Level Of Detail Based AI Adaptation for Agents in Video Games

Ghulam Mahdi, Yannick Francillette, Abdelkader Gouaich, Fabien Michel, Nadia Hocine

Level Of Detail Based AI Adaptation for Agents in Video Games

Ghulam Mahdi, Yannick Francillette, Abdelkader Gouaich, Fabien Michel and Nadia Hocine
LIRMM, Université Montpellier II, UMR 5506 - CC 477 161 rue Ada
34095 Montpellier Cedex 5 France
{*mahdi, francillet, gouaich, fmichel, hocine@lirmm.fr*

Keywords: Agents, Game AI, Level of Detail, Adaptation, MAS, Organization, Video games, Frame rate

Abstract: This paper suggests multi-agent systems (MASs) for implementing game artificial intelligence (AI) for video games. One of main hindrances against using MASs technology in video games has been the real-time constraints for frame rendering. In order to deal with the real-time constraints, we introduce an adaptation-oriented approach for maintaining frame rate in acceptable ranges. The adaptation approach is inspired from the level of detail (LoD) technique in 3D graphics. We introduce agent organizations for defining different roles of agents in game AI. The computational requirements of agent roles have been prioritized according to their functional roles in a game. In this way, adapting computational requirements of game AI works as a means for maintaining frame rate in acceptable ranges. The proposed approach has been evaluated through a pilot experiment by using a proof of concept game. The pilot experiment shows that LoD based adaptation allows maintaining frame rate in acceptable ranges and therefore enhancing the quality of service.

1 Introduction

Video games are considered nowadays as a mainstream entertainment and cultural industry. Once a small and focused discipline, it has turned into a large industry with applications in education and training.

As a simplification, a video game can be considered as a computational simulation of a set of game characters that interacts with one or several players. Each individual game character achieves particular task in a game and the overall game environment is governed by rules of the game.

Until recently graphical rendering has been considered as the Holy Grail for this sector. The main purpose has been to increase quality and naturalness of graphics. Research on game artificial intelligence (or behavior of game characters) was not on the agenda since simple models based on finite state machines were considered sufficient to meet requirements of gameplay and player experience. The development of successful titles such as Neverwinter Nights (BioWare, 2002), Black and White (Studios, 2001), Max Payne (Remendy, 2001) and Left 4 Dead 2 (Valve, 2008) have challenged this supposition. These titles, among others, have introduced more complex, organized and adaptive behaviors by using more sophisticated artificial intelligence techniques. In other words, they have clearly demonstrated the benefits of moving beyond simple behaviors.

It is worth noting, that despite using the term "artificial intelligence" in game sector and academia, this term refers to different concepts. In fact, the term AI in games often refers to any model and algorithm that steer behaviors of game entities. On the other hand, Artificial Intelligence (AI) as a scientific discipline studies and designs intelligent agents, where an agent is defined as a system that perceives its environment and takes actions that maximize its chances of success (Russell and Norvig, 2010). Despite having different constraints and objectives, collaboration between the game sector and AI community has been illustrated in several works such as (Rabin, 2002; Charles, 2007; Millington and Funge, 2009). Despite these improvements, AI in the games still remained largely simple and basic one. As a result, usually finite state machines (FSMs) and other simpler algorithms have been used for modeling behaviors of game characters (Niederberger, 2005).

According to Neiderberger et al. (Niederberger and Gross, 2005) current game characters exhibit behaviors that are too rudimentary to seem realistic ones. This may decrease the player sense of alief (Clark, 2008) that is an important feature to enable players' immersion in the game world. Orkin (Orkin, 2006) also states that current game characters lack flexibility and there is need of more flexible planning

for complex behaviors in game characters. In addition to that, depending on the nature and type of a game, there can be hundreds of secondary game characters that may face competition over getting computational resources for executing their actions.

Multi-agent systems (MAS) have been growing prevalence and increased usage in different research and applied domains. One can notice that there is a natural mapping between core concepts of MAS and games. In fact, agents are micro units in MAS that execute own behavior to reach their goals. The overall function of the system is observed at the macro level as the interaction among all entities of the micro level. Thus, conceptually each game character can be considered as an agent and the overall game as a multi-agent system interacting with one or several players.

Dignum et al. (Dignum et al., 2009b) argue for using the potential of MASs by incorporating agents into games for improving game AI. In order to get an idea of how closely MAS and games are related, one can observe that apart from conference papers, since 2009 there has been an annual workshop in major academic conferences such as AAMAS (Dignum et al., 2009a; Dignum, 2011; Dechesne et al., 2012) and ECAI 2012.

On the other hand, (Dignum et al., 2009b) note that modeling of game characters around agents may require relatively more resources so it may not meet real-time constraints of game engines where fundamental concern is rendering of frames with an acceptable frequency. This is one of the major challenges that prevents using agents in games.

Here in this paper, we address the question of how to build game character with agents without deteriorating player experience metrics and constraints of game quality of service (QoS). We propose modeling of game characters around agents and their dynamical adaptation to the computational resources for maintaining acceptable frame rate. The rationale behind our approach is to use the organization of MAS as a means to express different priorities to agents depending on their role. Whenever the quality of game is not met, agents with less important roles are asked to adopt simpler behaviors. Consequently, agents with central roles can continue executing their behavior and the overall player experience is not affected.

We experimentally show that our approach maintains an acceptable player experience when compared with a simple MAS approach without behavior adaptation.

The rest of the paper is organized as follows. Section 2 discusses the motivations for the work. Section 3 is about introducing Level of Detail (LoD) technique, its foundations in computer graphics and its recent usage in game AI. In section 4, we discuss our proposed approach, where we present LoD as an adaptation technique for game AI. Section 4 describes the experimental framework and the game which is used to evaluate performance of our approach ; then in section 5, we discuss the results of our experiments and prototype results of user evaluation. Section 6 discusses some relevant works in the domain and finally, we conclude the paper in Section 7.

