Adaptive Companions in FPS Games

Jonathan Tremblay
School of Computer Science
McGill University, Montréal
Québec, Canada
jtremblay@cs.mcgill.ca

Clark Verbrugge
School of Computer Science
McGill University, Montréal
Québec, Canada
clump@cs.mcgill.ca

ABSTRACT

Non-player characters that act as companions for players are a common feature of modern games. Designing a companion that reacts appropriately to the player's experience, however, is not a trivial task, and even current, triple-A titles tend to provide companions that are either static in behaviour or evince only superficial connection to player activity. To address this issue we develop an adaptive companion that analyses the player's in-game experience and behaves accordingly. We evaluate our adaptive companion in different, non-trivial scenarios, as well as compare our proposed model to a straightforward approach to adaptivity based on Dynamic Difficulty Adjustment (DDA). The data collected demonstrates that the adaptive companion has more influence over the player's experience and that there exists an orthogonality between our companion adaptivity and the more traditional combat/health scaling approaches to difficulty adjustment. Using adaptive companions is a step forward in offering meaningful and engaging games to players.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*product metrics*; K.8.0 [**Personal Computing**]: General—*games*

General Terms

Design and Measurement

Keywords

Artificial Intelligence, Video Games

1. INTRODUCTION

In First-Person Shooters (FPS) and Third-Person Shooters (TPS) the player's relationship with team-mate or companion Non-Player Characters (NPCs) is intricately linked to the *gameplay*, challenge, narrative and experience. The companion's goal is to help the player accomplish in-game goals, simulating the effect of actual co-operative gameplay. Interesting challenges and problems arise from this situation, however, and even in AAA games we frequently find that companions are overly superficial; they may co-exist with the player, but they often fail to appropriately cooperate [2], reducing their value to players, and interfering with immersion. In a fundamental sense, these problems arise from the companion's lack of understanding of the game world and the player's dynamic, changing experience.

In this work, we investigate a novel approach to behavioural adaptivity of companions. In our design the companion is given a basic understanding of the player's experience and uses that knowledge to change its behaviour accordingly. In order to validate our adaptive companion, we developed a prototype TPS game with a basic level wherein the player is expected to interact with this companion. An Artificial Intelligence (AI) was also implemented to act as a player when running tests and gathering player-companion metrics. We show that using our approach to companion adaptivity allows the companion better control over the player's game experience than a classic game-industry approach to NPC design. We also demonstrate a parallel between behavioural adaptivity and difficulty adaptivity, and how behavioural adaptivity can be embedded in narrative wherein DDA is lacking. Specific contributions of this work include:

- We define a novel, online adaptive model for companions in war games, enabling companions to change behaviour according to basic knowledge of the player's experience.
- We develop a set of metrics to quantify companion performance, including *game intensity*, as well as novel measures such as *personal space*, and *combat load ratio*.
- Using a simple but representative game environment developed in Unity, we compare our adaptive design to a companion AI derived from the popular *Skyrim* game, and also to traditional attribute-scaling approaches to DDA. Our approach shows measurable improvement, and indicates that complex adaptivity can be used as an orthogonal adjunct to DDA.

2. BACKGROUND

Our focus in this paper is on real-time companion NPCs in war games. Below we present background information on companion design in the mainstream game industry. Of particular interest is whether and how interaction with an AI companion affects player experience. Note that detailed related work is presented later, in Section 5.

2.1 War Game Companion

Use of companion AI in war games and combat scenarios is relatively common in current games, enabling frequent, complex group-battle scenarios, without the potential tedium of detailed manual control of several characters. There are perhaps two major trends in the video game industry in this respect. The first is to provide fully autonomous companions that are expected to demonstrate appropriate behaviour without player input or control. A second common approach is to give limited, high-level control over the companion's behaviour to the player.

Fully Autonomous Companions. Games like *Skyrim* (2011) from *Betheseda Softworks* or *Left 4 Dead* (2008-2009) from *Valve* provide a companion that is expected to behave as a relatively independent, if subservient character in the gameplay. *Skyrim*, for instance, gives the player the option to acquire a companion, who then closely follows the player throughout the virtual world. The companion's tasks are to help the player in combat, carry objects the player cannot hold (weight limit), and add depth to the narrative through (limited) interaction.

In combat, a non-adaptive companion AI is frequently problematic for players [6]. The player often has a variety of combat options; she may, for example, use close-range or long-range combat weapons, or choose to directly engage enemies or *sneak* around enemies to perform a *silent* kill or avoid combat. In such cases the companion would ideally help the player in their intent—using matching (or complementary) close-range or long-range combat weapons, or by otherwise choosing a particular tactic which best comports with player actions. Unfortunately, failure of the companion to behave properly in combat is a frequent player lament. Taking *Skyrim* as an example, the companion will establish line-of-sight to the enemy even if it means walking in the front of the player when she is using range weapons, and so resulting in frequent "friendly fire" incidents, and tends to aggressively engage enemies in combat, causing a sneak-attack or combat avoidance strategy to fail [6]. Such unsatisfying and immersion-destroying experiences are common, and have also been noted in the FPS series *Left 4 Dead*.

