TÉCNICO LISBOA

Artificial Populations in Games with Dynamic Social Roles

Defining a new Taxonomy and Implementation mechanism for Social Agents in Games

Diogo Miguel Barradas Eusébio

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Rui Filipe Fernandes Prada

Examination Committee

Chairperson: Prof. Pedro Miguel dos Santos Alves Madeira Adão
Supervisor: Prof. Rui Filipe Fernandes Prada
Member of the Committee: Prof. Carlos António Roque Martinho

October 2021

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my sisters, grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

I would also like to acknowledge my dissertation supervisors Prof. Rui Prada and colleague Diogo Rato for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

Each year virtual game worlds get larger as we are able to provide players with more and more space to explore; their graphical quality is also getting better with time, but one lacking area that has not been evolving at the same pace is the quality of agents that populates these worlds, specifically at a social level. As players demand more and more from games, the agents that inhabit them do not seem to be able to provide the expected behavior when reacting to the players. This work will describe the study process and identification of overall useful social roles for games. It also reports the development of an architectural structure as an application outlet of the previously studied roles by associating them to goal and action sets. And lastly provides a realization of said structure to serve as an example and testing platform, as a means for providing better, more believable socially capable agents at a gameplay level.

Keywords

Gameplay Agent; Non-Player Character; Social Behavior in Games; Socially Interactive Agents.

Resumo

De ano para ano os mundos virtuais de jogos são cada vez maiores, uma vez que somos capazes de fornecer jogadores com mais e mais espaco para explorar. A qualidade grafica dos jogos também fica cada vez melhor com o passar do tempo, mas em comparação, a área responsável pela qualidade do comportamento que estes agentes demostram tem ficado para trás. Enquanto os jogadores exigem mais e mais dos jogos, os agentes que os habitam não são capazes de forncer o comportamento de que lhes é expectado quando interagindo com os jogadores. Este trabalho descreve o processo de estudo e de identificação de papéis sociais para agentes de jogos. Para além disso, reporta o desenvolvimento de uma potencial estrutura sob a qual a aplicação dos papéis socias será feita. Finalmente fornece uma realização dessa estrutura, para servir de exemplo e como plataforma de testes, com o objetivo de ajudar designers a fornecer melhores e mais crediveis agentes ao nível de jogabilidade.

Palavras Chave

Agente de jogo; Personagens não Controláveis por Jogadores; Comportamento Social em Jogos; Agentes Sociais Interactivos.

Contents

List of Figures

List of Tables

1

Introduction

Contents

Characters created in games interact with the player across multiple contexts. They establish conflicts and dilemmas for the player, help to support the gameplay dynamics, and serve as a vehicle for the narrative. The outcome of players' interactions with these Non-Player Characters (NPCs) is often restricted by the character's drives and goals and the interaction context, for instance, the physical surroundings and their previous interactions. Additionally, besides engaging with the player, NPCs can also interact with each other, promoting a dynamic environment with social characters and, ultimately, laying the foundations for the emergence of societies in games.

With the increasingly larger dimension of digital games environments, it is necessary to enrich the players' experience with believable characters. Still, this believability should not only rely on high-quality graphics; the living actors must exhibit intelligent behavior, in particular, social behavior. As a result, creating believable NPCs becomes demanding and expensive to scale. Then, even if this is achieved, replayability is minimal, as characters will often fulfill the same social roles. Resulting in designers either putting a lot of effort into a small number of really well thought out characters that may hook the player into replaying the game as is the case in games like Dragon Age: Inquisition [BioWare, 2014] and The Last of Us [Naughty Dog, 2013]; or avoiding making games where multiple forms of social interaction are key to player immersion and game experience.

This work will describe (a) the study process and identification of overall useful social roles for any game; (b) the development of a potential architectural structure as an application outlet of the previously studied roles; and (c) a realization of said structure to serve as an example and testing platform, as a means for providing better, more believable socially capable agents at a gameplay level.

1.1 Background

Rato et al. explored the use and implications the term Socially Interactive Agents (SIA) has as a potential tool for games. SIAs can be NPCs, players, or even commentators and spectators that show an interest in a game. For this work, we will adopt this term. They also define three distinct Game Layers: (a) the *Gameplay Layer*, "This layer frames SIAs, and their interactions, as elements of the gameplay mechanics." [Rato et al., 2019] and encompasses every action and change in the game environment. Being the layer where this work focuses its efforts; (b) the *Narrative layer* that amounts to the story and fictional content of the world and presentation of SIAs; and (c) the *Player Layer* that refers to players and agents involved in the game from outside the game world, this may include AI that plays through an application programming interface (API), human players, and even any other SIAs that shows interest in the game, for example, commentators and spectators of the gameplay.

For the remainder of this work, we will also use the terms *character* and *agent*. A *character* encompasses the narrative layer of the being and supports the game interaction based on its story towards

portraying itself as intelligent. It regards its plot roles, emotional attachment, story relevance, and relations with other characters. In contrast, an *agent* is the physical representation of this character in a game world, it encompasses the gameplay layer of the being and it is subject to the game systems and dynamics. It controls the behavior the character demonstrates and usually displays a variety of sensors and actuators to interact with the game world.

1.2 Contributions

As highlighted this work intends to provide three clear contributions:

- A take on the definition of social roles applicable to games agents, that reflects the behavior the agent displays at a gameplay level, and the adequate comparison to the state of the art;
- The creation of an architecture that can make use of the resulting defined roles, providing a theoretical scheme to their implementation;
- Lastly an application that realizes the execution of the provided architecture, to serve as means of proof and for testing its real-world usefulness.

1.3 Organization of the Document

This thesis is organized as follows: In Chapter 2 we revise the state-of-the-art taxonomies for game agents and mechanisms to apply them. Then, in Chapter 3, we define a new taxonomy and provide an architecture for its implementation. In Chapter 4 we go over the implementation process of theory presented. The evaluation of the test environment is done in Chapter 5. Lastly, in Chapter 6, we revise the proposed taxonomy to identify the future work and the limitations of the developed work.

2

State of the Art and Related Works

Contents

There is no standardized, widely accepted taxonomy to describe and identify the social nature of game agents. The results, however, vary greatly depending on the initial premise each work is built upon, its scope, and its ultimate goal. For example in [Warpefelt, 2016] *Warpefelt* sets out to "provide a holistic description of NPCs" that "takes the form of a typology and a model of NPCs". While *Bartle* focuses on the functionality of NPC's for world-building [Bartle, 2004]; and *Rivera et al.* designs patterns specifically for NPCs in shooters [Rivera et al., 2012]. This results in a myriad of terms that can, only sometimes, be interchangeable and other times, due to minute differences in definition - hence why the use of the different terms - can not. With that in mind, we look into what is transmissible from these and other works to our gameplay-focused taxonomy for social roles for game agents.

2.1 Study of taxonomies and typologies

Rato et al. define the multiple roles they believe SIAs can take, distinguishing them according to the collaborative nature of the social interaction towards the player. SIA's can be Teammates, and try to help the player; Opponents, by opposing the player; or Neutral if their goals do not align nor generate conflict with the player [Rato et al., 2019]. Then for each of these meta relations, they specify more specific roles based on the circumstances in which the interaction takes place.

Warpefelt defines a layered typology [Warpefelt, 2016] that is summarized in Table 2.1. This typology revolves around the intent NPC's have towards the player and the resulting gameplay experience. *Warpefelt* reports that there are "areas in which NPCs tend to fail to uphold believability". This refers mainly to interactions in which the NPC's are required to interact and/or react to events generated by the player. *Warpefelt* then creates a model based on his typology that incorporates the distinction of embedded and emergent gameplay behavior leading to the definition of a large design space to be filled with different character types. Then concludes that even though his model sets the bare minimum requirements for an NPC to behave believably, and that the design space his typology covers is relatively small compared to the entirety of the design space, it should be encouraging to see that there is still a lot of room for character types definition to grow as players start to demand more.

On a Meta Level, [Warpefelt, 2016] describes the role the NPC takes based on the type of gameplay it is expected to deliver, then on a more specific level describes how that gameplay feature is exposed, and finally goes on highlighting potential specifications that type of gameplay features might be shaped after. Evidencing that the typologies used should explain the intended gameplay experience an interaction between an NPC and the player should take, instead of describing the goals each NPC type might have related to the game world.

