Improving Behaviour and Decision Making for Companions in Modern Digital Games

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

Abstract

Non-player character companions in video games should cooperate with players, understand them, and follow their lead during gameplay. In current games, however, companions tend to exhibit mainly static behaviours, and rarely live up to player expectations. In general, our work is aimed at improving this situation, developing both techniques and tools which allow companion NPCs to behave more appropriately, respecting player preferences and offering a more immersive gameplay for players.

Problem Introduction

Non-Player Characters (NPC) bring interesting challenges and problems when evolving side by side with human players. For example, in First Person Shooter (FPS) games, a given NPC playing the role of a companion has to behave in a certain way to offer support to the player and help the player during challenging moments. Unfortunately, companion behaviour is often unchanging, static, and frequently very limited as well, resulting in companions that, as neatly summarized by Bakkes, are overly superficial: they may *coexist* with the player, but they often fail to appropriately *cooperate* (Bakkes, Spronck, and Postma 2005), reducing their value to players, and interfering with immersion.

In this proposal, we are interested in using different Artificial Intelligence (AI) techniques to solve problems caused by NPC companions and to improve the state of game companions. In other words, companions should behave as human players are expecting them to, mimicking human companions behaviour or choosing best response behaviour. To achieve this broad goal, we need both improved AI techniques for companions, appropriate to their supporting role in various situations, as well as sufficient knowledge of the player's state and goals to select the right companion's behaviour. This results in a complex design challenge, involving the creation of multiple and varied companion AI components, the need to detect or predict player intention, and for the companion to adapt appropriately. We organize our approach to this problem around the following three goals or themes.

- *Creating and adapting companion behaviour*. This includes defining combat and non-combat behaviours that correlate and change with respect to the player's in-game intention and behaviours. In this we want to create a companion who is there to improve the player's immersion and help accomplish harder challenges, while leaving most of the work load to the player.
- *Player metrics & plan recognition*. Knowledge of the player's intent is crucial to the decision making process of the companion, since the companion should cooperate with the player, and wrong beliefs about the player will negatively influence the player's experience and immersion. We thus aim to develop player monitoring and prediction techniques that can help the companion act most appropriately.
- *AI game design tools*. The problems we address are complex, and exploration will benefit from tools that can facilitate the research and design processes by showing and quantitatively evaluating different algorithms and game situations. Our approach thus includes development of non-trivial tool support, with the expectation that this will aid research, and also be of value to game designers, as they also need to perform interactive and iterative refinement of game levels and companion behaviours.

Proposed Research

In this section we expand on the three main themes of the research—developing appropriate companion behaviours, analyzing players, and tool construction. Note that our approach is intended to apply primarily to FPS (and perhaps RPG) games, although some of these techniques may extend to other genres.

Companion Behaviour

There are many possible ways companion behaviour can be improved. We divide our efforts, addressing combat and non-combat situations, as they are key components of a player's in-game experience. In each case we focus on solving a specific problem, as an incremental approach to building a larger overall solution.

Combat Companion Behaviour - When a player enters a combat accompanied by a companion, she is expecting that the companion will follow her lead and make reasonable,

even *smart* choices. This sort of behaviour is not achievable when the companion uses a static AI, and so we need a general approach that can adapt to the player and current game situation. This general adaptivity should occur together with appropriate combat choices, such as which target to select first.

Adaptive Companion Behaviour - Adaptivity is an important concept that has the power to improve player's game experience, reach a broader range of players, and allow for longer periods of *flow*, *etc.* (Lopes and Bidarra 2011). An adaptive system first observes the player's in-game actions and infers knowledge about her experience, and then the system uses these beliefs to modify the game's system, such as in terms of enemy health levels, availability of combat resources (ammo), or parameters of different challenges, such as the length of a gap in a platformer game, *etc*. To maximize immersion, this adaptivity should be well contextualized, and so relatively invisible to the player.

In previous work, we introduced a system that infers the player's in-game intensity level and uses this value to choose the right companion behaviour (Tremblay and Verbrugge 2013). We wanted to show how different behaviours can influence the player's experience and workload. For this we separated behaviours into three classes:

- *Cautious,* where the companion follows the player but will not engage combat with enemies until the player decides to.
- *Support,* where the companion follows the player and engages combat with seen enemies.
- *Aggressive,* where the companion explores the space to find combat in advance of and irrespective of the player.