2 Motivations

2.1 Behavior modeling around agents

Schreiner (Schreiner, 2003) points out the lack of pro-activeness in case of first person shooter games. Currently the enemies do not react until they find player entering into their areas. This reactive approach of enemies often results in repetitiveness of same actions by the player as well as enemies as they know that all they can to react to one another's moves. The repetitiveness may result in lack of dynamism and unnaturalness for a player after few trials. The proactive nature of non-player characters (NPCs) may enable enemies to hunt the player dynamically so that there would be different hunting strategies and they keep changing as situation or difficulty levels change.

In case of the games like *Sims*, a player creates a city for Sims to live in or a theme park to visit. Once the city's initial stage of build up is completed, each Sim do different things to make itself happy. Sims have means to evaluate their state of happiness along with set of behaviors for doing what they want to achieve the happiness(Delgado-Mata and Ibáñez-Martínez, 2008). When the Sims are not directly controlled by the player, the autonomous notion of agents can be suggested to make them act in more human-like manner ; for example the Sims preferring swimming over going to library would be able to choose swimming more frequently than going to library. This human-like decision making feature of doing things autonomously can make games more realistic and fun-oriented.

Teamwork has been considered as one of important features in many games including *Counter-Strike (CS)* (Schreiner, 2003). For example, in *CS* players can join teams of terrorists, counter-terrorists or become spectators. In each team, the players play in quite well-organized manner with the mission objective of trying to eliminate the opposing team in given time limit. Although the idea of playing in teams can be observed in non-team members as well but that is most often on irregular and self-serving basis. Mi-

crosoft's *Halo 3* game can be considered as another example of the games featuring notion of teamwork (Mott, 2009). In *Halo 3* enemies travel and act in groups and in case the player hides for a minute, most of the group members stay behind to guard their current location, just couple of them get assigned to search the player. Apart from searching, *Halo 3* uses teamwork in other situations as well, particularly when enemies use suppressing fire on the player meanwhile facilitating other allies to find a better vantage point or to take cover.

The notions teamwork, autonomy, reactivity and pro-activeness can be considered as some of most important features which can be exploited to increase realism and fun-orientatedness in video games. There has been quite a few video games which incorporate some of these features. Making all these features accessible to NPCs requires the kind of abstraction as provided by Multi-agent systems.

2.2 Multi-agent systems

Multi-agent systems can be defined as an agglomerations or artificial societies of "agents" ; however it is quite difficult to precisely define agents due to wide differences in agent researchers on the definition. Wooldridge et al. (Wooldridge and Jennings, 1995) characterizes agents through some well recognized common features of autonomy, reactivity, proactiveness and, sociability, which we discuss as under :

1. **Autonomy :** Autonomy is mean that agents are capable of acting independently on their own, without any external influence. This is one of most important and distinguishing features of agents by which they exhibit their control over their internal state.
2. **Reactivity :** Reactivity implies that agents respond to the changes occurring in the environment in due time. In other words for some stimulus there will be corresponding response. This feature of agents make them aware as well as responsible to the environment that which behavior they can adopt to fulfill their design objectives in some particular situation.
3. **Pro-activeness :** Pro-activeness makes agents to behave actively instead of passively. To fulfill their design objectives agents identify the opportunities and do subsequent actions for achieving their respective purposes. Pro-activeness enables agents for making initiatives to achieve goals instead of just reactively responding events in the environment.
4. **Sociability :** It is the ability of an agent by which it interacts with other agents or humans in the system. Communication is the means to coordinate agent actions, there can be different means of communication among agents.

2.3 Answering the challenge of player experience

Player's interaction with a game can be considered as one of critical performance metrics for determining quality game experience. The level of interaction can be evaluated through the number of frames displayed in a second. It is estimated that for interactive computer graphics, a new picture is to be rendered between 25 to 30 times in a second. Hence the process of refreshing a frame need to be done in 30 milliseconds or so. The time available to a frame is divided between rendering and the AI parts of the game. In the time, the simulation of whole game world requires a real-time behavior generation of game entities and their rendering.

A significant increase in the computational requirements of a frame would cause delay or lag in the game and make interaction difficult for the player. The lag in the interaction may degrade frame rate and would eventually disrupt the players' ability to synchronize their actions with the game. Ideally for a player the interactivity would mean that there is no lag time in game response. Cumulative game lag can be said as a sum total of following three factors of lags namely player input, rendering and AI. Cumulative game lag can be represented as linear combination of all three factors in following manner :

`Cumulative game lag = Player input delay + Graphical rendering delay + Artificial intelligence delay`

Although any one of above factors can be sufficient to deteriorate game flow and interactivity. In this paper we are focused on the AI factor. The primary motivations for the selecting AI is due to the fact that among three factors the AI factor can be said as one of most ignored areas in current research. Even in the case of time distribution normally game AI gets significantly short time than rendering. According to Niederberger et al., normally between 5 and 10 percent of the overall CPU time is reserved for AI in a real-time game (Niederberger and Gross, 2002).

Here our focus on minimizing lag value in game AI would ensure availability of sufficient computational resources required for simulating game entities and above all averting undesirable side-effects on game AI's part. To minimize lag in AI part of video

games one of very first steps is to identify the factors which can influence frame rate.

2.3.1 Game frame rate

Usually a frame gets around 30 milliseconds as "frame time" for both rendering and updating game AI from the game engine. The frame time for updating game AI is used for computing and simulating the AI behaviors of the NPCs in a frame.