Metatype	Type	Subtype
Functions	Vendor	
	Services	
	Questgiver	
Adversaries	Enemy	Boss
	Opponent	Manipulator
Friends	Sidekick	
	Ally	
	Companion	
	Pet	
	Minion	
Providers	Storyteller	
	Loot provider	

Table 2.1: Taxonomy for NPC Social Roles. Table from [Warpefelt, 2016]

Imagine a game where the player is a merchant, very simple ways to define it's goals could be, to make a lot of money, or create a monopoly over a market, etc... Now imagine another game where the player interacts with an NPC that is a merchant. It's much more useful as a design guideline to describe the goals of the merchant to be to provide resources to the player, even if in fact from a narrative perspective the merchant NPC actually wants to get a lot of money or said monopoly over a market.

But if we intent to merely have a simulation, with only artificial agents and no players involved, then designing goals around the players is limiting. As such we should strive for a model that socially shapes players and NPCs the same way. Allowing developers to create agents merely based on the relationships they are meant to have with the other agents, be it a player or an NPC.

One of the bases for *Warpefelt*'s work was *Bartle*'s typology for NPC functional roles [Bartle, 2004]. And although *Warpefelt*'s work demonstrates its flaws, mainly that it's over-fitted to "Multi-User Dungeons" games. It can reveal useful information when pitted against other typologies. In Table 2.2 we can observe that these typologies focus on specifying different areas. For example, *Rato et al.* identified more specific roles when it comes to exposing narrative from a gameplay point of view by proposing multiple roles agents can take to do so by defining an Advisor; a Commentator, and a Background Agent. Definitions that would all fit into a Storyteller NPC from Warpefelt. In contrast, *Warpefelt* defines more roles for positive characters that accompany the player, like the Sidekick, Ally, Companion, Pet, and Minion, while *Rato et. al.* encapsulates all of them just as Companion or subordinates. Studying these differences and identifying how specific roles can and should be defined is key to providing a new typology that hopefully has more quality, is more robust, and is less ambiguous. This also shows us how hierarchically these specifications tend to be interpreted has, giving us even more information on how to structure future typologies.

Rato	Warpefelt	Bartle
Companion	Sidekick	Do stuff for players.
	Ally	
	Companion	
Subordinate	Pet	
	Minion	
Adviser	Storyteller	Supply background information
Commentator		
Background		Make the place look busy.
Challenger	Questgiver	Dispense quests
Provider	Vendor	Buy, sell and make stuff.
	Services	Provide services.
Opponent	Loot Provider	Get killed for loot.
	Enemy	Guard places.
	Opponent	

Table 2.2: Comparison of reviewed taxonomies

2.2 Study of mechanisms to apply the defined taxonomies

By defining goals *Rato et. al.* architects a structure of social relations while leaving the complexity of how the goals are achieved to run-time decision and execution mechanisms. With a goal-action planning system, we can define goals and assume the actions are correctly defined by each game's designers and are believable. This results in a modeling system that is generic enough to be applied to any type of game while still providing the ever so useful and desired social structure of relations.

Additionally, *Rivera et al.* in their work on design patterns for NPCs in shooters describe the types of characters in a goal-oriented manner [Rivera et al., 2012], not just on what the character tries to achieve from a gameplay point of view, killing the player, but highlight the design decisions and reasoning each type of agents takes to achieve a good gameplay experience. For example, they define that one of the goals of the *grunt*, an opponent easy to overcome, is to bait the player into following them, serving as a tool to guide the player through the levels. This also opens the design possibility for traps where the player is presented with a harder challenge if it takes this bait of attacking the grunts, in contrast with ignoring them and exploring other routes.

Both these arguments suggest that the use of a Goal-Oriented Approach (GOAP) is a very good mechanism to be delegated the desired behavior for the defined roles. As such we use the work of [Orkin, 2006] to once again reinforce that GOAP is an adequate mechanism. *Orkin* describes their model as sets of goals and sets of actions. They provide one example where characters with different sets of actions are placed in the same environment with the same set of goals, and as a result manifest different behaviors to achieve said goals. This delegates the complexity of all interactions of states to A.I. systems, as they demonstrate in Figure 2.1.

"Designers are not responsible for scripting the behavior of individuals, other than for story elements.

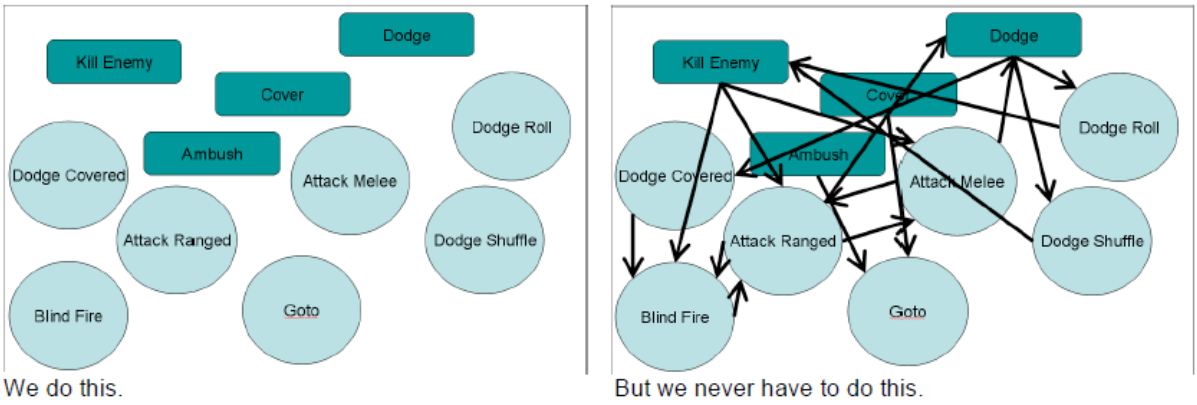


Figure 2.1: Reduction of design complexity and workload by using a goal-action abstraction. Figure from [Orkin, 2006]

This means that the A.I. needs to autonomously use the environment to satisfy their goals." [Orkin, 2006]. Having A.I. systems resolve how agents should behave at run-time using a goal-action system lends designers the freedom to still define how agents behave without having to solve the non-trivial task of deciding when each action should be taken and all the implications and repercussions that action may take on the environment.

3

Social Roles for Gameplay Agents

Contents

In this work, agents will be defined based on their social goals. As such, to easily identify them and lay a foundation of terms that can be used by game developers and enthusiasts to quickly mention said agents, we will now propose a taxonomy that tries to separate and encapsulate the various qualities each agent might possess, based on their contributions to gameplay.

3.1 Defining a new taxonomy

The following taxonomy builds mainly on the SIAs defined by *Rato et al.* [Rato et al., 2019] being the same that was presented in the proposal document #ADDREF and separates roles into the three main groups, regarding relations to other agents: *allied*, *neutral* and *opposing*, while identifying in-game SIAs with the main focus on the gameplay layer.

3.1.1 Allied SIAs

Can be separated into **Companions** or **Teammates**, characters that accompany the journey of the player. Sometimes referred to as "sidekicks". They are "on screen" with the players most of the time of the gameplay experience. They take special attention to goals and actions players try to perform but may have their own goals as well. Companions often have a presence in the narrative layer, as well, to support building deep interpersonal relationships with the player. Examples of this type of character are other players in multiplayer games that share a common goal and must work together to achieve it; Pillars of Eternity [Obsidian Entertainment, 2010] Companions, characters the player meets throughout the game and can choose to have them accompany them on his journey. Some of them prompt specific quests related to their narrative. Two of the most famous sidekick in games are Clank from Ratchet and Clank [Insomniac Games, 2002] games series and Daxter from Jak and Daxter [Naughty Dog, 2001] games series, where they provide the player utility tools, that the player's main character wouldn't otherwise have access to. Video games based on movies also tend to have companion characters as they place the player in the shoes of a protagonist and have the side characters take this role, such as the case with Hermione and Ron in games from the Harry Potter [Electronic Arts, Griptonite Games, 2001] series.

Subordinates, agents that perform tasks for the players. Also referred to as henchmen or minions. They mostly perform tasks and goals delegated by players. This involves a power relationship as players have control over the goals that subordinates commit to. These agents have autonomy, but only to fulfill the designated goals. They may be proactive, nevertheless, and autonomously take goals they believe are relevant to the player. Examples of these types of characters can widely be seen in RTS games like Age of Empires 3 [Ensemble Studios, 2005] or StarCraft [Blizzard Entertainment, Saffire, 1998], where

the player controls multiple characters and each of them fulfills this role. Examples in other genres can be seen in games like Darkest Dungeon [Red Hook Studios, 2015], where the character belongs to the player's roster without having significant narrative relevance; minion cards also fulfill this role in Heartstone [Blizzard Entertainment, 2014].

From a gameplay point of view, there may be no difference in the behavior of Companions and Subordinates, as their differences are usually present in the narrative layer. Nonetheless, as these two layers are highly interconnected we believe it makes sense to distinguish these two types, in hopes of providing a more familiar and easier to understand topology to developers.