Measuring the game *intensity* then gives motivation for the companion to switch between these different modes. This allows the game to adjust the workload on the player, offloading it to different degrees onto the companion. This design not only makes companion behaviours more appropriate, matching the player's perception of combat difficulty, but also has the advantage of acting as a natural form of difficult adjustment, orthogonal to and more immersive than classical approaches to dynamic difficulty adjustment, such as Hunicle's (Hunicke 2005).

Target Selection - In most combat games, game entities such as enemies are defined using numerical values: *e.g.* health, attack, defence, *etc*. These values directly influence how combat should be approached for optimal results, and are properties players quickly learn to use when optimizing their own approach to combat. Companions, however, often make highly simplistic and sub-optimal decisions in this respect (*e.g.* attack closest), partly due to the fact that optimally solving such basic attrition games is expensive: the enemy target decision problem known to be PSPACE-hard (Furtak and Buro 2010). Companions, however, do not need to always reach optimal results in order to show better, more appropriate and human-like behaviours. Thus, rather than solving the general form of the problem, we have developed a fast heuristic that can be easily computed in polynomial time and which provides excellent results in a FPS setting.

Our heuristic uses only the attack, a, and health, h, values of an enemy, e, of all the enemies, E. From this it is possible to find the enemy with the highest "threat" value,

$$\max_{e \in E} [e.a \cdot (\lceil E.h - e.h \rceil)] \quad (1)$$

and use that to improve target selection. In a non-trivial game simulations, we can then show that a companion that uses classic strategies such as picking the enemy that is closest, strongest, or the lowest health, *etc.* will perform significantly less well than one using the above heuristic. Improving companion combat behaviours in this way greatly reduces the frequency of companions demonstrating immersion-breaking, clearly unrealistic target choices, without requiring players take control over companions to perform manual target selection.

Non-Combat Behaviour - A large part of FPS and RPG gameplay involves non-combat activities, such as gathering resources, mapping, or sneaking by enemies. These passive behaviours are important to the player, adding variety and depth to the gameplay, but are also situations where companions tend to be either useless or can even interfere with player intent. We focus initially on sneaking as a popular non-combat oriented aspect of many combat games.

Sneaking Companion - In many games, players may engage in a *sneaking* strategy, either to conserve resources, temporarily avoid particularly difficult enemies, or even as a necessary part of the overall game flow. *Metal Gear Solid* from *Konami* or *Commando* from *Pyro Studio*, for instance, allow both sneaking and attacking behaviour, and so require a companion that can both fight and sneak. This, however, requires the companion to both recognize the player intent, and for the companion to demonstrate appropriate and effective sneaking themselves, again with an eye toward using this ability to improve adaptivity and player immersion. This part of the research will therefore focus on developing game analysis and design techniques that allow one (or more) companion NPCs to properly plan a sneaking path.

Work here builds on a prototype tool already under construction, and is (initially) directed at simple game scenarios, involving a 2D space, deterministic movement of other (enemy) NPCs, and a well specified start and ending goal. Within this context, we will develop basic heuristics for efficiently determining a path that avoids enemy lines of sight. This is actually a geometric visibility problem, combined with dynamic path-finding, and our current approach uses a technique borrowed from the robotics community that projects the 2D space through discrete time steps, building a 3D space in which path-finding can be performed.

Further solutions for more complex scenarios can then be derived as extensions of this basic approach. The problem of having a human player and an AI companion following her removes our assumption about deterministic movement, and thus forbids pre-computation of sneak paths. Here we expect to take advantage of the fact that in many game contexts there are only a few reasonable ways for players to proceed: if we run our sneak path detection from the players position, finding and clustering sets of appropriate paths, we can use observations of the player's first few movements as a

predictor of both whether a player is actively sneaking, and which sneak path she may be following.

Another interesting and useful extension of this work involves incorporating sound as a concern, and in general extending the problem to consider multiple and probabilistic sources of detection. Variable movement rates are also interesting to explore, and further magnify the problem complexity. In all this work we will aim at both determining optimal solutions (to the extent possible), and at deriving efficient heuristics that can realistically be used in real-time game contexts. Understanding the relation between optimal and realistic behaviours gives us yet another mechanism for contextualizing game adaptivity.