The elements influencing "frame time" within the scope of game AI mainly include computational requirements of NPCs and their quantity in a frame. The frame simulating large number of NPCs or more complex behaviors would require more than expected computational resources for game AI. These elements of video game performance can be considered as the deciding factors in generation of frames from game AI's perspective. In such situation a game may result in a lower frame rate as game entities would take more time for simulating agents which require significant computational resources. On the other hand, if the same agents run on another platform (with increased resources), it is likely to maintain required frame rate. In other words, any changes in the computational requirements of NPCs and their quantity would necessarily bring corresponding adjustments in the frame rate. Ignoring changing requirement of agents would lead to undesired consequences of either irrelevant AI or lag into it. In case of more than required computational resources, game AI can be programmed to improve its realism and naturalness by introducing additional features into it. Using additional features of game AI can prevent unnecessary wastage of resources.

Ideally game engines need to provide some scalable and viable means which could provide QoS support in case there is a change in the performance requirement of agents. Absence of such QoS support methods may lead to lag in updating game AI and subsequently reduced frame rate which would eventually result in lack of interest and frustration for a game player.

2.3.2 Bidirectional adaptation of agents' behavioral "complexity" as a means of QoS support

In order to avoid degraded frame rate or resource wastage we need a flexible mechanism which can handle performance requirements of NPCs at runtime. A bidirectional adaptation mechanism can serve the purpose. The adaptation need to lessen its performance requirements when it notices a decrease in the frame rate and allows requirements in case of otherwise. This decrease and increase in the performance requirements can be made possible through the conditional permission to execute certain features in one case while their prohibition in the other. Here QoS support would be about suggesting a mechanism for graceful degradation and progressive enhancement of agent behaviors.

Graceful degradation is meant to provide an alternative version of the game AI in case the system gets overloaded. In our context, it would take up frame rate as higher as possible. Progressive enhancement can be considered as the other side of the QoS coin in video games. Progressive enhancement works the other way around of graceful degradation. It starts with a baseline version of game AI and then feature enhancement goes till the most sophisticated version by maintaining frame rate throughout the process. Progressive enhancement can work in the situations where despite sufficient number of frames there is no change in the game AI, here it can progressively increase game AI features.

In our context of game AI in video games, both graceful degradation and progressive enhancement can be applied to maintain QoS in most of the situations. The approach would reserve advanced game features of game AI and/or gameplay subject to availability of sufficient computational resources. One of main differences between graceful degradation and progressive enhancement lies in determining the point where each one begins and here the LoD technique can provide relatively automatic and easy solution for its determination. The LoD technique also provides ways by which game agents can have different representations.

3 Level of Detail

Level of detail (LoD) technique suggests to modulate the complexity of a 3D object representation according to the distance from it is viewed or any other criteria (Luebke et al., 2002). The technique, as introduced by James Clark (Clark, 1976), is meant to manage processing load on graphics pipeline while delivering an acceptable quality of images. He suggested that structuring the rendering details can optimize the processing quality if a 3D object's visible quality and details are made to have a correspondence with the distance from it is viewed.

In figure 1, we can observe the rational of Clark, here with changing the number of vertices we see the change in quality and visualization of a sphere. In other words, LoD technique trades spatial fidelity for

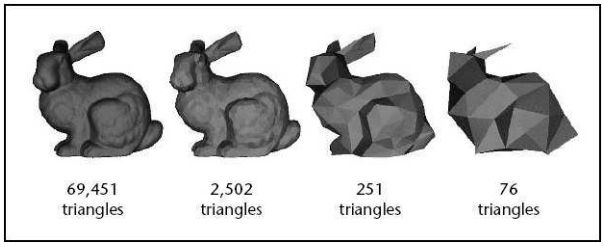

FIGURE 1: Basic concept of LoD : An object is simplified by different representations through varying number of polygons čiteluebke2001perceptually

temporal fidelity in a way that a less complex model would be instantly rendered by compromising over its coarser representation.

3.1 How LoD technique works ?

Since geometric datasets are usually too large in data size and complex in terms of time and computational resource demands so their rendering can become a tedious and time consuming process. The LoD approach suggests different representations of a 3D object model by varying in the details and geometrical complexity. The geometrical complexity and time demands of a graphical object comes can be determined in terms of the number of polygons used for its rendering. Usually an object model having more number of polygons consumes more time and resources than the one having less number of polygons in its rendering. Although there can be other factors involved in the complexity and resource demand of a graphical model of an object but the relations between polygonal quantity and resource consumption are generally considered as established ones (Deering, 1993). For example, one can clearly determine the difference in rendering quality by observing following figure 1 and can draw general conclusion that how number of polygons affect the rendering quality of an image's graphical representational model.

Once these different representations of a model are on hand (as shown in the above figure for our example), the LoD technique will suggest their selection at a particular time point based on particular positive selection bias. The latter can be their size, camera distance or any other criteria. The net benefit of the approach would be that only necessary objects that will get maximum amount of processor time and resources at one time point while others are either shown with less details or just ignored at that time point. These objects can get less shared resources when the situation changes and their importance is substituted by other objects. Basically, the approach is inspired by rendering of the polygonal geometry of object models by using coarser LoDs for distant or invisible objects (David Luebke, 2003).

3.2 LoD in game AI

The LoD technique can be used in game AI for selecting most interesting agents and correspondingly priorirtizing them for computational resources distribution. The priority in computational resource distribution would ensure that the selected agents would have access to the game AI which requires more computational resources. The agents having lower priority value would get the share in computational resources required for a basic version of the game AI. In other words, the technique focuses on finding relatively interesting or irrelevant characters in a scene and progressively upgrade or turn down their behavioral details. Evidently, the notion of LoD in game AI may not be regarded as an exact transfer of the technique from rendering part as the nature and issues particular to the game AI part would be different, despite the fact that basic concept remains the same. Here we need to address some specific issues which are particularly related to the AI part of the games.