Adviser or **Helper** is an SIA that indirectly contributes to the efforts players make towards achieving their goals in the game. Advisers convey information to players about the game state and provide advice about gameplay actions and strategies. They can be specialized in certain areas of gameplay (e.g. economic, military, research, as in the Civilization [MicroProse, 1991] game series). The information may be proactively suggested or only given when explicitly requested by players. They often introduce players to the game mechanics and support their learning about the game. A clear-cut example of this can be seen in Anno 2205's [Blue Byte Software, 2015] Supervisor Sam Beaumont a character whose primary role is to help the player by introducing mechanics and alerting them to miss-management of resources.

3.1.2 Neutral SIAs

The first neutral SIA is the **Provider**, an SIA that provides resources, information, services, and tools that players need to progress in the gameplay and fulfill their goals. The players get what they need after a successful interaction with providers. This means that they, typically, need to make an effort to succeed in the interaction. This may be a simple commercial exchange (e.g. if the provider is a shopkeeper) or may require some kind of negotiation. But, the option to freely provide the resources after a simple contact is open as well. The effort in this case is the time spent to go to the provider. Providers may also unlock new gameplay options (e.g. avatar abilities). Examples can be seen in numerous games from virtually all game genres taking many different forms. To list a few: in Hades [Supergiant Games, 2018] gods provide boons and Charon a market; in Kingdom Come: Deliverance [Warhorse Studios, 2018] trading involves haggling, and in Total War Saga: Troy [Creative Assembly, 2020] the diplomacy status between factions influences how likely a trade is to be acceptable.

Then there are **Challengers**, agents that provide challenges to players (e.g. a quest giver). These are somehow similar to providers as they may provide rewards as well. But, their main role is to explicitly define goals for players to follow. For example, the professors from the Pokemon [Game Freak, 1996]

games franchise, that send the play on a quest to fill a Pokedex. They may serve as “gatekeepers” that lock and unlock the game progression as they may have strong control over the goals that are open to the players. The challenger role may be taken by an opponent as it may raise a special confrontation goal for the player that is triggered once the player meets it for the first time (e.g. a boss battle). For example Dr. Robotnik from the Sonic the Hedgehog [Sonic Team, 1991] games franchise.

Commentator is a SIA that describes the gameplay action and may present an assessment of the gameplay results as well. Commentators are not at the service of the player as the advisers. Although, the information they provide can be useful to help and guide players. They present a shared view of the game state to all the agents in the game world, that can influence the gameplay decisions. However, they often serve the audience of the game (e.g. commenting on a Soccer match on a match of FIFA from the FIFA [Extended Play Productions, 1993] games franchise), which includes players. In this sense, they have cross-layer agent-player interactions, or may even be placed outside the game world (at the player layer). Other examples of this can be seen in more artistic styles of narration, as is the case with the narrator in Darkest Dungeon [Red Hook Studios, 2015].

Background SIAs are used to bring social life to the game world. They do not influence the game progression but may be affected by it. These SIAs react to players and engage in social interactions if requested. Background SIAs provide context to the interactions with other SIAs and may depict and support understanding of the game’s social world. They, often have a strong representation in the narrative layer to help enhance the social dimensions of the fictional world. Nevertheless, they are actors in the gameplay layer as they may constrain the gameplay actions players to take (e.g. a player may not be allowed to kill an opponent in a public space). Classical examples of this can be seen in games from the Hitman [IO Interactive, 2000], Assassin’s Creed [Ubisoft Montreal, 2007], Grand Theft Auto [DMA Design, 1997] and Elder Scrolls [Bethesda Softworks, 1994] franchises where witnesses of the players actions are crucial features of gameplay and bring this social life to the game world. Going back to already given examples, characters like Clank from Ratchet and Clank [Insomniac Games, 2002] and Anno 2205’s [Blue Byte Software, 2015] Supervisor Sam Beaumont are also great vehicles to bring life to the game in stagnant situations, by using animations when the player is not moving or making remarks and even jokes insignificant to game progression. A joke in the industry that surges from the lack of attention to this role of characters, takes the form of the question ‘Can you pet the dog?’, originating from the lack of interaction, and therefore break of immersion when players tried to interact, and couldn’t, with characters whose role was to give the world life. A personal favorite implementation of this role is the chickens from Call of duty: Modern Warfare 3 [Infinity Ward, 2011], which just give the player something else that moves to shoot at.

Hosts or **Referees** are SIAs that manage in one way or another the interaction between other SIAs and the player in a certain context, be it by directly enforcing rules, or indirectly influencing their behavior when interacting with them. Probably the most recognizable character that portrays this role is Buzz from the Buzz! series [Relentless Software, 2005]. Once again, just like commentators are portrayed in sports games, so are referees, and an iconic interaction is observable in FIFA 94 [Extended Play Productions, 1993] where the player can run from the referee when being booked.

Hostages and **VIPs** are SIAs that the player must protect, guide, and/or contain while competing with opposing SIAs or players to achieve a certain goal. These characters are usually seen as temporary objectives, but unlike mere objects, the player must move, protect, and/or save, these SIAs display social behavior. Examples include prisoners the player can save in games like Assassin's Creed Valhalla [Ubisoft Montreal, 2020]. There are also many games with escort missions, a classic one being ICO [SCE Japan Studio, 2001] where the player must protect Yorda to not lose.

3.1.3 Opposing SIAs

Finally, we define three types for opposing SIAs. **Competitors**, SIAs that directly compete with the players over one or more objectives. They have the same or very similar goals as the player. If the chance arises they will proactively act to try and make sure they get a better result than the player. Examples in games can be opposing Nations and their respective units and buildings in Age of Empires 3 [Ensemble Studios, 2005]; Opposing heroes and minions in Heartstone [Blizzard Entertainment, 2014]; or opposing racers in Mario Kart [Nintendo Entertainment Analysis & Development, 1992].

Stoppers, SIAs that don't care about the player until a player's action triggers them to react. The most common use of this SIA is as security guard characters, as their goal is to protect something from the player and not necessarily defeat the player, as they might give up chasing the player if he acts in a way the SIA no longer considers threatening to the goal they're protecting. A policeman from Grand Theft Auto V [Rockstar North, 2013] is a great example of this as they'll only react to the player when he commits acts they perceive as illegal.

In contrast, **Colliders** are SIA's whose goals are to disrupt the player, by always looking to interfere with them. Pigmen and Zombies from Minecraft [Mojang Studios, 2009] can be seen as portraying the stopper and collider's roles respectively. Zombies will attack the player if he is at a certain distance; as Pigmen will only attack the player if he is at a certain distance and has attacked one of them first.

In Sum at the time of the proposal, we believed the just presented roles covered a sufficient spectrum

of possible social interactions in games, having associated multiple game tropes to the defined roles, making them effective as short communication terms that rapidly convey a piece of sizable information while keeping it distinguishable.

However, due to results discussed in the two following chapters: 4 Agent and Game Environment Implementations, and 5 Evaluation, in chapter 6 Conclusion, we will review this taxonomy.

3.2 Architecture

To realize the use of this theoretical taxonomy an architecture is needed to bridge the theory to the practical results.

In figure 3.1 we present a UML Class Diagram that establishes our definitions, of agent, social role, goal, and action; and the connections each of these aspects has. We start by defining that an agent can have multiple roles, multiple objectives, and multiple actions to fulfill them. Each goal can have multiple actions that can be used to achieve it. By establishing an agent manager we can dictate what actions each type of agent can use. *Orkin et. al.* in [Orkin, 2006] exemplifies that agents with the same goal set can react differently depending on their action sets. "Imagine that we created a Goal Set named GDC06 which contains only two goals, *Patrol* and *KillEnemy*. When we assign this Goal Set to our soldier and run the game, he no longer ignores the player. Now he patrols through a warehouse until he sees the player, at which point he starts firing his weapon. If we place an assassin in the exact same level, with the same GDC06 Goal Set, we get markedly different behavior. The assassin satisfies the *Patrol* and *KillEnemy* goals in a very different manner from the soldier. The assassin runs cloaked through the warehouse, jumps up and sticks to the wall, and only comes down when he spots the player. He then jumps down from the wall, and lunges at the player, swinging his fists. Finally, if we place a rat in the same level with the GDC06 Goal Set, we once again see different behavior. The rat patrols on the ground like the soldier, but never attempts to attack at all. What we are seeing is that these characters have the same goals, but different Action Sets, used to satisfy the goals. The soldier's Action Set includes actions for firing weapons, while the assassin's Action Set has lunges and melee attacks. The rat has no means of attacking at all, so he fails to formulate any valid plan to satisfy the *KillEnemy* goal, and he falls back to the lower priority *Patrol* goal." Now by designing Agent goals, more specifically *social goals* for the taxonomy we get an easy modular system whereby changing the roles an agent has, we get changes to its behavior. Meaning that the roles an agent has directly influence the way it sees and is seen by the world, which will hopefully provide more interesting and rich experiences for players interacting in the same world.