Semi-autonomous Companions. Whereas *Skyrim* and *Left 4 Dead* greatly limit player control over companion behaviour, other games such as the war/combat game *Army of two* (2008-2010) from *Electronic Arts* or the RPG *Dragon Age II* from *Bioware* allow the player some control over companion AI behaviour.

In *Dragon Age II* this control is very high-level, consisting primarily of switching companions between aggressive, normal, and defensive combat modes. *Army of Two* gives slightly more complex control: when engaging in combat the player can give different orders to the companions, such as *hold position*, *follow me* and *push attack*. These orders provide high-level direction to companion behaviour; when the companion is asked to *hold position*, for example, it will shoot at every enemy coming towards its position and will also follow the player at a distance, whereas when the companion is asked to *follow me*, they will remain close to the player at all times and shoot at every enemy on sight. When the companion is asked to *push attack*, they will approach enemies on their own to engage in combat and will only move back to the player when she is too far away.

The different behaviours in such games can be seen as an attempt to improve player experience by letting the player inform the companion AI of her high-level intent. Drawbacks exist, however, both in the necessity of defining useful and appropriate high-level behaviours, and in requiring the player to perform meta-game actions during what is otherwise what is one of the more immersive parts of game interaction.

2.2 Player Experience

Companions are an essential part of the player experience, ensuring the player has fun and has a sense of immersion in the game world. In order to explain fun in games, scholars use the notion of *flow* described as the mental state in which a person playing a game is fully involve and immersed. Within *flow* there is an underlying structure to challenge that is of great interest for game developers. If a game is too hard for the player's skill level, she will endure anxiety. Alternatively, if the game is too easy for her skill level, she will experience boredom. For a player to enjoy a game she needs a challenge that suits her skill level. The challenge has to be epsilon bigger than her skills, in order to ensure *flow* [4].

According to Booth, a good game experience also comes from well constructed game intensity *pacing* [3]. Pacing refers to the game having fluctuations in intensity, presenting periods of low, medium, and high intensity at various points in the game. Overly lengthy periods of low or high intensity gaming are not interesting for the player as she might get bored or frustrated respectively [14].

A direct advantage to use of companions in games is in their ability to provide unique social experience for the player. For example, in *Skyrim* it is possible for the player to marry her companion. This allows for a layer of narrative depth where the player defines an intimate bound with their companion. Peter Molyneux designed *Fable 2* around this particular idea, where the player is accompanied by her dog and is expected to develop a loving relationship with it [5].

It is not common to find companions that adapt their behaviour to the player's behaviour in AAA games. In general the player will be given two options. The first is that they will have to explicitly tell the companion what to do, which removes any representation of the companion's will or freedom [13]. Alternatively, the players will have to live with a behaviour that might not respect their gameplay style and will consequently break their in-game immersion [14].

3. METHOD

This section explores the different structures needed in order to compare an adaptive companion to a modern, non-controllable companion model. For this we need a suitable game context for experimentation, an autonomous companion representative of that used in current AAA titles, a full design for our adaptive model, as well as a suite of metrics for quantitative evaluation.

3.1 Game Prototype

In order to test our approach to adaptivity, a simple TPS (prototype) was developed using *Unity 3D pro*[1]. The purpose of this prototype was to have an open platform with which to test companions and to gather information about their performance. The in-game goal of the player is to

[1] `http://unity3d.com/`

gather blue boxes as seen at the top-left of Figure 1. The challenge arises from enemies; they walk around in a predetermined path and engage in shooting combat with the player or the companion, if either are seen. If the player or companion runs away from them, they will chase them to their last known position. In Figure 1 two enemies can be seen dressed in black, the companion approaches from the right, and the player is bottom center. The companion's goal is to help the player accomplish the in-game challenge by shooting at enemies and defending the player. The non-controllable companion is further described in Section 3.2 and the adaptive companion is deconstructed in Section 3.3.

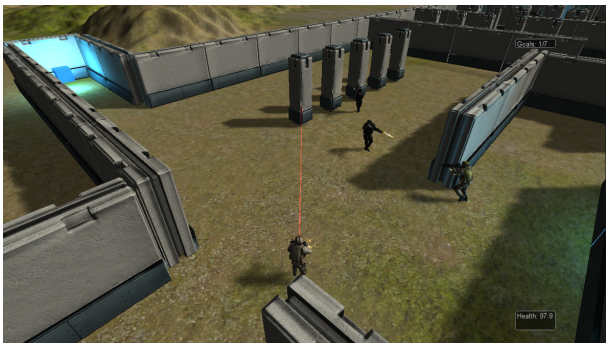

Figure 1: Simulation game used for testing

The game does not involve any resource management. The player has infinite ammunition and there is no way to recover health. In doing so, issues of adaptation under resource management such as ammo, health packs, *etc.,* were removed in order to focus on the behaviour of the companion. Further research will be to consider integration of resources into the problem space, but since player dissatisfaction with companions often concentrates on companion movement and engagement tactics, our prototype is directed primarily at these concerns.