Objects varying in priorities and then in simulating most relevant ones is common to both rendering and AI based LoD, the primary difficulty lies in the notion of importance and its different representations. The notion of distance or visibility may not be in the same form or as clear as it is in the rendering part. Neiderberger et al. (Niederberger and Gross, 2005) argue that invisible objects do not stop living even when they are out of visible area. Moreover, in some situations, even the objects which are only partially visible in a scenario may need more detailed AI compared with the ones which are currently visible. Hence the distance from the camera matters as much as the behavioral functionality in a scenario. Here the notion of importance may not be resolved solely by the camera distance. In addition to that we need to address the problem of how to provide different representations of an agent.

Hence, to deal with this problem, we suggest using agent organizations as a means to provide different representations of agents and decide their importance with respect to other agents in the game AI.

4 Related works

In 2002, Brockington (Brockington, 2002) presented a paper on extending the usage of existing computer graphics based LoD to game AI. Brockington uses LoD for the *Neverwinter Nights(NWN)* game.

He divides *NWN* game creatures under five classifications for determining their update frequencies. The classification determines a creature's LoD and subsequently percentage of CPU time for each LoD. The approach argues the interest of dedicating most of CPU time (60%, to be precise) for computing best algorithms to *Player characters (PCs)* as they always remain on the screen. The CPU time percentage and priority get decreased with increased LoD value according to the classification of game creatures.

Wißner et al. (Wißner et al., 2010) have tried to generalize the approach by extending LoD technique to the navigation, movement updates, collision avoidance and behavior execution of game entities based on 8 LoD levels. The approach's significant features include the diversity and heterogeneity of behaviors which can be reduced with it. Moreover, the behavior simplification is applied at the behavior execution stage rather than during the behavior selection process. Another significant feature of the approach is the addition of visibility factor besides traditional camera distance based LoD implementations. Explaining visibility criterion besides camera distance, the authors argue that some situations may make agents relevant and important enough for showing their full behavioral details regardless of their camera position.

Kistler et al. (Kistler et al., 2010) extend the work by (Wißner et al., 2010) using the same parameters. The only difference between these works is the Kistler et al. 's detailed explanation of their implementation of the approach on *Virtual Beer Garden* while Wißner et al. (Wißner et al., 2010) are more interested in *Augsburg3D* along with addition of two value of LoDs making it a total of 10 LoDs which were 8 in the earlier case.

Osborne et al. (Osborne and Dickinson, 2010) introduce a hierarchical approach for presenting agent behavior details according to the camera distance. The main problem addressed by this work is the simulation of large number of agents acting in groups.

Neiderberger et al. (Niederberger and Gross, 2005) 's work can be considered as one of the most comprehensive works in the domain. This work not only applies LoD on both individual and collective behavior levels, but also provides a number of 21 LoDs, which is the highest number we encountered in the literature. They suggest navigation, path planning and collision avoidance as AI behaviors of game entities. LoD is also applied for scheduling of agents, collision avoidance, path planning and group decisions of agents. The LoD approach is implemented on the combined distance and visibility factors for effective selection of entities. A special scheduling algorithm distributes simulation time to the agents depending on their visibility and camera distance.

5 LoD framework

The framework proposed uses the LoD technique to provide different levels of behaviors for an agent. Each level varies in computational overhead and behavioral complexity. To address the challenge of frame rate, the best possible level is allowed to carry out its tasks while the more complex behaviors are subject to availability of additional computational resources. When the frame rate degrades, a behavior requiring less computational resources is permitted. Besides, to assign priorities amongs agent behaviors, we use an organizational model so that priorities are related to roles.

5.1 Organizational model of agents

Our LoD framework uses agent organization to address two issues :

- Organizing and representing the different behaviors of an agent so that, whenever there is a change in the frame rate, a particular representation of an agent is always available.
- Evaluating the relative importance of an agent in a particular situation in order to select the most appropriate agents' representations. That is, when the system is overloaded, the organization settings are used to decide which agents can be abandoned altogether since they are less important than others in the situation.

So we follow the AGR model (Ferber et al., 2003) for the organizational structure of agents. Here we briefly define the AGR model. The AGR (Agent, Group, Role) model advocates organization as a primary concept for building large scale and heterogeneous systems. The model does not focus on the internal architecture nor the behavior of individual agents but suggests organization as a structural relationship between collection of agents. The AGR model defines three main concepts as its basis for an organizational structure : agent, as an active and communicating entity ; groups are comprised of agents in the set by tagging them under a collection ; finally an agent's functional representation in a group is given by defining its role.

The AGR model helps us to define how different behaviors can be glued together to construct an agent role in an agent society. Different combinations of behaviors would create distinct roles which would serve as representations of an agent. The behavioral bonding provided by AGR allows us to have different rep-

resentations of an agent through multiple roles. The model provides an organizational methodology which can be used to program agents which can have different representations to act within a justifiable time according to the available computational resources.

The way AGR associates different roles in a group makes it convenient to define importance of a role in a particular situation. Here agent roles can be associated together according to their functional roles or/and availability of computational resources in a game environment. Taking an example, when a group of agents moves to a particular point, the path finding result of each agent would be almost the same. This repeatability of the path finding algorithm can be avoided, if we can glue all agents in the collection in some organizational structure. Here our organizational approach can guide us to group different game entities according to their roles. In this example of path finding, some specific agents having group leader role would be allowed to access particular behavior while others agents would just follow the group leader.

Organization is also used for quantifying and filtering agent roles in a frame instead of normally used distance or size measures. A programmer specifies configuration file lists defining the allocated computational resources for the agent roles according to our model of organization. The game engine uses the configuration file based for distributing the given computational resources to agents. The game engine estimates an elementary value for the frame rate as a starting point and then uses it as a metric of shifting the QoS support modes for the agents AI.