Player Actions Modelling

In order to make rational decisions in a game, the companion needs two sorts of knowledge. The player's internal action state such as her skill level, game intensity level, action likelihood, *etc.* is used by the companion to make decisions such as how aggressive to be in combat or how to share resources with the player, movement behaviour, *etc.* (Bakkes, Spronck, and van Lankveld 2012). The companion, however, also needs knowledge about the the goal of the player, such as whether she is sneaking, gathering resources, or engaged in a particular combat strategy. The only way a system within a classic video game setup (keyboard and mouse or gamepad) can gain that sort of knowledge is by observing different player actions and then inferring beliefs about the player state. This is challenging as the player state space is non-trivial to define and explore. We propose two approaches to ascertaining the player state: metrics based observation, as they are non-expensive to compute, and Bayesian network recognition which allows for more complex plan recognition.

Metrics - Metric design is not an easy task; one has to find correlation between the player's state and observable game actions, and while some correlations are obvious and trivial (low health implies difficulty), some states may need inferance over multiple features and observations (such as detecting boredom, or frustration). For the latter we hope to apply techniques from machine learning, such as linear regression, and basic data-mining/aggregation techniques such as principal component analysis.

We have some initial, although quite simple results in this area, inspired by the well known game intensity metric developed for *Left 4 Dead 1* and *2* from *Valve*. The intent here is to measure the player's game intensity, as a second order metric based on measures such as how far the player is from a kill, the player's health variability, and other traumatic in-game events. This metric was then used by an adaptive system in order to offer a tailored game experience, and we used a variant of this approach in our earlier research on adapting companion combat behaviours.

Bayesian Plan Recognition - For higher level understanding of player behaviour, we plan on using Bayesian plan recognition (Charniak and Goldman 1993). Here we consider every plan as a different behaviour for a NPC denoted h. Each behaviour is given a starting probability $P(h_i)$ and it is linked to at least one observable game action denoted e_m. Observations influence the probability of a given behaviour to be more interesting for the player, and it is defined as $P(e_m|h_i)$. The final probability of a behaviour is then given by:

$$P(h_i|e_1 \dots e_m) = \frac{P(e_1|h_i) \cdot P(e_m|h_i) \cdot P(h_i)}{\sum_{j=[1,n]} P(e_1|h_j) \cdot \ldots \cdot P(e_m|h_j) \cdot p(h_j)}$$

Where m is the number of observations observed and n the number of behaviours. A behaviour with probability 80% will not always be picked, giving rise to unpredictable behaviour. Within this structure it is possible to express adaptivity in different situations, such as the player choosing a certain weapon or having low health. This structure is not only usable to model adaptive difficulty, but also to modify NPC behaviour within respect of player's own interests and actions (Macindoe, Kaelbling, and Lozano-Pérez 2012).

Tool Design

Much of the space that we need to explore when designing companions is not trivial, and would be easier in the context of interactive design and measurement tools. We therefore intend to develop design tools and corresponding game simulations within a realistic, industry standard game context, such as *Unity 3D*. Designers and researchers (us) can then better understand and measure the impact of different choices, observing and tuning companion behaviours in an environment that can be directly applied to non-trivial game development. In this we want to explore combat and non-combat level design.

Combat Tool - When designing levels for combat, designers have to think about a variety of geometric properties, such as presence of choke points, potential movement paths, as well as semantic properties, such the potential intensity progression induced by enemy placements. Given a level, with an enemy distribution, we want to explore the different combat scenarios that could happen. This work will build on two concepts. First, we will use influence maps, as a coarse way of understanding the strategic value of different initial and dynamically evolving unit positions (Hagelbäck and Johansson 2008). This will allow us to give companions basic strategy, and to potentially recognize the strategy being employed by the player (at least under suitably constrained circumstances). A second, more precise approach will be to focus on geometric analysis, (pre-)computing coverage and ranking potential movement paths (Shi and Crawfis 2013). Such tools allow designers to iterate level design faster, by being able to see potential game-time behaviours. We are of course most interested in including the behaviour of a companion within the level exploration, and observing how the companion behaviours modify level metrics and game difficulty.

Non Combative Tool - For non-combat contexts, we will initially aim tool design at the sneaking problem we discussed previously. Here we are interested in getting knowledge about the different paths a player and/or companion could take from A to B while avoiding visibility to other NPC (enemy) game agents. We have already begun work

in this area, using Rapidly Randomized Tree search (RRT) in order to efficiently explore the different sneak paths a level presents, an approach inspired by Bauer's work (Bauer, Cooper, and Popović 2013). These paths are then clustered together and metrics are offered to the designer about the different paths, such as time stamps, visibility, duration *etc.* This information is then presented to a researcher or designer as part of the interactive game-editing context of *Unity 3D*, letting one explore final or incremental results in a potential gameplay. Again, we hope to use this tool in both developing appropriate companion behaviours, and aiding in predicting possible player choices.