In order to facilitate testing, an AI player was designed to automatically play the game. The AI player walks around collecting blue boxes. If it sees an enemy it will engage in combat using a pattern of shooting then dodging to the left or the right (all agents use this core strategy). This AI player thus represents a classic human player, who attempts to clean out the level as rapidly as possible by engaging in every combat and moving to the next goal without pausing.

3.2 Base Companion

In order to validate our adaptive model, a base companion was developed. This companion is inspired by *Skyrim* and so represents the game industry standards in non-controllable companion design. Figure 2 shows the *behaviour tree* [8, 11, 12] outline for the base companion's behaviour.

The base companion offers standard companion behaviour but it does not take into consideration the player's in-game experience. The behaviour tree in Figure 2 is translated as follows. The companion will fight with any enemy it sees. If the companion knows about the enemy, it will move to the last known enemy position (Move to Enemy). When the player is immobile, the companion will pick a random position near the player and move to that position every five to ten seconds. If the player is moving, the companion will follow her around.

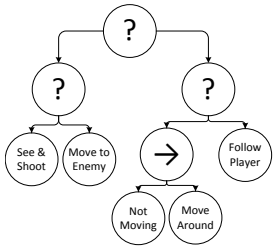

Figure 2: Base companion behaviour trees

3.3 Adaptive Companion

Adaptivity in modern computer games happens in multiple forms. The approach used in this paper was inspired by a component of the DDA system in *Left 4 Dead*. They used a game intensity metric to tailor the game experience to the player's needs [3], *e.g.* if the game intensity is too high, the DDA system will remove enemies from the map to temper the difficulty. Our adaptive companion also makes use of intensity in order to switch between different behaviours. The adaptive companion uses three distinct behaviours: *cautious*, *support* and *aggressive*.

3.3.1 Cautious

This behaviour was designed to respect the player choices when it is time to enter combat. The companion will respect if the player is sneaking around as it will not attack enemies until the player decides it is time to do so. The *cautious* behaviour tree in Figure 3 A) is translated as follows. The companion will not engage in combat with any enemies it sees. If it sees an enemy it will try to move away to a hidden location near the player, and will only start fighting if the player is already in combat. In this case the companion will move to the player's left or right if it cannot see the enemy. The companion will otherwise only move toward the player if they cannot see the player or are too far away.

3.3.2 Support

This behaviour was designed to develop a close companion to offer great support at any time and to be efficient when in combat. The *support* behaviour tree in Figure 3 B) is translated as follows. The companion will enter combat with any enemies it can see, and will also chase an enemy to its last known position. If the player is in combat and the companion cannot see the enemy, then it will try to move to player's left or right. If there is no combat the companion will closely follow the player.

3.3.3 Aggressive

This companion was designed to remove combat load from the player. The *aggressive* behaviour tree in Figure 3 C) is translated as follows. The companion will engage in combat with any enemies it sees, and will also chase an enemy to its last known position. If there are no enemies, the companion will explore the level to find enemies, only stopping if they are too far away from the player.

3.3.4 Adaptation

The different sub-behaviours of our AI are mixed within the same behaviour tree using the structure shown in Figure 4. This design makes use of the *intensity* metric we define in the next section; when the game intensity is over a