5.2 QoS support modes

An agent requires a certain number of computational resource units for carrying out particular set of actions in a particular role. These resource requirements of an agent are variable and depend on the computations involved in a role. If a handful agent play roles which require realtively more computational resources than affordable by the game engine it would necessarily result in degraded frame rate. On the other hand, if the game engine can afford to provide sufficient computational resources it would be better to use them for providing higher levels of game AI rather than simply wasting them.

A QoS support mode associates the frame rate with the computational resources distribution of game AI agents. The approach provides several levels of QoS support by encompassing multiple modes of game AI. The game engine defines a threshold level of frame time for each mode and any noticable change in the frame time would switch the mode and subsequently computational requirements of game AI. In other words, a QoS support mode governs the policy by which computational resources are distributed to the game AI agents in a frame. The game engine uses multiple QoS support modes for determining which agents would be allowed to play a specific role at a particular time. The approach provides QoS support by adding flexibility and differentiation levels for game engine according to the available computational resources for a frame.

The available resource units for an agent are determined by the mode in which it is operating. Once an agent gets resource units awarded, it can access a particular role in the organisational based agent society. An increase or decrease in frame time switches the particular mode on or off. So that depending on the mode there is a difference in the QoS support and role access for the agents. This way, the modes approach will not fumble optimal utilization of computational resources with increasing demand of resources as there would be QoS support for all possible situations in form of relevant modes. By switching QoS modes, our approach provides programmers the flexibility and versatility to meet various time performance needs for the agents.

5.3 Agent life cycle model : Perception-Deliberation-Action Model

Normally, an agent model defines a set of activites and the data and control flow among them as a general methodology for carrying out any task. The set of activities and their interactions would define the life cycle of an agent. A brief description of agent activities is given as under :

1. **Perception :** It is about observing and getting a sensing stimulus from the game environment. On receiving a stimulus, the perception mechanism decides further meaning of the stimulus. Moreover, the perception mechanism examines and determines whether to take into account the sensing stimulus as an input or ignore it if found irrelevant. Assigning computational resources in terms of tokens would allow developers to specify the time taken by agents for perceiving their environment.
2. **Deliberation :** This activity can be considered as the central process in the agent life cycle. The main purpose of the activity is to determine the next action of an agent. An agent deliberates according to its perception of the environment and

its deliberation model. Depending upon the context and requirements, an agent's deliberation can be very simple like stimulus-response activity for reactive agents to very complex ones using spatial reasoning or any other algorithms. Because our game agent model is independent of any particular deliberation approach, game programmers will be able to implement any deliberation methodology.

3. **Action :** The action step provides means for an agent to select and carry out an action in its environment. Considering possible outcomes of the deliberation step, an agent selects one or more actions to perform in its environment. Once an agent selects an action, it is executed by updating its state or the environment. Depending on the available computational resources, an action is assigned a specific number of tokens. The tokens can be considered as the computational resources required for changing the agent's state or making the effects of the action appear in the environment.

FIGURE 2: Agent life cycle

Agent life cycle is a dynamic, iterative process of input handling, incorporating deliberation mechanism and carrying out actions in the agent environment. The agents would need computational resources to carry out a particular activity. The agents need to consider the computational requirements involved for performing any task. The life cycle of an agent can be summarized in figure 2. The figure describes the life cycle of an agent as it evolves and shows the process that the agent description supports for input, inference, and output modules in the agent environment.

5.4 Valued sensing and action model for game agents

Game agents use different sensing and actions to carry out their tasks corresponding to their roles in the organization. Each sensing and action task requires some computational resources for its execution. In other words, behavioral tasks come with certain cost in terms of resources. The execution cost of a task varies according to the computations involved in it. For example, a random move task could has a low cost compared to a path finding task requiring more CPU cycles. So, agent behaviors come with an absolute cost in terms of CPU cycles.

Distributing computational resources with an absolute measurement of agent behaviors would make it practically impossible to program game agents as differences in resources for different game platforms would require customized programming for every single hardware configuration. Moreover, the absolute cost measurement of behaviors would not help in differentiating among different versions of behaviors and later selecting a cost effective one as it would require comparisons of memory and CPU usage between different behaviors. Therefore we propose "resource tokens" as an abstract notion of computational resources.

5.4.1 Tokens

Here in our proposition, we use the logical notion of computational resources in our framework. For example, the logical time associates the wall clock time and the QoS modes. The wall clock time measures the execution time of a task on physical clock, for example a move action may require 40 milliseconds on a specific processor, while a path finding action takes 120 milliseconds on the same processor. A correspondence between the logical and physical notion of computational resources would make it their distribution an easier job for the programmers.

A "token" can be described as an abstract measurement unit for the notion of computational resources for weighing agent behaviors. The idea relies on providing an independent behavior measurement abstraction which is only related to the complexity of the behaviors, and not to the hardware configuration.

Each sensing and action task is assigned a value in terms of tokens through a configuration file. So, the programmer relatively analyses and evaluates each task with respect to other tasks and bounds their execution according to a determined number of tokens. The game engine evaluates which agents can be requested in a particular game loop iteration and assigns them tokens by reading the configuration file. The token assignment is based on the computational resources based relative evaluation of agent tasks. This relative value is assigned as per the absolute computational cost of different tasks in the game. For example, a move action might take 2 tokens of logical time and path finding action takes 6 tokens and after executing these two actions 160 milliseconds of physical time might have passed.