Progress & Validation

In order to bring this interconnected project to reality, the different sections will unfold at different time. This also means the work will be realised in parallel as some parts are needed for other to exist, *e.g.* plan recognition in order to switch to the right behaviour.

The sections on building adaptive AI using different formalisms is already explored and was published as a full paper at FDG 2013 where it won best paper award in technology and interaction (Tremblay and Verbrugge 2013). The part on the target selection problem is partly explored.

Now we are working on the companion sneaking behaviour by building the tool first. At this point we have an interactive prototype already, implemented in *Unity 3D*, and are currently exploring different efficiency improvements and tuning the interface. This work was accepted at the Second Workshop on Artificial Intelligence in the Game Design Process (IDPv2 at AIIDE 2013).

As a further extension, and to allow for more in-game, dynamic use of the results, we are also investigating the idea of running our sneak-path analysis on a cloud server, and so offload the computational costs. This is meant to augment, rather than replace the results of simpler heuristics and ahead-of-time computation, and so allow companion quality to scale with available resources.

After the sneaking project we will work on putting combat and non-combat behaviours together for the companion. This will be achieved using machine learning algorithm to learn policies to understand when the player is sneaking or not. Some preliminary work has already been done in order to build behaviour trees linked to a Bayesian network, again integrated in *Unity 3D*. Depending on previous experience, we may also move this computation to a cloud sever in order to better infer the player's state and to improve the data set the algorithm is learning from.

Finally, in order to validate our work we are relying on the usage of NPC as a human player. This process allows a deeper analysis as NPC's behaviours are stable, precise and duplicable. These analyses will use metrics and comparative techniques that are accepted by the literature, such as game intensity, team dynamics (*robocup*), behavioural analysis, *etc*. In order to ensure the validity of this approach we are planning on defining key human study experiments.

Conclusion

NPC companions have enormous potential in games. They can facilitate group gameplay in the absence of multiple real players, inspire increased emotional investment on part of players, and have mechanical benefits in terms of being highly adjustable, offering an easily contextualized means of performing difficulty adjustment or otherwise helping players. Current implementations, however, provide only simple, static behaviours that either greatly limit their utility, reducing them to mere pack-beasts or cannon-fodder, or which result in immersion-breaking actions that frustrate player intent.

Our proposal aims at producing a collaborative and helpful companion. This will be achieved by exploring algorithms that will make smarter choices in important situations, building on non-trivial interpretations of the player's state to use appropriate behaviours that match player intent. The tools we develop in the course of this work are also expected to be helpful to game design in practice, providing means for designers to explore companion activity and better integrate companion AI into real games.

References

Bakkes, S.; Spronck, P.; and Postma, E. O. 2005. Best-response learning of team behaviour in quake iii. In *Workshop on Reasoning, Representation, and Learning in Computer Games*, 13–18.

Bakkes, S. C.; Spronck, P. H.; and van Lankveld, G. 2012. Player behavioural modelling for video games. *Entertainment Computing* 71–79.

Bauer, A.; Cooper, S.; and Popović, Z. 2013. Automated redesign of local playspace properties. In *FDG 2013*, 190–197.

Charniak, E., and Goldman, R. P. 1993. A bayesian model of plan recognition. *Artificial Intelligence* 53–79.

Furtak, T., and Buro, M. 2010. On the complexity of two-player attrition games played on graphs. In *AIIDE 2010*.

Hagelbäck, J., and Johansson, S. J. 2008. Using multi-agent potential fields in real-time strategy games. In *AAMAS 2008*, 631–638.

Hunicke, R. 2005. The case for dynamic difficulty adjustment in games. In *ACE 2005*, 429–433.

Lopes, R., and Bidarra, R. 2011. Adaptivity challenges in games and simulations: A survey. In *CIG 2011*, 83–108.

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

Shi, Y., and Crawfis, R. 2013. Optimal cover placement against static enemy positions. In *FDG 2013*, 109–116.

Tremblay, J., and Verbrugge, C. 2013. Adaptive companion in fps game. In *FDG 2013*, 229–236.