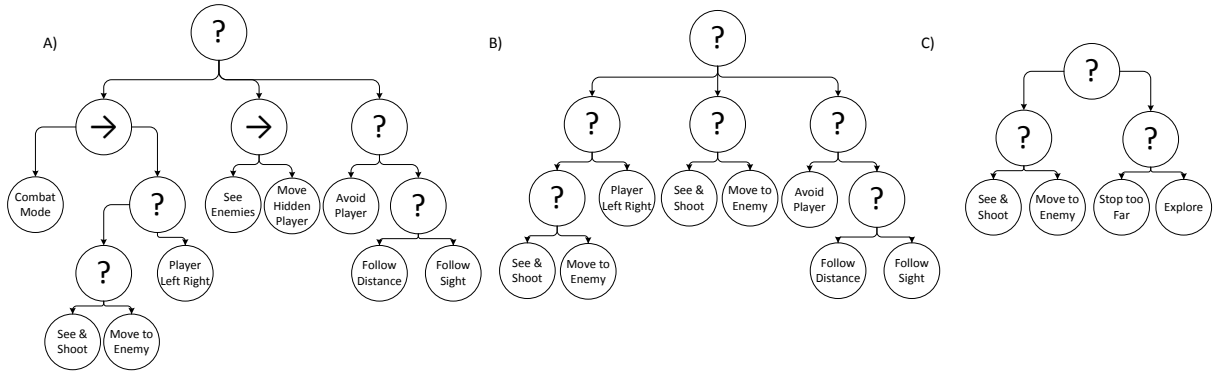

Figure 3: A) Cautious, B) Support, and C) Aggresive behaviour trees

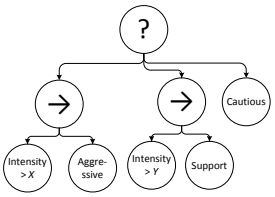

Figure 4: Adaptive behaviour trees

threshold, X, the adaptive companion will pick the *aggressive* behaviour to aggressively engage the enemy and so try and reduce the intensity. In the situation where the intensity is moderate, over threshold Y and assuming $Y < X$, the companion will support the player, trying to share the combat load more evenly with the player by switching to a *support* behaviour. When the game intensity is low, it assumes that the player is in control. In this case the companion uses the *cautious* behaviour, letting the player take on the bulk of the combat role and decisions. Note that this design embeds adaptivity in a relatively simple, static behaviour tree model. It would also be possible to vary behaviour by creating or modifying trees at runtime; *e.g.,* in the game *Driver: San Francisco* behaviour tree structure was modified dynamically by rearranging child nodes based on *hints* [12].

As expressed in Section 2.1, the companion is also there to generate narrative value. Using the presented adaptive model and communication, it is possible to note that there are intense sections where the companion could acknowledge its behaviour and game intensity. For example, after a long and intense combat, the companion could say: "It was a rough one, but we survived!" or during combat say: "I am going to push harder on them, they can't have us!" as it is switching to a more aggressive behaviour. This kind of dialogue gives life to the companion compared to predetermined dialogues.

Overall the adaptive companion was designed to have a better understanding of the right behaviour to use in different game intensity situations. The companion offers a more complete behaviour to the player than the base companion presented in Section 3.2.

3.4 Metrics

In order to develop a meaningful companion, it was important to develop quantifiable metrics that express its value. The metrics developed here are intended to relate to the player's in-game experience and performance, as well as the companion's performance.

3.4.1 Player's Experience and Performance

In order to represent the player's experience we developed a game intensity metric based on *Valve*'s DDA system used in *Left 4 Dead* [3], as well as previous work from the McGill Game Lab [1]. Game intensity is defined to be a real number, $0 \leq i \leq 1$. The intensity mainly varies according to the amount of health lost between two intensity updates, $\triangle h$. In Equation 1 the sigmoid function is used to update the game intensity, along with further parametrization chosen to limit the value of an incremental update.

$$i = \frac{1}{e^{\triangle h}} \tag{1}$$

The game intensity is also influenced by the distance squared to the companion when it gets shot, and the distance to the enemy squared when the enemy is killed. The game intensity decreases linearly over time when there is no combat. This representation helps quantify what the player is experiencing due to game combat.

A second metric, called *personal space* (PS), was designed to help determine how the player senses her companion. A companion should usually be close enough to a player for the player to be aware of the companion, more so during combat, but not so close as to interfere with movement or intended actions. We experimented with a few versions of this metric; here we use a simple design that just measures the number of times a companion enters a given (overly close) radius around the player.

In order to represent the cooperation between the player and her companion, a ratio, R was used, given the Player's (PHE) and Companion's (CHE) Health at the end of the level. The ratio is then calculated: $R = 1 - (PHE/CHE)$. As R approaches zero, the player and companion share an equivalent combat load. If the ratio is positive, it means the player had the most intense combat. If the ratio is negative it means that the companion did more work.

To measure the player's performance in-game, common metrics like number of kills (PK), average health, average game intensity (GI), number of successful hits (PHi & CHi) *etc.* were also used. Along with intensity these simple metrics show how the player interacted with the game, and can

also show how much the companion influenced the player's gameplay and experience.

3.4.2 Companion's Performance

Companion performance can be measured using many of the same metrics as applied to player performance. We were specifically interested in how much the companion potentially interfered with the player, and so also measured the time the companion spent in front of player during combat (LS), the number of times the companion accidentally fired at player, the number of times the player fired at the companion, and the companion's average distance to the player[2]. By comparing player and the companion metrics it is possible to understand how the companion influenced the player's experience.

4. EXPERIMENTAL RESULTS

In order to evaluate the behaviours of the adaptive and the non-controllable companions, different scenarios were designed and implemented in our game prototype and analysed through our metric suite. These scenarios were designed to reflect different combat situations common to FPS games, and were analyzed separately and in aggregation as a full game level.

Figure 5 shows a top-down view of the overall level used for most of the testing. The player and companion start at the green dot and have to reach the magenta dot. The trace overlay shows the output from a single simulation, where the red and purple lines represents player and companion movement respectively. The blue X's show when the player entered combat and the explosion marks indicate where an enemy died. The black circles are where the companion and player entered in collision. The white squares were the blue goals (they turn white when collected).

The tests are presented in three parts. The first part focuses on comparing the companion's influence on the player's game experience and performance. The second parts explore the usage of an adaptive companion for different player skill levels. The third part compares our adaptive model to a more classical approach to adaptivity, such as increasing/decreasing fire power.