5.5 Agent delegation model

Delegation can be explained as a request of one entity to another for performing some actions on behalf of the former. Our approach generates and pro-

cesses actions by delegating them to the game engine as services. The action delegation model is based on the notion of providing standard and consistent mechanisms to generate and process actions independently from any specific hardware configurations. Basically, our delegation model for agents provides a resource request mechanism for access to the sensing and action services from the game engine. The life cycle of

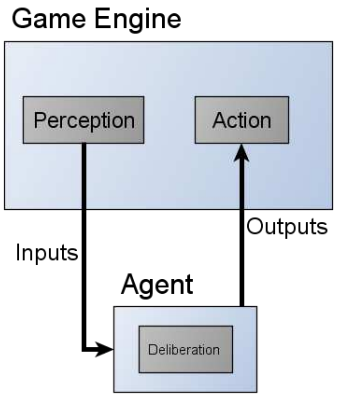

FIGURE 3: Agent life cycle using delegation

an agent using delegation can be summarized in following figure 3

In fact, the interest of the delegation model relies on the idea that in order to limit the incompatibility of computational resources among different game platforms, we need another entity for carrying out tasks delegated by agents. In our case, the game engine plays that role and, ensures availability of specific services and their token-based distribution to the game agents. So, an agent can request for delegating the sensing and action tasks to the game engine according to its available number of tokens. But thanks to delegation, when computing resources are lacking, the game engine can now answer with a simplified version of the same service requiring less computations (i.e. tokens) by using our LoD approach. This request and grant model of services delegation in the context of LoD mainly requires two things :

1. **Tokens :** Tokens are relative measurement abstraction units which are used to differentiate between different costs of behaviors.
2. **Services library :** A services library is maintained by the game engine for assembling all services and their corresponding cost in terms of tokens. In this way, once an agent submits a request, the game engine can easily search and grant or refuse the service according to the cost of the service and available tokens to the agent.

In our service delegation model, the services must associate with a source role for receiving a notification. This service and role bonding association provides an important benefit : request notifications are sent to only those services that want to receive them for example, decoration clouds can not send pathfinding service request. The main benefit of this design is that the perception and action AI that processes these services is cleanly separated from the deliberation logic which generates these requests. This separation of perception/action and deliberation logic serves our purposes of delegating a game agent's processing through a separate piece of code namely a service.

Agent actions are decomposed into small independent services. The approach ensures to carry out the separation of concerns in different hardware configurations. The game engine maintains a collection of services and correspondingly required tokens for the sensing and action services. As the services library would be used for selecting appropriate services according to the LoD level so services having different simplified and advanced versions are provided in the library ; however their degree of resource requirements is associated with the number of tokens for them. The services requiring more computational requirements would have relative higher cost so they would require comparatively large number of tokens and the simplified ones (requiring lesser CPU time) would require relatively less number of tokens. In this way, when a game engine receives a requests for a particular sensing or action service from an agent, it looks up the services library. On the availability of the service, If the agent qualifies for the service in its functional role and has required number of available tokens, the service would be granted otherwise it will be refused.

6 The pilot experiment

The objective of this experiment is to evaluate the impact of using the LoD based AI adaptation on a player's interactivity and game experience.

6.1 The game : My Duck Hunt

My Duck Hunt is a single player shooting game developed with AGDE framework `http://gforge-lirmm.lirmm.fr/gf/project/agde/frs`. The player controls a reticle on the screen in order to acheive following goals :

– Kill ducks before they leave the scene.

– Avoid to kill flamingo.
– Prevents gombas from eating flamingos.

Clouds evolve in the sky as decoration elements. The figure 4 shows a screen shot of the game.

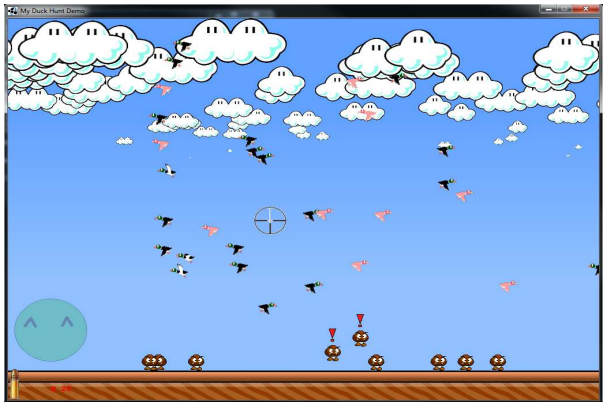

FIGURE 4: A screenshot of My Duck hunt

FIGURE 5: Pictures of the experiment

6.2 LoD configuration

In this experiment, we use three modes : advanced, best effort and basic for distributing resource tokens among agents. One of following three modes get switched depending on the available time for a frame.

1. **Advanced mode :** This mode of QoS supports full features of agents and it is triggered when the time for a frame exceeds the estimated range of time (i.e FrameTime >t1). The advanced mode of game AI operates, hence agents have access to most of the behaviors with sufficient resource tokens that are required for their execution. In this mode, entities have the access to the full set of actions that have been defined.
2. **Best effort mode :** The normalized mode gets triggered when time for a frame lies in the estimated range (i.e. t0 <FrameTime <t1). In this case, only most important behaviors of agents that get resource tokens. In this mode clouds can not move.
3. **Basic mode :** This degraded mode provides basic functionality of game agents despite frame rate falls behind the required range (i.e. FrameTime <t0). The basic design of game AI remains accessible to agents instead of total collapse or lag in the game AI. In this mode, gombas can not perform the Jump action.

In order to simulate a decrease of available resources, an entity can be added to the game. This entity send an action that requires important amount of resources in **Optimal** and **Medium** modes.

6.3 Participants

The prototype test was conducted on 8 subjects having ages between 23 and 28 years old. Most of the participants reported themselves as regulars video game players playing atleast once a week except two candidates that do not play video games.