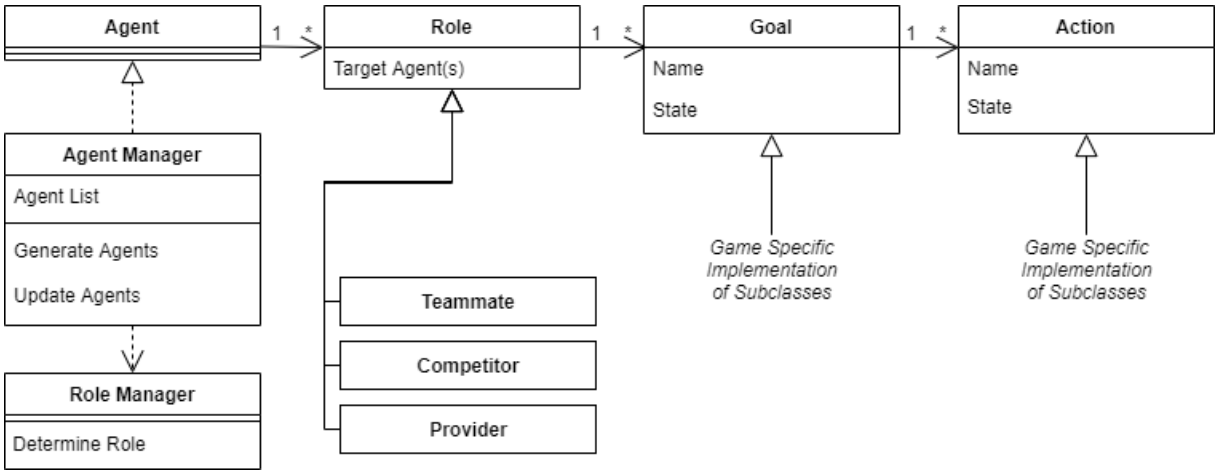


Figure 3.1: UML Class Diagram

3.2.1 From Social Role to Action Execution Cycles

We identify and suggest the use of the following two major cycles when implementing our architecture. The first is the role attribution and variation, which we intrinsically connect to the agent generation, meaning the agent's existence is entirely dependent on the roles they are meant to fulfill; the second is the attribution and achievement of goals given to agents by the former cycle. This can be seen in Figure 3.2.

The first Cycle is essentially a feedback loop resulting in the information flow from a Social Role to an Agent Goal. Once a Goal is removed from an agent, be it by completion or failure, it can have Social ramifications to the agent's relationships with other agents. So at this time of execution, we need to verify the integrity of these relationships and update them accordingly.

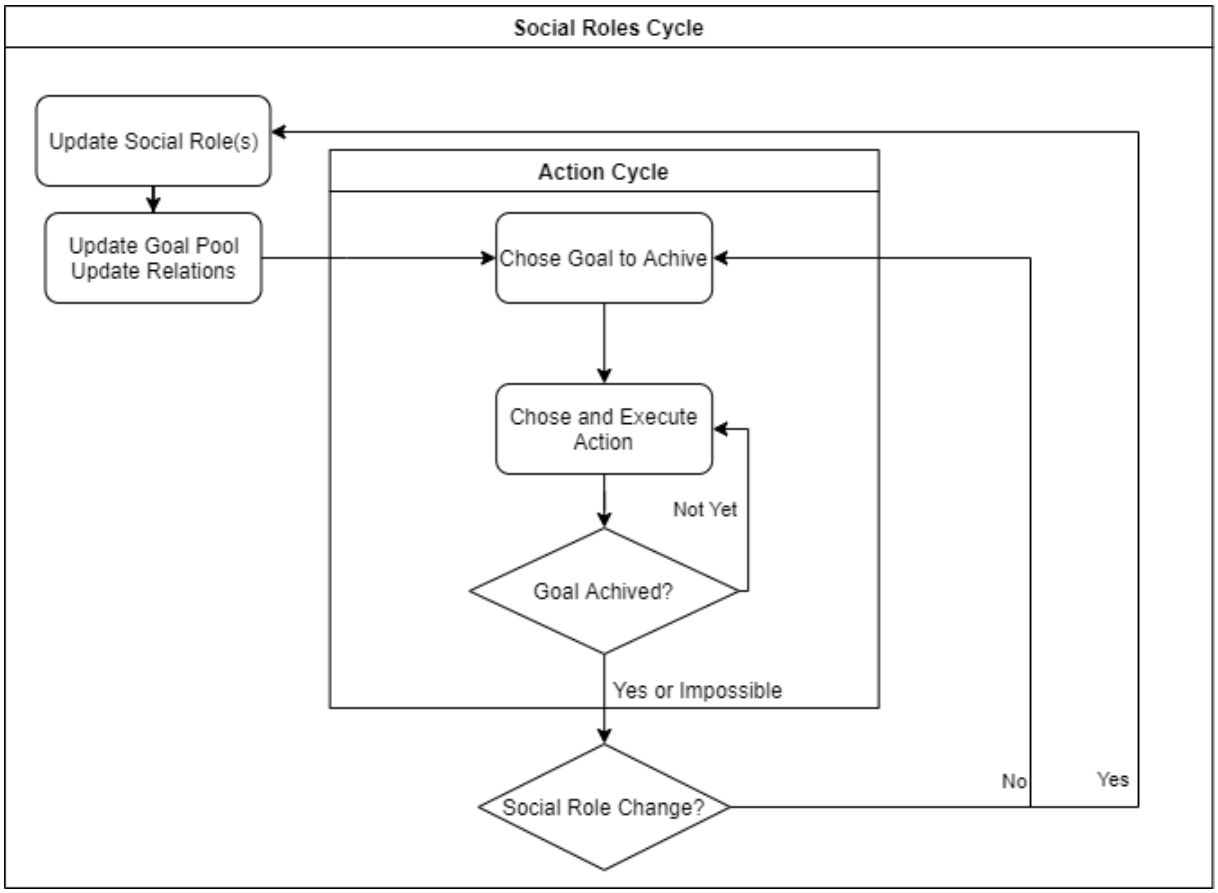


Figure 3.2: Major Execution Cycles of the Architecture

The Second Cycle is yet another feedback loop, yet this one is a result of the Action Execution information flow to the Goal Achievement. One Very important aspect of actions is that they should be designed to be executed in small time frames. For example instead of defining ”walk from point A to point B” we can define ”walk at each engine update in the direction of point B”. This way keeping track of progress is simpler since we don’t have to define smaller milestones inside the attempt to achieve a goal. With that, all we have to do when the execution of an Action results in the accomplishment of a Goal is relay the information to the previous Cycle that will verify if the Social Environment changes and eventually redefine our Goals, so this cycle can start anew.

4

Agent and Game Environment Implementations

Contents

To provide a practical example of the developed architecture that uses the defined taxonomy, we developed a small game with a simple environment. And intentionally simple Agents to: (a) simplifies the development process; (b) establish that the proposed theory works independently of the capabilities of the Agents that apply it; and (c) control and limit as much outside noise as possible since, agents can (and should, just not for our testing purposes) incorporate other aspects apart from our gameplay level theory, such as graphical appearance and presentation to the players and other agents; narrative involvement and influence.

Initially, we decided that the game objective would be a *Tug-of-war*, an idea that was scrapped and changed into a *Capture the Flag* for our development simplicity when adding a 3rd team since we want to evaluate the social relation in more than a 1 to 1 axis. After which we developed the agents while realizing the proposed architecture.

4.1 Game Environment Implementation

The game proposes a simple objective, a *Capture the Flag* competition between three teams, each identified by a color, red, green, and blue, as we wanted to explore opposing relations on more than one axis, this being agents having other multiple opposing agents that may not be related to each other in a friendly fashion, as is usually the case in two teams versus competitions. As such the environment implemented is a 3D hexagonal arena that agents can be interpreted as a 2D space (since for simplicity sake we did not explore Agent movement in the vertical axis) that has six interest points: (a) three team bases, that house three flags, that react to Agents deploying opposing teams' flags and are the spawn and respawn points for agents of this team; (b) three resource nodes; from which resources can be collected to be consumed to heal Agents, that regenerate the resource sometime after it being collected. An initial game setup is provided in Figure 4.1

For the tests in the following chapter 5 each team will be comprised of 3 agents, one of each of the first three types, a Cone, a Cube, and a Sphere. There are also be 3 neutral/independent Tetrahedron agents with the provider role that spawn in the center of the map and try to collect and deliver resources to the closest agent that belongs to a team. An example of the initial setup can be observed in Figure 4.1 The win condition is: have five out of the nine total flags in the team's base.