4.1 Scenarios

In order to compare the adaptive and base companion, three core scenarios were developed within the context of a full game level. These scenarios were inspired by common components of level design found in AAA action games. Below we present the different scenarios and their results. Note that all scenarios are smaller parts of the full level (see Section 4.1.3). Every scenario was run 20 times for each type of companion, using a basic AI player to fulfil the role of the human participant.

4.1.1 Cul-de-Sac Scenario

A major level-design feature in war games involves the use of narrow, dead-end paths wherein the player finds a reward. While common, this induces a basic problem for companion AI: companions follow the player, but upon reaching the end and turning around to continue on her quest, the player finds her companion blocking the way out. A simple way to solve this problem is to increase the distance the companion must maintain to the player, but this requires limiting the depth of such dead-ends to the follow-radius of the companion, and can overly separate player from companion. The proposed adaptive companion understands that when not in combat, it should be out of the way when the player approaches. In Figure 5, the yellow section represents the maze part of the level. For this section, the player started at the entrance, and then she had to reach a goal at the end, and then return to the entrance.

[2]Note that for space reasons we cannot discuss all these metrics in this work.

Table 1: Cul-de-Sac scenario results

Type	Time (*s.*)	LS (*s.*)	PS (*s.*)
Base	24.9 ± 0.1	5.2 ± 0.1	7.7 ± 0.1
Adaptive	23.3 ± 0.1	4.1 ± 0.0	6.22 ± 0.0

Table 2: Pillars scenario results

Type	Time (*s.*)	PHE	CHE	R
Base	35.1 ± 4.0	83.1± 4.0	90.8 ± 4.1	0.3
Adaptive	30.3 ± 2.4	84.4 ± 3.5	86.7 ± 4.5	0.1

CHi	PHi	PK	GI
78.6 ± 8.1	36.8 ± 9.9	3.8 ± 0.8	0.68 ± 0.1
72.6 ± 4.1	43.2 ± 5.2	3.2 ± 1.2	0.52 ± 0.1

Table 1 gives basic metric results from this scenario. It is noticeable that gameplay with the adaptive companion took less total time than with the base companion. This is explained as the adaptive companion did not have to be pushed by the player in order to move out of the way. The adaptive companion also spent less time in the line of sight (LS) of the player, as well as less time in the player's personal space (PS). We also note that in every simulation of this level the base companion collided with the player, whereas the adaptive companion did not.

4.1.2 Pillars Scenario

The Pillars scenario was developed to test the companion behaviour inside intense, obstacle-rich combat zones. In Figure 5, the blue section represents the pillar scenario. The room is filled with six enemies and pillars behind which the player and other agents can hide or use as cover. This design is standard in FPS/TPS for combat zones or boss fights.

Table 2 gives interesting metrics results for this scenario. As with the cul-de-sac, the time taken by the adaptive companion to finish the scenario was shorter, albeit with more variance due to the randomness of combat resolution. In this case game intensity caused the adaptive companion to switch to a more aggressive behaviour, whereas the base companion just acts normally. The participation ratio (R) is closer to zero for the adaptive companion, showing a better distribution of the combat load. As the adaptive companion is more aggressive its successful hits (CHi) value was slightly higher than the adaptive companion, which brought the player number of hits down (PHi), although there was less noticeable impact on number of player kills (PK). An important overall result is that the average Game Intensity (GI) was lower for the adaptive companion.

In general, the data shows that the adaptive agent was able to reduce the intensity load on the player; this can be seen in more detail in Figure 6, which shows game intensity over time. The adaptive companion successfully dampens the more intense sections of the level, where the base agent does not evolve and does not help the player in accomplishing difficult tasks.