6.4 Protocol

The experiment follows a repeated-measures design. Subjects have to play the Duck Hant game on the same computer. Two versions of the game are available, one that includes LoD based AI adaptation and the other without AI adaptation. The experiment proceeds as follows :

1. The candidate gets a quick introduction about the game. This tutorial aims to provide an introduction to game's objectives and how controlling the game.
2. The candidate plays the two versions of the game with and without LoD (the order is random). Each game version consists of eight stages (called *waves*). The subjects are not informed about the difference between the two sessions. Throughout the session, the subjects were asked to report their feelings of "lag" by pressing the space button on the keyboard and we note the time of the button press and the total time of the wave. The time ratio of "space button" to the total time of a vague clearly signifies the participants' preference of our approach over the one not using it.

3. At the end of each wave the subject has to evaluate the quality of the interaction during each waves.
4. A questionnaire is proposed and the subjects were interviewed about the quality of the interaction during both versions of the game and about his opinion concerning the game experience.

   The figure 5 shows two pictures of the experiment. The first on the left shows a subject who evaluates the interaction after a game wave. The second on the right shows a subject plying the game.

6.5 Hypothesis

The experiment aims to demonstrate that the proposed LoD based adaptation influences the game experience. Our main objective is to provide a game experience in which (i) game AI can be adapted to the requirements of the platform ; (ii) the adaptation is meant to provide an increased level of user experience which is not the case otherwise. To achieve above mentioned objectives, we state following hypothesis :

- H.A.0 : There is no difference of interactivity and game experience between the game version with LoD adaptation and the other without LoD adaptation.
- H.B.0 : There is no difference concerning the global perceived quality of interaction between the game with LoD and game without LoD.
- H.C.0 : There is no difference of the ratio of time of holding the key space and the session duration between the game with LoD and game without LoD.

6.6 Results

Statistical analysis was performed using R(http ://www.r-project.org/) version 2.15.0. We have used t-paired tests to reject the three hypotheses. The results of the t-tests on the eight waves are summarized in table 1. The t-tests have rejected the first hypothesis H.A.0 in the first six waves. The difference between the perceived quality of interaction using the game with LoD and without LoD was statically significant with a df=7. As for wave 6, 7 and 8 the hypothesis has not been rejected with the t-test.

The second hypothesis H.B.0 has been rejected by t-paired test, thus the difference between the global perceived quality of interaction of game with LoD and without LoD was statically significant with a mean of the differences M=-0.75 , t(7) = -2.3932 and a p-value = 0.04794.

Wave	M	t(7)	p-value
Wave 1	-1.75	-7	0.0002116
Wave 2	-2.25	-13.7477	2.541e-06
Wave 3	-2.25	-13.7477	2.541e-06
Wave 4	-2	-7.4833	0.0001392
Wave 5	-1.625	-5.017	0.001536
Wave 6	-0.25	-0.5092	0.6263
Wave 7	-0.125	-0.2047	0.8436
Wave 8	0.375	2.0494	0.0796

TABLE 1: Results

The ratio of session duration and time of hitting the key space during the game with LoD and without LoD was different as the hypothesis H.C.0 was rejected using t-test. The difference was statically significant with a mean M= 0.2763045, t(7) = 9.5683 and p-value = 2.86e-05.

6.7 Discussion

The results reported by our experiments clearly show the effect of LoD adaptation on the participants' game experience. The first hypothesis H.A.0 has been rejected by the t-test in six waves. This mean that the players have perceived a significant impact of LoD adaptation on the interactivity and game experience. The experimental results signify the the link between the participants' game experience and the frame rate based QoS support for game AI. The game experience and QoS link can justiy the participants bidirectional adaptation of game AI or a non-flexible AI. The argument has supported by the questionaire when the participants were asked the question : "whether they would perfer having a flexible AI to maintain interactivity or not ?", 6 among 8 participants replied affirmatively, one of them supported it on the type of the game while one participant perfered to quit the game if quality game expereince is visibly compromised for interactivity or the reverse.

However, the hypothesis has not been rejected for the last three waves as no significant difference has been reported in the experiments. This can be explained by the fact that starting from the sixth wave the average FPS between the game without LoD and with LoD is almost the same. For example the figures 6 and 7 show the average FPS on the ten later seconds for a candidate and for each playing session. The last three waves start at 3 :45 on the game without LoD and 3 :31 one the game with LoD these moments are shown on the two figures by the red lines). We can see that the average FPS is up to 60 FPS for the game without Lod (figure 6) where the average FPS is around 59 FPS on the game with LoD (figure 7).

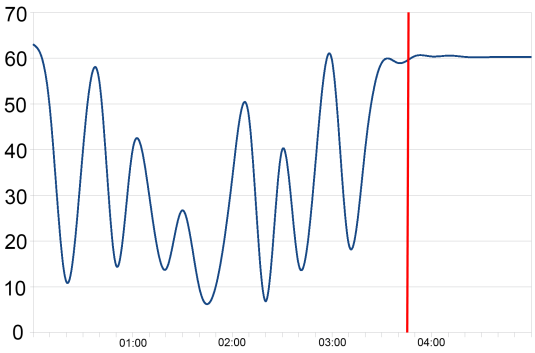

FIGURE 6: Average FPS per second without LoD adaptation of participant 2

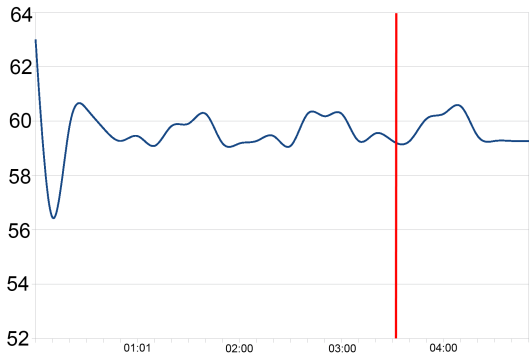

FIGURE 7: Average FPS per second with LoD adaptation of participant 2

The second hypothesis has been rejected as the participants report a significant degree of difference between the perception of the quality of interaction between the game with LoD and without it. The results support our approach for improving the overall quality of interactivity of the game experience. The statical results show that adapting the game AI as per the computational requirements of game agents significantly increased the overall interaction rating from the participants.