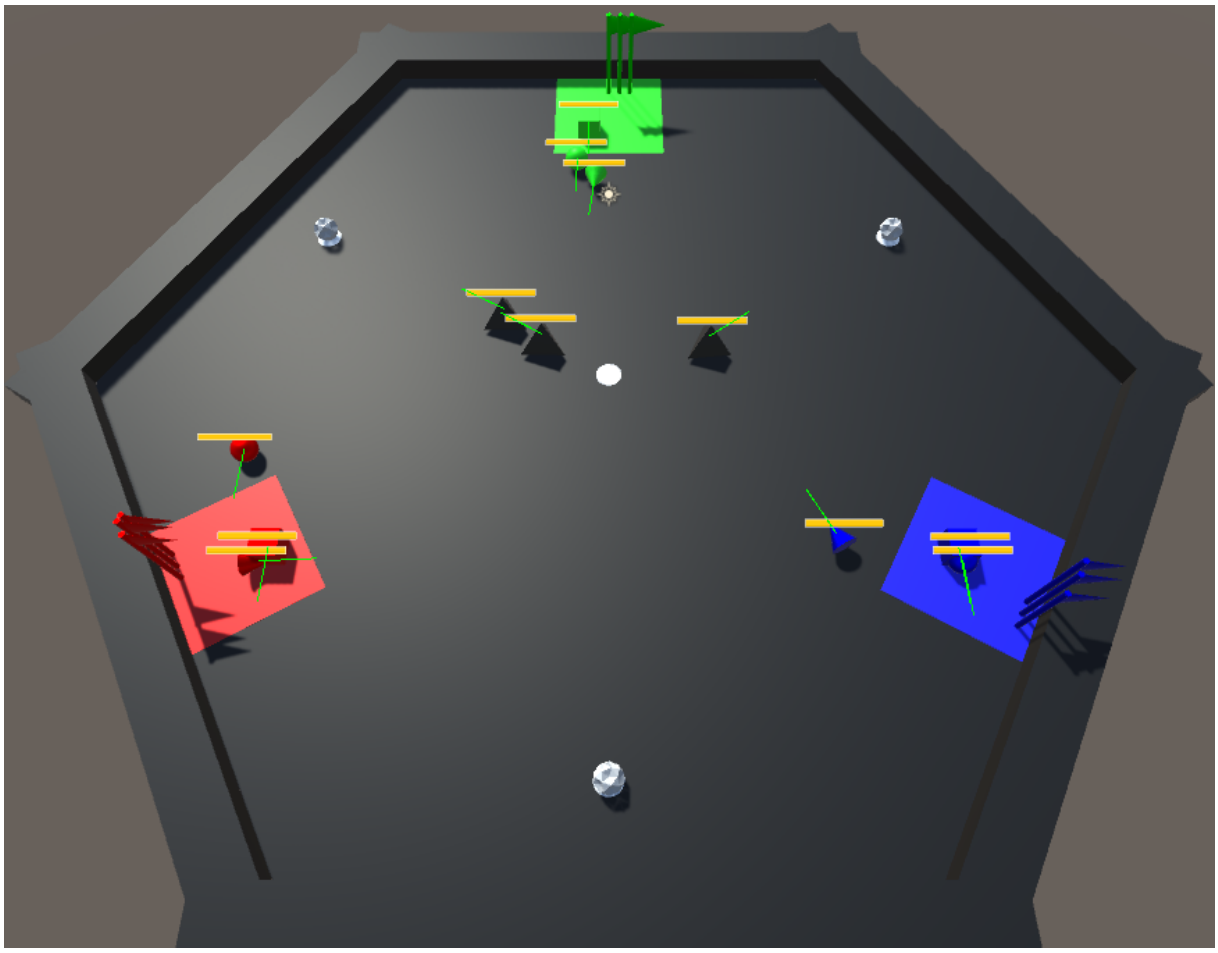


Figure 4.1: Initial Game Setup Screenshot

4.2 Agent Implementation

We implemented four types of agents, each with different capabilities that can be identified by their shape. To differentiate their behavior we give each type of agent a subset of goals they will follow during gameplay and define how they are capable to act to achieve said goal.

First, we developed the **Cone Agent**, this agent is intended to be fragile and have low mobility, but it possesses the ability to shoot projectiles at opposing agents from a distance. So we implemented the *Shoot Bullet Action*, that as the name implies shot a bullet, for simplicity we make this type of agent execute this action periodically every second while it intercalates it with other actions, such as the *Capture the Flag Action*. Then we developed the **Sphere Agent**, this agent is intended to be a melee attacker so we designed the *Charge Attack Action* for it. This agent can also capture flags. The **Cube Agent** was meant to fill the typical tank role seen in many games. So we gave it the ability to not be damaged by bullets shot by the Cone Agent. With this, we also gave it the Shield Ally Action that has

the Cube Agent position itself in between an ally and the center of the arena, from where the majority of bullets should be coming from, while still allowing some flanking scenarios.

These three agents were designed to function in a *Rock-Paper-Scissors* way, the Sphere Agent is the only agent capable of killing cube agents; the cone is good at killing sphere agents by shooting while they have to close the distance gap, and the Cube Agent nullifies the Cone Agent bullets. They are all able to perform the *Capture the Flag Action* that has the agent move toward the closest unattended flag belonging to a competing team when the agent has no flag and has the agent try to return to base if it collected a flag. The Sphere agent gives priority to capturing flags over, charging at enemies, while the Cube Agent gives priority to shielding allies and only tries to capture flags when the ally they are defending dies and if a flag is closer to them then another ally.

At this point using our architecture we attributed these actions, and the goals that provide them, to the respective social role. Resulting in mapping the Attack Enemy Goal and Capture the Flag Goal to the Competitor Role and the Protect Teammate Goal to the Teammate Role, introducing basic forms of the two most common roles in games, friendly and opposing agents to our game.

Attempting to introduce a Neutral role into the game we developed the resource system, where neutral agents would collect and deliver resources to the competing agents. These resources would, on delivery, slightly heal the entire team of the competing agent. As a result, we got the Provider Role and developed the fourth agent type the tetrahedron, whose mission was to prove that the existence of neutral agents in games enriches the experience, by adding a new layer of complexity to the game, in this case, being the healing from the resources. We then developed a new method of attack for the Sphere Agent, trying to support the point made previously by *Orkin et. al.* [Orkin, 2006], that by changing the action set while fixing the goal set the social dynamic would change. To summarize in Table 4.1 we layout all the goals each role has and the actions each type of agent can take to try and achieve said goals.

Also of note is that the neutral agents were programmed to dynamically change roles as a game mechanic. We establish an allegiance score the neutral agent has with each team, that increases and/or decreases when the agent provides a member of a team with a resource. Giving a resource to an agent increases the allegiance score with that agent's team. If the difference between the allegiance scores with two teams is bigger than a set threshold, the agent is now considered a teammate with the high score team agents and enemy with the low score team agents. The allegiance is shown by the current color of the provider agent. For example, consider a game where the agent starts neutral to all teams *(RGB black vector)*:

$$(redteamallegiancescore = 0, greenteamallegiancescore = 0, blueteamallegiancescore = 0)$$

If the threshold is one and the agent provides a member of the blue team with a resource, since the

<table>
<tr><th>Role-Goal \\ Agent Type</th><th></th><th>Cone Agent</th><th>Cube Agent</th><th>Sphere Agent</th><th>Tetrahedron Agent</th></tr>
<tr><td rowspan="2">Competitor Role</td><td>Attack Enemy Goal</td><td>Shoot Bullet Action</td><td>none</td><td>Charge Attack Action/ Lay Explosive Mine Action</td><td rowspan="2">none</td></tr>
<tr><td>Capture Flag Goal</td><td colspan="3">Capture Flag Action</td></tr>
<tr><td>Provider Role</td><td>Collect Resource Goal
Deliver Resource Goal</td><td colspan="3">none</td><td>Collect Resouce Action
Deliver Resource Action</td></tr>
<tr><td>Teammate Role</td><td>Protect Teammate Goal</td><td>none</td><td>Shield Ally Action</td><td>none</td><td>none</td></tr>
</table>

Table 4.1: Roles; Goals and Actions realization by Agent Type

difference the blue team's allegiance score and the other two teams allegiance scores (which are zero) is greater or equal to the threshold (one), the provider agent gains the teammate role to the blue team's agents and they reciprocate, and gains the competitor role regarding the green and red teams agents, becoming blue in the process *(RGB blue vector)*:

$$(redteamallegiancescore = 0, greenteamallegiancescore = 0, blueteamallegiancescore = 1)$$

Then if the provider agent delivers a second resource to an agent from the green team, it will become an ally with the green team too, becoming cyan *(RGB cyan vector)*:

$$(redteamallegiancescore = 0, greenteamallegiancescore = 1, blueteamallegiancescore = 1)$$

The agent resets to black when all allegiance scores become the same or when it dies. We did this to have a dynamic way of changing relationships during gameplay.