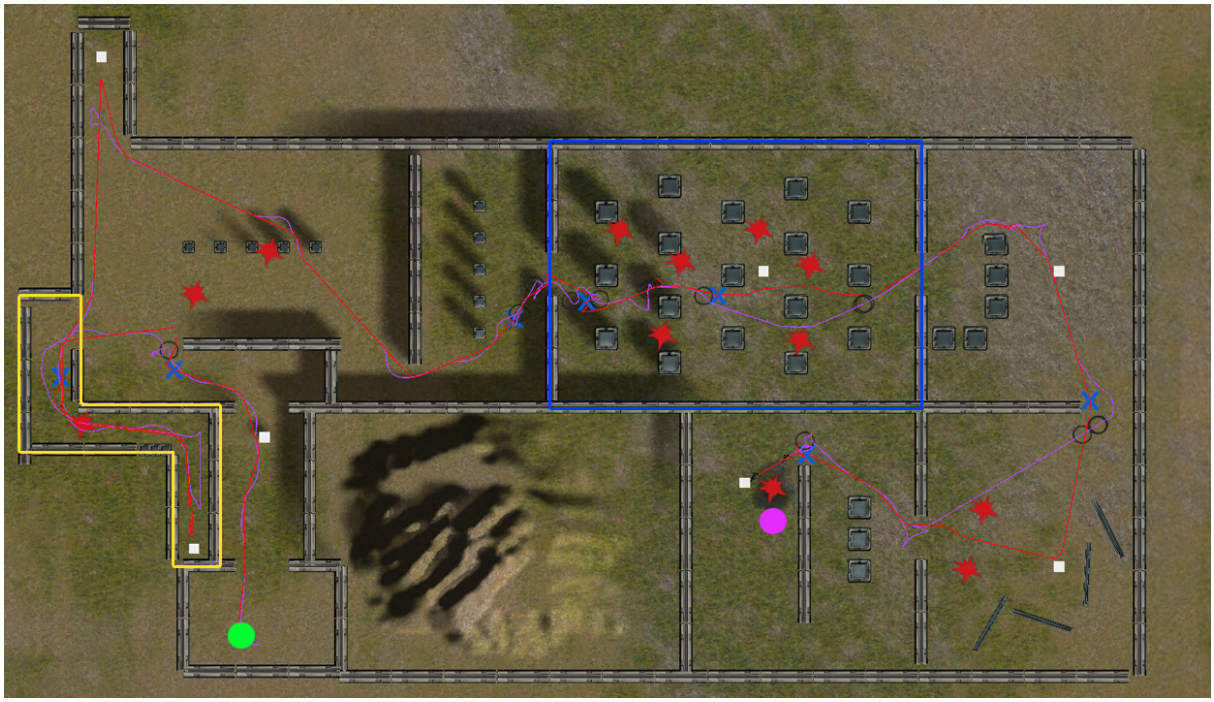

Figure 5: Tested level

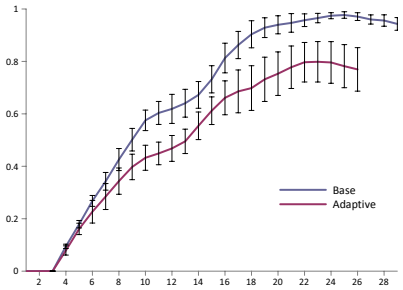

Figure 6: Game Intensity over time (s.) for the pillars scenario

Table 3: Level scenario results

Type	Time (*s.*)	PHE	CHE	R
Base	142.5 ± 8.8	63.5± 5.0	92.5 ± 6.9	0.31
Adaptive	131.9 ± 5.0	71.2 ± 3.9	83.3 ± 10.1	0.14

PHi	CHi	PK	GI
238.7 ± 24.3	51.6 ± 24.9	8.55 ± 1.6	0.43 ± 0.1
182.7 ± 13.2	108.9 ± 14.1	6.8 ± 1.4	0.36 ± 0.1

4.1.3 Level Scenario

As a more complete test, we also composed the scenarios into the full level shown in Figure 5. The AI player starts from the green spot and has to reach the last goal where there is a boss fight (magenta spot). This level was designed to test the companion in a realistic game environment, incorporating a maze/cul-de-sac section, combat situations with varying number of enemies and obstacle density, and a final boss fight.

Table 3 gives metric result for this level scenario. Note that variance is increased here in all factors, primarily due to the non-determinism built into combat aiming and combat positioning. Nevertheless, again gameplay with the adaptive companion can be seen to be more efficient, reducing average level completion time. Player hits (PHi) are higher with the base companion than the adaptive one, showing that the adaptive companion helps the player more than the base companion. This is further shown in companion hits (CHi) and player kills (PK), and also reflected in the health ratio (R), which is closer to zero for the adaptive companion – the adaptive companion worked harder and took more damage, which resonates well with the last argument.

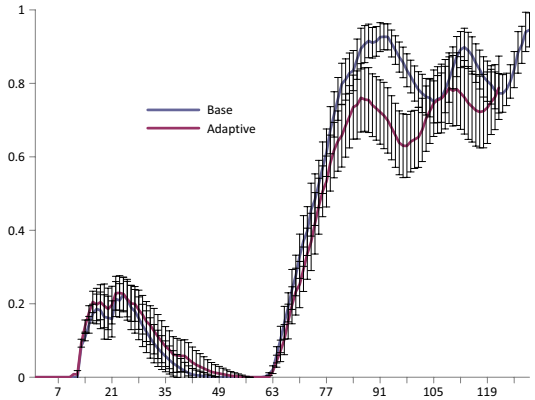

Figure 7: Game Intensity over time (s.) for the level scenario

Figure 7 shows the game intensity over time for this scenario. At around 80 seconds, the adaptive companion started using the aggressive behaviour, which caused the player to

experience less intensity in a shorter period of time. Overall the adaptive companion shows a better understanding of the game situation in term of intensity. This understanding has subtle differences in term of player and companion performances, but this difference has an impact on the player's in-game experience.