Finally, the t-test results have also rejected the hypothesis H.C.0. Here The experiment showed that the participants playing the without LoD version of the game reported significant difference as compared to the with LoD version.

7 Conclusion

In this paper, we have proposed a LoD based adaptation technique to prevent resource costly behaviors of game agents from degrading the quality of interaction. The main idea of our proposition is to consider the game as an agent organization where each agent has a role to play meanwhile roles are prioritized by the game designer. We introduce the logical notion of resource by which a programmer can associate the priority of roles to the distribution of resources to the agent roles. The logical notion of resources make game agents to delegate their tasks to the game engine and in this way agents' direct manipulation of resources is avoided. The delegation model eventually results in the better frame rate as agents can only consume the assigned resources.

We tested the effects of LoD in game AI with several players in our prototype. We statistically prove that there is significant difference in player's interaction and game experience when approach is used. As a future work, we plan to extend our agents to coordinate model to include more found that the technique significantly improved performance, and that players did not rate improvements in lag as more difficult or frustrating to use. Our study improves understanding of LoD in game AI and of how to link it for improving interaction in video games.

REFERENCES

BioWare (2002). Neverwinter nights - game information. [Online ; accessed 04-September-2012].

Brockington, M. (2002). Level-of-detail ai for a large role-playing game. *AI Game Programming Wisdom*, 1 :419–425.

Charles, D. (2007). *Biologically inspired artificial intelligence for computer games*. Information Science Reference.

Clark, J. (1976). Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10) :547–554.

Clark, J. (2008). Alief and belief. *Journal of Philosophy*, 105(10) :634–663.

David Luebke, Martin Reddy, J. D. C. A. V. B. W. R. H. (2003). *Level of detail for 3D graphics*. Morgan Kaufmann Pub.

Dechesne, F., Hattori, H., ter Mors, A., Such, J. M., Weyns, D., and Dignum, F., editors (2012). *Advanced Agent Technology - AAMAS 2011 Workshops, AMPLE, AOSE, ARMS, DOCM3AS, ITMAS, Taipei, Taiwan, May 2-6, 2011. Revised Selected Papers*, volume 7068 of *Lecture Notes in Computer Science*. Springer.

Deering, M. (1993). Data complexity for virtual reality : where do all the triangles go ? In *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*, pages 357–363. IEEE.

Delgado-Mata, C. and Ibáñez-Martínez, J. (2008). Ai opponents with personality traits in überpong. In *Proceedings of the 2nd international conference on*

INtelligent TEchnologies for interactive enterTAINment, page 1. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Dignum, F., editor (2011). *Agents for Games and Simulations II - Trends in Techniques, Concepts and Design [AGS 2010, The Second International Workshop on Agents for Games and Simulations, May 10, 2010, Toronto, Canada]*, volume 6525 of *Lecture Notes in Computer Science*. Springer.

Dignum, F., Bradshaw, J. M., Silverman, B. G., and van Doesburg, W. A., editors (2009a). *Agents for Games and Simulations, Trends in Techniques, Concepts and Design [AGS 2009, The First International Workshop on Agents for Games and Simulations, May 11, 2009, Budapest, Hungary]*, volume 5920 of *Lecture Notes in Computer Science*. Springer.

Dignum, F., Westra, J., Van Doesburg, W., and Harbers, M. (2009b). Games and agents : Designing intelligent gameplay. *International Journal of Computer Games Technology*, 2009 :1–18.

Ferber, J., Gutknecht, O., and Michel, F. (2003). From agents to organizations : An organizational view of multi-agent systems. In *AOSE*, pages 214–230.

Kistler, F., Wißner, M., and André, E. (2010). Level of detail based behavior control for virtual characters. In *Intelligent Virtual Agents*, pages 118–124. Springer.

Luebke, D., Watson, B., Cohen, J. D., Reddy, M., and Varshney, A. (2002). *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA.

Millington, I. and Funge, J. (2009). *Artificial intelligence for games*. Morgan Kaufmann.

Mott, K. (2009). Evolution of artificial intelligence in video games : A surve. In *Term Papers prepared for 810 :161, Artificial Intelligence, Spring 2009 University of Northern Iowa (UNIAI-09)*.

Niederberger, C. and Gross, M. (2005). Level-of-detail for cognitive real-time characters. *The Visual Computer*, 21(3) :188–202.

Niederberger, C. and Gross, M. H. (2002). Towards a game agent.

Niederberger, C. B. (2005). *Behavior Modeling and Real-Time Simulation for Autonomous Agents using Hierarchies and Level-of-Detail*. PhD thesis, Swiss Federal Institute of Technology, ETH Zurich.

Orkin, J. (2006). Three states and a plan : the ai of fear. In *Game Developers Conference*, volume 2006. Citeseer.

Osborne, D. and Dickinson, P. (2010). Improving games ai performance using grouped hierarchical level of detail.

Rabin, S. (2002). *AI Game Programming Wisdom*. Charles River Media, Inc., Rockland, MA, USA.

Remendy (2001). Games by remedy. [Online ; accessed 04-September-2012].

Russell, S. and Norvig, P. (2010). *Artificial intelligence : a modern approach*. Prentice hall.

Schreiner, T. (2003). Artificial intelligence in game design.

Studios, L. (2001). Black and white. [Online ; accessed 04-September-2012].

Valve (2008). Left 4 dead blog. [Online ; accessed 04-September-2012].

Wißner, M., Kistler, F., and André, E. (2010). Level of detail ai for virtual characters in games and simulation. *Motion in Games*, pages 206–217.

Wooldridge, M. and Jennings, N. (1995). Intelligent agents : Theory and practice. *Knowledge engineering review*, 10(2) :115–152.