As can be seen in Table 4.1 marked as *none*, there are a few cases where the agents have no actions that allow them to achieve certain goals, that is intended, not only did it make development simpler - since we didn't have to design an action for each agent that fulfills every goal - it makes the lack of capacity to act in certain scenarios, present agents limitations the player can use for their benefit, for example, a strategic advantage. It also serves to show the full influence the roles can have in the game environment through the established social panorama. Take the example of the changing tetrahedron agent: when this agent is neutral it is never attacked, but after becoming allied to a team and conversely determined as a competitor with that team's competitors, it now can be attacked and protected. Just having these

roles can affect the behavior of the agents the roles associate him to.

4.3 Limitations

It is of interest to regard the physical limitation of the agents and the environment. There is a finite number of objectives, in this case, Flags to capture, that in conjunction with the limited movement speed agents have, we can expect the number of flag captures over time to be constant, and only slightly influenced by the number of deaths and when they happen since an agent that dies is removed from the game over a cooldown that increases with total game length. However, since there are resources that heal agents that have been dealt damage, the number of death should be mitigated enough to affect this flag capture rate in a significant manner, hence why we expect it to behave closely to constant over time. Second, the agents only execute about one action per second, which sets a good time unit for our tests.

4.4 Development Tools

To develop the game we used Unity version 2019.4.22f1 since it was the most recent version at the start of the developing process. We opted to use the unity engine, due to familiarity, and since it is built over C Sharp, that provides an update mechanism that is a good fit for our vision of execution of actions. It allows us to design a decision mechanism that can change the desired goal without having to interrupt actions since we can make them execute virtually frame to frame due to the update system. Unity also provides the *MonoBehaviour* class that allows to easily associate the agent behavior and decision making to a physical body in the game environment. For the data analysis, Python Jupyter Notebooks were used, since it allows for more dynamic and comfortable testing, by separating code into blocks enabling to separate the processed data from the testing functions, not being necessary to reprocess data for every new test over it. All the developed code can be seen in a GitHub repository [1].

[1] https://github.com/DiogoEusebio/ThesisShowcaseGame

5

Evaluation

Contents

For an initial evaluation we had two scenarios to test the quality of the provided architecture, by comparing the effects of changing the relations agents have:

A A free-for-all with three teams where team members are Teammate to each other and Competitors to members of the other teams, with the presence of a group of neutral/independent Agents with the Provider Role relative to the members of every team;

B The same three-team game, but instead of a free-for-all, two of the three teams aren't directly competing with each other, meaning they are indifferent to each other, having no role to relate them. Resulting in one team competing with two teams, while they only compete back at them. In illustrating, let us have teams, A, B, and C. Team A only competes with team C, Team B also only competes with team C, Teams A and B have no roles relating them to each other, therefore do not directly interact with each other. As in the previous case, there will also be the same group of neutral/independent Agents with the provider Role relative to the members of every team;

To assert our results we will use game-length, the number of actions executed, and position dispersion over time as measurements of the changes in the environment as these metrics reflect changes in the game environment in objective ways, meaning that differences in these metrics can be seen as practical results originating from the changes in agent roles.

5.1 Test scenarios hypothesis

In this Section, we will hypothesize the results of the scenarios mentioned to later compare them to the actual results.

5.1.1 Scenario A: Balanced Competition

As the subsection title indicates, we expect this scenario to be the most balanced. As already depicted in 4.1 all the teams have the same members and the same number of flags. The agents all also have the exact same relations to each other. As such we can expect games to be drawn out. The only factor that makes the game possible to end will be the targeting randomness of the agents' actions. Depending on who each agent targets: (a) an agent might be shot by two competitors from different teams; (b) a team might be provided resources from multiple neutral/independent provider agents; or (c) a team's flags might be targeted by the other two teams at the same time, resulting in flags concentrating in two of the three bases.

As a result of the drawn-out game, we expect agents to take a considerable amount of actions.

We also expect games to have a balanced position dispersion over the physical environment, meaning that agents are expected to traverse mostly between team bases due to the shortest path to the

flags.

5.1.2 Scenario B: Competition Imbalance

In this Scenario we expect games to be very quick. There will be a team that is constantly attacked by the other two teams, and with the relaxed win condition of five out of nine flags there is a high likelihood of the first team to take two flags from the solo team will win. The solo team agents are expected to die sooner since they are outnumbered by the other two teams that will never attack each other. Position dispersion is also expected to change a lot, now instead of a triangular shape connecting the three bases, we expect to see a 'V' shape with the vertex on the team being attacked by the other two, since each of the two non-competing teams will never try to directly steal a flag from each other, the only option the agents have is to attack the one base they are allowed to, by their Social Roles.

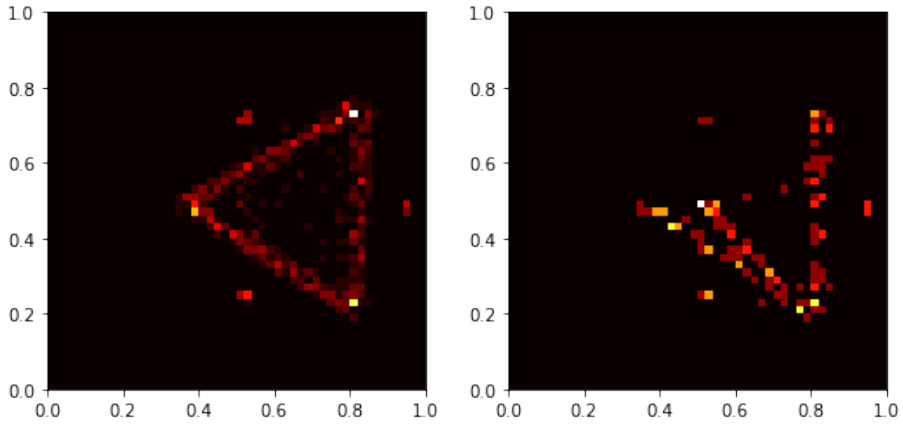
Regarding the impact the action targeting randomness might have, we expect the key point to be the target selection from the solo team agents, this will give the non-attacked team a huge advantage. In contrast, the impact provider agents have on this scenario will be close to none, since the imbalance just explained sways the game results a lot faster than the providers' actions results - delivering a resource to a member from a team that proceeds to heal the entire team - conjugated with the randomness of their targeting. We don't believe they will still have the potential to counteract the previous points, but they can certainly help, making the providers targeting a *"win more"* mechanism.

5.2 Test Scenarios Results

The balanced competition lasted for 105 seconds in comparison with the unbalanced competition that lasted only 25 seconds, as expected, creating a very unfavorable scenario for the team at disadvantage - the only team competing against two teams - with said team experiencing the only death.

Also, the position dispersion came out to be exactly what we expected, as can be observed in the Heatmaps in Figure 5.1. First note that Figure 5.1(a) that portrays the balanced competition has a lot more data points than Figure 5.1(b), due to the duration of the game and the limitation of actions an agent can take at a time. Nevertheless, we can observe that a 'V' shape is formed, revealing that the agents not considering another team as competitors influence the actions they take. In this case, in the unbalanced scenario, since they do not see one of the teams as competitors they don't get set as an objective to capture that team's flags, therefore do not even move towards that team's base.

For the attentive reader, there are mapped in both cases three dots alongside the most traveled spaces, these result from the presence of the neutral agents and reflect the resource nodes visible in Figure 4.1



(a) Balanced Competition **(b)** Unbalanced Competition

Figure 5.1: Position dispersion over full game time

6

Conclusion

Contents

Game characters are multilayered and complex. As already explained [Rato et al., 2019] we can identify that they have both narrative and gameplay relevance. Yet, in today's industry, socially capable and believable agents in large gameplay environments seem to be much more underdeveloped than their narrative counterparts. The latest big example of this was Cyberpunk 2077 [CD Projekt Red, 2020], having relatively bad reviews with one of the main complaints, being problems in the interactions with agents. Yes, their goals were maybe too ambitious, but the expectations players had of the game prior to its release prove that we should strive to improve our agents' social capabilities because there is demand for it.