4.2 Naive & Expert AI Player

The value of companion adaptivity can also be seen as a means to help games adapt to different player skill levels. A naive player will have weaker aim and be slower in completing a scenario than an expert player, and so player experience may be improved by a companion that can recognize the need for intensity reduction and respond accordingly. In this test we thus compare the impact of an adaptive companion on a naive player with weak aim with its impact on an expert player, using the same full level as above.

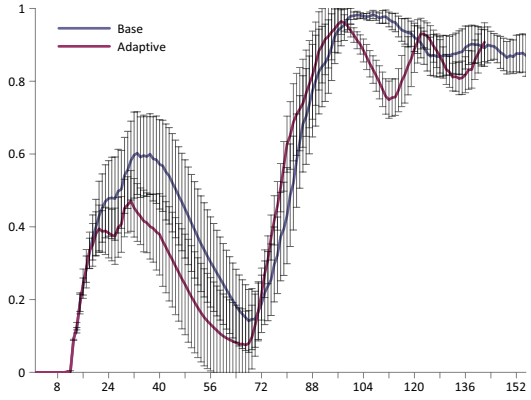

Figure 8: Game Intensity over time (s.) for the Naive AI player

For this particular test, the game intensity variability is the most interesting metric to look at. The earlier Figure 7 shows the game intensity produced assuming an expert player, while Figure 8 shows the game intensity data from a naive skill level run. In both cases, and as expected, the game intensity level for the AI player is lower when playing with the adaptive companion than with the base companion. The naive player scenario does show dramatically larger variance, but the impact of an adaptive companion results in a better moderation of intensity. Note that this impact is partly hidden by the way the intensity measure caps at 1.0; as the game proceeds, a naive player in conjunction with the base companion reaches the highest intensity levels, while gameplay with the adaptive companion is able to offer some reduction. A further, interesting observation is apparent in the shape of the curves from about 90s onward in Figure 8. At this point, the base companion is essentially unable to help the naive player, resulting in a continuously intensive latter third of gameplay, with no calmer periods. This highly stressful gameplay is in contrast to the results shown with our adaptive companion, which at least partially restores the intended, natural up-and-down pace to the level, producing a series of peaks and values much more similar to that experienced by an expert player.

In general, the adaptive model is more successful at helping a player adapt to subtle differences in difficulty. These sorts of differences happen when a player is not paying attention to the game or is learning a new game mechanic [14]. Of course stronger and more universally aggressive companions would also reduce intensity, although that brings concerns of over-trivializing gameplay, which is also not the goal of commercial computer games. The companion adaptivity is there to tweak the difficulty of the game to give a better experience to the player.

4.3 Comparison to DDA

Adaptivity within games is most frequently and easily expressed through some form of dynamic difficulty adjustment based on uniformly increasing or decreasing enemy or player power. We thus compare our design with a basic DDA. The DDA system uses the same structure as the proposed model in Section 3.3.4, but modifying only companion fire power according to the game intensity—the higher the intensity the stronger its fire power. Note that as this paper is interested in adaptive companions, fire power adaptivity was given to the companion only. This particular companion will be denoted DDA companion, and an intensity graph for a naive player with a DDA companion, an adaptive companion, and a hybrid DDA-adaptive companion (including both behaviours) is shown in Figure 9.

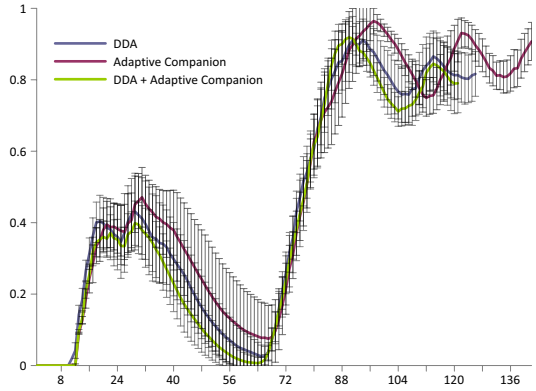

Figure 9: Game Intensity over time (s.) for the naive AI player with adaptive companion and DDA companion

Much like the adaptive, as the DDA companion is given more fire power it takes less time on average to finish the level, slightly compressing the intensity curve over the time axis (and so peaks/valleys line up less well). It is clear that improving fire power has a large impact, and is also able to recover a sinusoidal fluctuations in intensity for the naive player. The hybrid form, however, shows a consistent, further mitigation of intensity, and this leads to the thought that there is an orthogonality between the adaptive companion controlling the game intensity and a more classic form of adaptivity. An appropriate combination of the two techniques may thus enable the reduction in intensity of naive play to reach the expert levels, and result in a more uniform game experience. This additional impact must of course be measured in relation to the known drawbacks of DDA, in that letting players witness dynamic adjustments that grossly exceed their own abilities or contradict previous experience break immersion. In this sense our adaptive companion provides a much more contextually integrated form of adjustment.

