
- Reduce the burden of running FAtiMA's characters in DST. As it stands, players have to run a console application in parallel with the game for the platform to work properly. It is our belief that by removing this requirement, more players would be able to play with NPCs that make use of our platform. Ideally, this would be done by incorporating FAtiMA's DLLs into the embedded Lua interpreter that DST comes with. However, this would require some sort of collaboration with the game developers, Klei Entertainment. Alternatively, a public server could be set up to host the FAtiMA-Server component.
- Add an additional layer of decision. Players can express feelings through the use of gestures. Gestures are animations that represent different states of mind the player may be feeling (some of the available gestures are "annoyed", "bye", "angry", "joy", "dance", and "sad"). The addition of this capability to the platform would be a great improvement as far as believability goes for created NPCs.
- Add more composed actions to the platform. Currently the only composed behaviour incorporated in the platform is the wander behaviour. The addition of combat behaviour, utilization of chests, structure placement helpers, and others, would greatly improve an NPC's capabilities.
- Different NPCs running different RPCs. For each NPC there should be an ability to specify the RPC to be loaded. This would enable the creation of richer scenarios by using characters with different characteristics, goals, reactions, dialogues, among others.

REFERENCES

[1] Nuno Afonso and Rui Prada. 2008. Agents That Relate: Improving the Social Believability of Non-Player Characters in Role-Playing Games. In *ICEC'2008 - 7th International Conference on Entertainment Computing (Lecture Notes in Computer Science)*, Vol. 5309/2009. Springer Berlin / Heidelberg, Pittsburgh, PA, USA, 34–45.

[2] Ruth Aylett, Natalie Vannini, Ana Paiva, Sibylle Enz, and Lynne E. Hall. 2009. But that was in another country: agents and intercultural empathy. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Vol. 1. IFAAMAS, 329–336.

[3] Ruth S Aylett, Sandy Louchart, Joao Dias, Ana Paiva, and Marco Vala. 2005. FearNot!–an experiment in emergent narrative. In *International Workshop on Intelligent Virtual Agents*. Springer, 305–316.

[4] Joseph Bates et al. 1994. The role of emotion in believable agents. *Commun. ACM* 37, 7 (1994), 122–125.

[5] Emily Brown and Paul Cairns. 2004. A grounded investigation of game immersion. In *CHI'04 extended abstracts on Human factors in computing systems*. ACM, 1297–1300.

[6] Filipa Correia, Patrícia Alves-Oliveira, Nuno Maia, Tiago Ribeiro, Sofia Petisca, Francisco S. Melo, and Ana Paiva. 2016. Just follow the suit! Trust in Human-Robot Interactions during Card Game Playing. *IEEE RO-MAN 2016, New York, USA*. (2016).

[7] João Dias, Samuel Mascarenhas, and Ana Paiva. 2014. *FAtiMA Modular: Towards an Agent Architecture with a Generic Appraisal Framework*. Springer International Publishing, Cham, 44–56.

[8] BioWare Edmonton. 2014. Dragon Age: Inquisition. PC, PlayStation 3, Xbox 360, PlayStation 4, Xbox One.

[9] Klei Entertainment. 2013. Don't Starve. Android, iOS, Linux, Microsoft Windows, OS X, PlayStation 3, PlayStation 4, PlayStation Vita, Wii U, Xbox One, Nintendo Switch.

[10] Klei Entertainment. 2016. Don't Starve Together. PlayStation 4, Microsoft Windows, Linux, Xbox One, macOS.

[11] Richard Evans and Emily Short. 2013. Versu — A Simulationist Storytelling System. *Computational Intelligence and AI in Games, IEEE Transactions on* 6, 2 (2013), 113–130.

[12] Miguel Oliveira Ferreira. 2017. *A Merchant Model Improving player experience with NPC vendors in Role Playing Games*. Master's thesis. Instituto Superior Técnico, Portugal.

[13] Guerrilla Games. 2017. Horizon Zero Dawn. Playstation 4.

[14] Manuel Guimarães. 2016. *Skyrim Mod for Social NPCs*. Master's thesis. Instituto Superior Técnico, Portugal.

[15] Manuel Guimarães, Pedro Santos, and Arnav Jhala. 2017. CiF-CK: An Architecture for Social NPCs in Commercial Games. *CiG 2017, New York, USA* (2017).

[16] Wijnand IJsselsteijn, Yvonne de Kort, Karolien Poels, Audrius Jurgelionis, and Francesco Bellotti. 2007. Characterising and Measuring User Experiences in Digital Games. (2007), 1–4.

[17] Theodore D. Kemper. 2006. *Power and Status and the Power-Status Theory of Emotions*. Springer US, Boston, MA, 87–113. https://doi.org/10.1007/978-0-387-30715-2_5

[18] Michael Mateas and Andrew Stern. 2003. Façade: An experiment in building a fully-realized interactive drama. In *Game developers conference*, Vol. 2. 4–8.

[19] Josh McCoy, Mike Treanor, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. 2011. Prom Week: social physics as gameplay. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. ACM, ACM.

[20] Ubisoft Montreal. 2014. Assassin's Creed Unity. PC, PlayStation 4, Xbox One.

[21] Andrew Ortony, Gerald L. Clore, and Allan Collins. 1988. *The cognitive structure of emotions*. Cambridge Univ. Press, Cambridge, New York.

[22] Ana Paiva, João Dias, Daniel Sobral, Ruth Aylett, Sarah Woods, Lynne Hall, and Carsten Zoll. 2005. Learning by Feeling: Evoking Empathy with Synthetic Characters. *Applied Artificial Intelligence* 19, 3 (2005), 235–266.

[23] Sérgio Hortas Rodrigues, Samuel Mascarenhas, João Miguel Assis Dias, and Ana Paiva. 2009. I can feel it too! Emergent empathic reactions between synthetic characters. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*. IEEE, 1 –7. https://doi.org/10.1109/ACII.2009.5349570

[24] Ira J. Roseman and Craig A. Smith. 2001. Appraisal theory: Overview, assumptions, varieties, controversies. In *Appraisal processes in emotion: Theory, methods, research*, Angela Schorr and Klaus R. Scherer (Eds.). Oxford University Press, London, 221–232.

[25] Bethesda Game Studios. 2011. The Elder Scrolls V: Skyrim. PC, PlayStation 3, Xbox 360, PlayStation 4, Xbox One, Nintendo Switch.

[26] Anne Sullivan, April Grow, Michael Mateas, and Noah Wardrip-Fruin. 2012. The design of Mismanor: creating a playable quest-based story game. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, ACM, 180–187.

[27] Thomas. 1981. *Disney Animation: The Illusion of Life*. Abbevillee Press, New York.

[28] Palli R. Thrainsson, Arnkell Logi Petursson, and Hannes Högni Vilhjálmsson. 2011. *Dynamic Planning for Agents in Games Using Social Norms and Emotions*. Springer Berlin Heidelberg, Berlin, Heidelberg, 473–474.

[29] Geogios N. Yannakakis. 2012. Game AI Revisited. In *Proceedings of the 9th Conference on Computing Frontiers (CF '12)*. ACM, 285–292.