Nevertheless, the gameplay and narrative layers of characters are usually interconnected and display multiple dependencies on one another. In *Batman: Arkham Asylum*, [Rocksteady Studios, 2009] Batman only being able to knock out other agents in-game because narratively he strives not to be a hypocrite in solving crimes through killing, which is a crime, is an example of narrative affecting gameplay from the roots of the design of the game. Both layers are built around this fact. Furthermore, it has become a trope to have player choices - done at a narrative level - limit player capabilities at a gameplay level, as simple as making the narrative choices an NPC considers correct result in them unlocking an item that the player can not get by any other means and having this item change the gameplay possibilities of what the player can or can not do.

Another proof of the interconnection of both these layers is the identification of *Ludonarrative Dissonance*. This suggests that agent behavior at a gameplay level that goes against its character's narrative is a culprit of breaking immersion and providing a worse game experience.

And we now believe that these dependencies influenced the proposed taxonomy. With this in mind, we will now review the proposed roles in a way that tries to decouple the gameplay from the narrative while trying the minimize the effects and limitations the use of our architecture at a gameplay level might have on its integration with the narrative.

6.1 Revising the Proposed Social Roles

6.1.1 Opposing SIAs

We defined and implemented the **Competitor** role, this is arguably the most common role an Agent will take in-game, the players usually need someone or something to play against. On top of that we proposed the **Stopper** and the **Collider** role. And while it is useful to distinguish multiple Agent applications for opposing other Agents at a meta-level, where we as designers consider the game as a whole, including all of its three layers. From an exclusive Gameplay comparison, these roles differ in only that, their application, and the important aspect, their functionality stays the same. Agents serve as obstacles for the player while representing this opposing role.

In the case of the **Stopper** the distinction made is the time of application, and the events that led to it. And since we aim to dynamically change their roles, this role becomes redundant. Since we can at any time give an agent the **Competitor** role, in a way that the exhibited behavior fit into the definition we gave the **Stopper**. As such we remove it from the proposal. This however doesn't change the usefulness of this type of role application. And we should keep using event triggers to apply this opposing role. And perhaps further work can be done not in defining the opposing role itself, but the types of transitions that lead to and from the application of this role. Be it a teammate turning on an agent due to friendly-fire, or an unrelated (socially) agent becoming adverse due to our Agent's behavior. Recall for example the Policeman from Grand Theft Auto V [Rockstar North, 2013], or Security Guards from Hitman 3 [IO Interactive, 2021], they become socially relevant to the player only when this "application trigger" happens.

The case of the **Collider** suffers from a similar problem, but instead of being concerned about *When* the application of the role happens, it is concerned about the *How*. As a result, we would be bringing a problem that can - and should - be solved at the Goal and Action definition level to Role definition, which same as before would lead to redundant definitions of actions, goals, and roles over each other. If we are looking into manipulating how an agent behaves toward another, using the solution to the previous problem - having different triggers - to apply this opposing role in conjunction with method overloading - having multiple ways to provide agents different sets of actions for the same sets of goals - we can achieve environments where similar agents have similar roles, and even similar capabilities (at a narrative layer) but behave differently. For example, consider an agent that owns a gun that can shoot at opposing agents. We can achieve what the **Collider** role proposed by defining different ways for the agent to use that gun. Consider a case where a traditional **Competitor** uses the gun to shoot enemies when competing to get a resource or an advantage, for example. Now if we want a use more representative of the **Collider**, trying to cause disadvantages to other players/agents, maybe the agent can instead shoot every time he spots an enemy.

As such we now believe that one opposing role is sufficient, and propose that we refer to it as an **Opposing SIA**. And encourage the study not of specifications this role can take but of the transitions 'to' and 'from' it, from the remaining roles as potential recognizable terms to aid designers communicate the different ideas of application of this role.

6.1.2 Allied SIAs

Of the proposed SIAs we implemented the Teammate role, but we also defined the **Subordinate** and the **Adviser** or **Helper**. As already identified in the proposition in section 3.1.1, and much like the opposing roles we previously discussed, at a gameplay level there is no difference between the **Teammate** and **Subordinate** roles. For example, let us compare a Companion from Pillars of Eternity [Obsidian

Entertainment, 2010] with a simple Subordinate military unit from Age of Empires 3 [Ensemble Studios, 2005]: the Companion from Pillars of Eternity is much more fledged at a narrative level than the military unit from Age Of Empires 3, but now imagine both of these games actual gameplay where you explore a map and fight enemies, in both games the player controls and orders both units in very similar ways, specifying where to position and how and when to attack. Exclusively at a gameplay level, these agents are essentially the same, achieve the goals set for the player, even if this goal is shared with the companion, having repercussions at the narrative layer. Nevertheless, when viewing all the layers simultaneously we believe the distinction to have value. But when defining a typology of roles exclusively at a gameplay level we must cut it to avoid redundancy.

The **Adviser** role on the other hand is nonexistent at a gameplay level. Yes, when it is present it affects the *gameplay experience*, but it does not exist in the gameplay layer, it affects the gameplay from the narrative layer. Adviser agents don't directly affect the gameplay environment, they indirectly guide the player's decisions in it, but never change the environment when performing an advising task. Because they're intended not to, it's a way of guiding the player agency, if they were to act and directly influence the gameplay environment the player agency would be diminished. Which is not what this role is designed to achieve.

6.1.3 Neutral SIAs

Of the proposed SIAs we implemented the **Provider** role and verified, as have many other games that the existence of agents that fulfill this role as gameplay mechanics can enrich gameplay experiences, imagine playing The Elder Scrolls V: Skyrim [Bethesda Game Studios, 2011] or similar role-playing games without the existence of merchants to whom you, as the player, can sell all the loot you have, and to whom you can buy handy resources and items; The gameplay experience would be arguably poorer since there is less the player can do.

Even though not implemented and tested we believe the **Challenger** role to be well defined, as it translates directly into the game mechanic of providing the player challenges as a result of interactions with other agents, usually resulting in the player receiving 'quests'. Nowadays there are two main types of **Challengers** in use, (a) those that heavily rely on the games' narrative to motivate the player to act, requiring handcrafted exposition and (b) those that completely automate this crafting, by making use of AI to generate said quests. In any case, how developers make use of this role, is up to them and the needs of their games.

Much like the **Adviser**, the **Commentator** role does not exist at an exclusive gameplay level. For example in a Fifa 22 [EA Sports, 2021] match with Commentators and a match without them, the gameplay stays the same, you simulate a soccer game. What changes is the game experience, by making use of commentators Fifa 22 heightens the player experience. But at a gameplay level, this role provides

nothing. Therefore, as we're only considering gameplay relevant roles we should not include this role in our taxonomy. More examples can be seen as previously mentioned in Darkest Dungeon [Red Hook Studios, 2015]. Without the narrator, the gameplay stays the same, but the game experience changes, meaning this role is not present at the gameplay layer. But should acknowledge its usefulness at the other layers, and suggest its consideration when viewing any game as a whole.

The **Background** role is another that although not explored in our testing we believe remains relevant at a gameplay level. This role's purpose is to remind developers that every agent that is put into the game should have a purpose for being there, and as such should display expected behavior, regardless of how minimal it is. For example in or game, we could add agents that surround the game arena, and react to the game, be it by displaying emotions through the use of emojis when an agent dies or captures a flag. Maybe they can also be affected by the game, by being able to be damaged by stray bullets that come out of the arena. Whatever the case, there are multiple ways to make the agents that portray this role behave, and we should choose the one that fits the overall atmosphere the game tries to bring the player to, be it a deadly game where even spectating agents can be killed or an emotional competition to be celebrated by a large group, much like soccer matches.

Host or **Referees** is arguably the hardest role to explore at a gameplay level since video games specifically usually do not delegate the verification and enforcement of rules to gameplay agents since it's much simpler and arguably clearer for players to have this rule enforcement be done by the computer in the background. The rules are what they are, normally there is no need to have an agent act to justify their enforcement. Nevertheless, the example of the unintended interaction in FIFA 94 [Extended Play Productions, 1993] proves that having an agent to enforce the rules instead of merely relying on background processes can heighten the game experience, by providing more scenarios for the player to act upon.

Lastly, we defined **Hostages** or **VIPs**, which was not practically tested, but since its definition translates into a game mechanic we are very confident that its existence can be used in a way that provides a better gameplay experience, by creating scenarios where the player can do everything that it could before, but now there are new constraints that change the gameplay. For example, imagine a first-person shooter where the agent can run around shooting at enemies and blow up the environment in a very consequence-free way. If you add Hostage agents that should not be shot, the agent now has to act in a more careful way, changing the game experience. In our game, this could be reflected in each team having an agent that needs to be rescued from the enemy's base, and it can affect the game by making the Cone agent have to hold fire when facing them, even if an eventual shot would hit an enemy.