5. RELATED WORK

There are multiple formalisms to use when it comes to building an intelligent agent. We make use of a *Behaviour Tree* (BT) formalism, as one that has gained ground in the game industry [8]. A BT is a tree structure formalism that uses depth-first search combined with an execution model for individual tree nodes. Each node is considered a *task* and when evaluated returns *true* or *false*. The tree evaluation stops when the root node returns true or false. The behaviour of internal nodes is based on the output function of their children, where external nodes, leafs or actions/conditions provide or test basic game operations [3]. As a caveat, the formalism is not well defined in the industry and hacks are done in order to accomplish certain function, such as hiding parts of the tree [8].

Game adaptivity can be applied to multiple facets of the game: difficulty, agent tactics, player companionship, *etc.* One simple technique that is often used in game design is negative feedback [13], which may be applied in a video game to, for example, reduce the gap between two players' scores or other attributes. This is achievable by giving a proportional bonus to the player with the lowest score.

In general, the standard approach to DDA is to monitor the player, and based on the observations make decisions on how to improve the game experience, apply the changes, and repeat [9]. Hunicke [7] was interested in building an adaptive system for a level in the FPS *Half-life*. She argues that the dynamic economic system from the player to the game can be controlled in order to produce smooth adaptivity. Ocio [12] used adaptive behaviour trees to change the order of action nodes based on in-game knowledge. This tree was then used to build levels tailored to the player. Their goal was to accommodate casual players by decreasing the difficulty.

The focus of companion adaptivity research has been towards understanding the player's actions or improving performance. Macindoe *et al.* [10] use a POMDP model game structure to develop a companion that understands how its actions influence human intentions in puzzle games. This kind of reasoning enables the companion to interact indirectly with the player and thus solve game puzzles. Tan and Cheng [15] proposed a neural network approach to improve survival chances of a group of NPCs and a player in an apocalyptic zombie world. By adapting action selections based on previous outcomes, their system demonstrated an approach that was more efficient in terms of survival than pre-scripted behaviour.

6. CONCLUSIONS AND FUTURE WORK

Modern computer games rely on companion agents to help the player accomplish in-game challenges and create deeper narratives. Nevertheless, this introduction brings new problems such as the companion breaking the player's experience by not respecting the player's intentions and gaming style.

We presented an adaptive model for real-time companions that takes the player's game experience into consideration when making decisions. We showed that this particular model was able to temper the player's game intensity level when compared to a non-adaptive companion. We demonstrated that the effect is similar in scale to the use of a more traditional approach to adaptivity based on dynamic difficulty adjustment, but has additional advantages in being both orthogonal to DDA, and of allowing for better narrative justification of any adaptivity.

A simple extension of this work would be to apply it to behavioural adaptivity of enemies. Further work is also possible in developing an adaptive companion that takes into consideration the full complexity of resource management. Finally, player action recognition should be explored, as this would allow the companion to understand and potentially assist with the higher-level actions the player is attempting.

7. ACKNOWLEDGEMENTS

This research was supported by the Fonds de recherche du Québec - Nature et technologies, and the Natural Sciences and Engineering Research Council of Canada.

8. REFERENCES

[1] M. Ashton and C. Verbrugge. Measuring Cooperative Gameplay Pacing in World of Warcraft. FDG, pages 77–83, 2011.

[2] S. Bakkes, P. Spronck, and E. O. Postma. Best-response Learning of Team Behaviour in Quake III. In *Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 13–18, 2005.

[3] M. Booth. The AI systems of Left 4 Dead. Keynote presentation at AIIDE, 2009.

[4] J. Chen. Flow in Games (and Everything Else). *Commun. ACM*, pages 31–34, 2007.

[5] B. Crecente. Fable 2's Big Thing: A Pet Dog, 2007. `http://kotaku.com/241952/fable-2s-big-thing-a-pet-dog-update`.

[6] M. Hughes. The Elder Scrolls V: Skyrim followers guide, 2011. `http://www.gamesradar.com/elder-scrolls-v-skyrim-followers-guide/`.

[7] R. Hunicke. The Case for Dynamic Difficulty Adjustment in Games. ACE, pages 429–433, 2005.

[8] D. Isla. GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI. `http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php`, 2005.

[9] R. Lopes and R. Bidarra. Adaptivity Challenges in Games and Simulations: A Survey. *CIG*, pages 83–108, 2011.

[10] O. Macindoe, L. P. Kaelbling, and T. Lozano-Pérez. Pomcop: Belief space planning for sidekicks in cooperative games. In *AIIDE*, 2012.

[11] I. Millington and J. Funge. *Artificial Intelligence for Games*. Morgan Kaufmann, second edition, 2009.

[12] S. Ocio. Adapting AI Behaviors To Players in Driver San Francisco: Hinted-Execution Behavior Trees. In *AIIDE*, 2012.

[13] K. Salen and E. Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, 2003.

[14] J. Schell. *The Art of Game Design a Book of Lenses*. Elsevier/Morgan Kaufmann, 2008.

[15] C. T. Tan and H.-L. Cheng. Personality-based Adaptation for Teamwork in Game Agents. In *AIIDE*, pages 37–42, 2007.