6.2 Future Work and System Limitations

We believe that the use of the roles through our architecture is an acceptable method for establishing good gameplay experiences through believable agent behavior.

We acknowledge that to make use of this system developers are limited to having to implement a goal-action system, which can be overwhelming and depend on many other design decisions.

Another limitation resulting from someone implementing our gameplay system into their game, are the inevitable integration problems between the gameplay and narrative layers. By completely separating the narrative from the gameplay we allow the use of A.I. systems to help generate believable gameplay agents, since this way there is no big authoring load bottle-necking the development of a large scale social environment. But this in turn can lead to a problem that has been commonly posed under the name of *Ludonarrative Dissonance*, where what the agents do at a gameplay level conflicts with the narrative being portrayed. An example of this can be seen in game series like The Last of Us [Naughty Dog, 2013] and Uncharted [Naughty Dog, 2007] where during gameplay the player agent is okay with killing multiple enemies, while the game narrative suggests that it is something that the player should not do. But not doing it results more often than not in the player being punished by ”losing the game“ due to dying or not completing a mission on time, for example.

The next step for this work would be to help define ways of integrating the roles with one another and identifying cases of transitions from one role to another. Namely as the result of game mechanics. For example, we identify the fact that an agent shooting a Teammate can be a reasonable cause for their relationship to go from Teammates to Opposing or Neutral, with the mechanic here at hand being *Friendly fire*. If we now define a system that relates role transitions to game mechanics we can argue that we created a system that can help designers and developers communicate their vision of the game in easier and faster ways, setting a stone for a path with a standardized communication system.

Furthermore, to assert the value of the proposed taxonomy, the next evaluation step is to study its identification by players and designers. Having people identify our taxonomy in games can reveal its flaws and strengths, and it can be done two ways: studying already existing games and seeing how the games' systems correlate with the taxonomy; and by developing more games that incorporate the taxonomy as we defined it and evaluating how well it is perceived by the test subjects.

Bibliography

References

[Bartle, 2004] Bartle, R. A. (2004). *Designing virtual worlds*. New Riders.

[Isla, 2005] Isla, D. (2005). Handling complexity in the halo 2 ai. Game Developers Conference 2005.

[Liapis et al., 2013] Liapis, A., Yannakakis, G. N., and Togelius, J. (2013). Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of the 8th Conference on the Foundations of Digital Games*, pages 213–220.

[Orkin, 2006] Orkin, J. (2006). Three states and a plan: the ai of fear. In *Game developers conference*, volume 2006, page 4.

[Rato et al., 2019] Rato, D., Mascarenhas, S., and Prada, R. (2019). Societies in games: How do players perceive groups of game characters? In *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, CHI PLAY '19 Extended Abstracts, page 639–644, New York, NY, USA. Association for Computing Machinery.

[Rivera et al., 2012] Rivera, G., Hullett, K., and Whitehead, J. (2012). Enemy npc design patterns in shooter games. In *Proceedings of the First Workshop on Design Patterns in Games*, DPG '12, New York, NY, USA. Association for Computing Machinery.

[Ruela and Guimarães, 2017] Ruela, A. S. and Guimarães, F. G. (2017). Procedural generation of non-player characters in massively multiplayer online strategy games. *Soft Computing*, 21(23):7005–7020.

[Short and Adams, 2017] Short, T. and Adams, T. (2017). *Procedural generation in game design*. CRC Press.

[Sweetser and Wiles, 2002] Sweetser, P. and Wiles, J. (2002). Current ai in games: A review. *Australian Journal of Intelligent Information Processing Systems*, 8(1):24–42.

[Treanor et al., 2015] Treanor, M., Zook, A., Eladhari, M. P., Togelius, J., Smith, G., Cook, M., Thompson, T., Magerko, B., Levine, J., and Smith, A. (2015). AI-based game design patterns.

[Warpefelt, 2016] Warpefelt, H. (2016). *The Non-Player Character : Exploring the believability of NPC presentation and behavior*. PhD thesis, Stockholm University, Department of Computer and Systems Sciences. At the time of the doctoral defense, the following papers were unpublished and had a status as follows: Paper 5: Manuscript. Paper 6: Manuscript. .

Games

[Bay 12 Games, 2006] Bay 12 Games (”2006”). ”Dwarf Fortress”.

[Bethesda Game Studios, 2011] Bethesda Game Studios (”2011”). ”The Elder Scrolls V: Skyrim”.

[Bethesda Softworks, 1994] Bethesda Softworks (”1994”). ”The Elder Scrolls: Arena”.

[BioWare, 2014] BioWare (”2014”). ”Dragon Age: Inquisition”.

[Blizzard Entertainment, 2014] Blizzard Entertainment (”2014”). ”Heartstone: Heroes of Warcraft”.

[Blizzard Entertainment, Saffire, 1998] Blizzard Entertainment, Saffire (”1998”). ”StarCraft”.

[Blue Byte Software, 2015] Blue Byte Software, R. (”2015”). ”Anno 2205”.

[Bungie, 2004] Bungie (”2004”). ”Halo 2”.

[CD Projekt Red, 2020] CD Projekt Red (”2020”). ”Cyberpunk 2077”.

[Creative Assembly, 2020] Creative Assembly, Creative Assembly Sofia, F. (”2020”). ”Total War Saga: Troy ”.

[DMA Design, 1997] DMA Design (”1997”). ”Grand Theft Auto”.

[EA Sports, 2021] EA Sports, E. (2021”). ”FIFA 22”.

[Electronic Arts, Griptonite Games, 2001] Electronic Arts, Griptonite Games (”2001”). ”Harry Potter and the Philosopher’s Stone”.

[Ensemble Studios, 2005] Ensemble Studios (”2005”). ”Age of Empires 3”.

[Evolutionary Games, 2010] Evolutionary Games (”2010”). ”Galactic Arms Race”.

[Extended Play Productions, 1993] Extended Play Productions (”1993”). ”FIFA International Soccer”.

[Game Freak, 1996] Game Freak (”1996”). ”Pokémon Red & Blue”.

[Infinity Ward, 2011] Infinity Ward, Treyarch, S. R. N. (”2011”). ”Call of Duty: Modern Warfare 3”.

[Insomniac Games, 2002] Insomniac Games (”2002”). ”Ratchet & Clank”.

[IO Interactive, 2000] IO Interactive (”2000”). ”Hitman: Codename 47”.

[IO Interactive, 2021] IO Interactive (”2021”). ”Hitman 3”.

[MicroProse, 1991] MicroProse, K. (”1991”). ”Sid Meier’s Civilization”.

[Mojang Studios, 2009] Mojang Studios (”2009”). ”Minecraft”.

[Naughty Dog, 2001] Naughty Dog (”2001”). ”Jak and Daxter: The Precursor Legacy”.

[Naughty Dog, 2007] Naughty Dog (”2007”). Uncharted.

[Naughty Dog, 2013] Naughty Dog (”2013”). ”The Last of Us”.

[Nintendo Entertainment Analysis & Development, 1992] Nintendo Entertainment Analysis & Development (”1992”). ”Super Mario Kart”.

[Obsidian Entertainment, 2010] Obsidian Entertainment (”2010”). ”Pillars of Eternity”.

[Red Hook Studios, 2015] Red Hook Studios (”2015”). ”Darkest Dungeon”.

[Relentless Software, 2005] Relentless Software (”2005”). ”Buzz!: The Music Quiz”.

[Rockstar North, 2013] Rockstar North, Rockstar San Diego, R. R. R. R. (”2013”). ”Grand Theft Auto V”.

[Rocksteady Studios, 2009] Rocksteady Studios, Warner Bros. Games Montréal, S. N. A. (”2009”). Batman: Arkham Asylum.

[SCE Japan Studio, 2001] SCE Japan Studio (”2001”). ”Ico”.

[Sonic Team, 1991] Sonic Team (”1991”). ”Sonic the Hedgehog”.

[Supergiant Games, 2018] Supergiant Games (”2018”). ”Hades”.

[Treyarch, 2010] Treyarch (”2010”). ”Call of Duty: Black Ops”.

[Ubisoft Montreal, 2007] Ubisoft Montreal (”2007”). ”Assassin’s Creed”.

[Ubisoft Montreal, 2020] Ubisoft Montreal (”2020”). ”Assassin’s Creed Valhalla”.

[Warhorse Studios, 2018] Warhorse Studios (”2018”). ”Kingdom Come: Deliverance”.