Game AI Planning Analytics: The Case of Three First-Person Shooters

Éric Jacopin*
MACCLIA
CREC Saint-Cyr
Écoles de Coëtquidan
F-56381 GUER Cedex
eric.jacopin@st-cyr.terre-net.defense.gouv.fr

Abstract

We present a general framework for Game Artificial Intelligence Planning (AIP) Analytics. The objective is to provide analytic tools to study and improve AIP components and their use in video-games. Extraction and formatting of AI data is first described and discussed. Then AIP metrics are listed with examples and illustrations from three popular First-Person Shooters: *F.E.A.R.* (2005), *Zone 3* (2011) and *Transformers 3: Fall of Cybertron* (2012). The patterns we discovered in our study clearly show the AIP component is called more often by the game over the years.

Introduction

Game Analytics (Seif El-Nasr, Drachen, and Canossa 2013) is the process of discovering patterns in data obtained from playing a video-game. To this end, these data are transformed into some kind of interpretable measures, the so-called metrics, which, in the case of video-games, can be categorized into the three following types (Seif El-Nasr, Drachen, and Canossa 2013, (2.1.4)): (i) user metrics (related to game players), (ii) performance metrics (related to game software) and (iii) process metrics (related to game development). The goal is to turn the patterns into decisions across all levels of a (game) company; this is the role of business intelligence.

This paper reports on various performance metrics designed to improve the Artificial Intelligence (AI) Planning features used in video-games to generate the behaviours of the Non-Player Characters (NPCs) populating these games. Since the very first introduction of Goal-Oriented Action Planning (GOAP) (Orkin 2006) in the game *F.E.A.R.* (Black 2005), implementing a real-time planner in a video-game can now be considered as a solved problem: GOAP's runtime in the game is perfect[1] and has been widely reused, partly because of the opening of the C++ code in the SDK later released (Monolith Productions 2006). Runtime measures can be useful during game development in order to check whether the time-budget allocated to the planner is met. But performance metrics should focus on the planning features in the game, so as to support AIP intelligence, such as data analytics supports business intelligence.

The planner in a game acts as the decision-making component, generating plans of actions to be executed by the NPCs. AIP intelligence has thus do with the usual "Who?", "What?", and "Where?" questions: Who is this NPC using this action in this location? That is, does the AIP we implemented happen how we want it to? And when planning indeed does happen the way we want it to, is it useful? For instance, if we discover some actions we designed and implemented eventually are very little used by the NPCs, is this right? That is, should have we spend our time and development effort for some other feature in the game? Finally, how do our planner compare to the AI Planners of other games? Indeed, not only GOAP is used in games but also HTN Planning (Ghallab, Nau, and Traverso 2004); how does the former compare to the latter?

In order to formulate answers to the previous questions, AIP data has been extracted from one game using GOAP: *F.E.A.R.* (FEAR), and two games using HTN Planning: *Kill-Zone 3* (KZ3) and *Transformers 3: Fall of Cybertron* (T3). A set of Mathematica (Wolfram 1999) functions has been developed to process these data. As, to our knowledge, no other work had ever been done on this topic, several iterations on the Mathematica functions and many discussions with game AI programmers were needed to produce the results presented in this paper.

This paper is organized as follows. The next section describes the kind of data which were extracted from the games, and the metrics we designed to study and compare the AIP component in FEAR, KZ3 and T3. Section 3 presents AIP metrics we used to discover the patterns described in section 4. Finally, we report on Game AIP Intelligence in section 5.

No algorithmic knowledge of planning is expected from the reader who, however, should be familiar with box plots.

*Endless thanks to Jeff Orkin for discussing and answering so many questions and to Arjen Beij and Troy Humphreys who saved my life in producing the AIP data as well as answering my questions about KZ3 and T3, respectively. Special thanks to Alex Champandard for his comments on a first report of this work.

[1]The HTN planner in *Transformers 3: Fall of Cybertron* is also reported to be the fastest AI component in the game (Humphreys 2013).

AI Planning data

The AIP data production chain is the following: the game code is instrumented to produce AIP in-game data (time-stamp, NPC name, actions and their parameters, plan length, ...) which is logged to a file while playing the game; specific tokens may be added to that file so as to facilitate the next parsing and rewriting step to suit the input format of your preferred analytic tools.

This section describes the format of the AIP data which are extracted from the data logged in files while playing the games.

We began this study with FEAR and modified the freely available SDK source code so as to record, in a log file, each plan generated by the planner for a given NPC, together with a time stamp so as to know when this plan was generated.

Playing FEAR, we adapted the planning data as we began to better understand what was needed. That is, the AI Planner must be given a chance to produce meaningful plans: we learnt not to rush in a room and immediately kill all the NPCs. As a plan in FEAR is a totally ordered set of actions, we listed the actions of a plan in their order of appearance. The following format was designed for an easy input to Mathematica and was eventually used for KZ3 and T3:

```
{359000, noname1817, AttackFromCover, {kWSK[TargetIsDead], 1}, 1}
```

`359000` is a millisecond time stamp since the beginning of the playing session. The plan is for the NPC named `noname1817` and is of length `1` (the last piece of information); the (only) action identifier is `AttackFromCover`, which means that, hopefully, `noname1817`'s `TargetIsDead` after this action is executed.

As simple as it may look, getting, directly from the game, the AIP data in the exact format above for any game should be considered as awkward due to various implementations schemes and the features of the languages used to implement the AI Planner[2]. It is probably quicker and easier to first extract in-game data in the easiest way, and then parse and rewrite these data to suit your analytic tools. Consequently, the raw data extracted during a playing session (both KZ3 and T3), was specifically parsed and rewritten[3] in the above format.

We finished FEAR several times and eventually sampled 17 session files (about 3 hours and 45 minutes) between level 2 (Infiltration) and level 11 (Sayonara, Sucker) (see (Black 2005) for details about these levels).

The data from KZ3 was produced by Arjen Beij from Guerilla Games, when playing (during 13 minutes and 2 seconds) the first two sections of the Scrapyard level of KZ3 (Hassan et al. 2011).

The HTN Planner of T3 is written in Kismet and some unexpected lexical parsing[4] was eventually needed to format the HTN Planning data logged in files; Troy Humphreys, from High Moons Studio, produced the data when playing 4 levels of T3 for a total of 35 minutes and 30 seconds.

We end with some remarks about the AIP data:

- **Time Stamp** We used milliseconds in FEAR and the AIP component is so fast that the time stamps of a (very) few series of consecutive plans have the same numeric value. We thus advise a counter with a higher resolution: the KZ3 counter has `79,800,000` ticks per seconds and the T3 counter has `10,000` ticks per seconds.
- **Initial situation and Goal** were never recorded. Goal would help detecting whether re-planning happens; that is, several consecutive plans are generated for the same NPC, but is it for different goals or for the same one? The initial situation results from sensing: when the sensing is wrong, the generated plan can be wrong, although rightly achieving the goal we want; for instance the NPC can use a weapon not adequate to the situation. Consequently, recording the initial situation can be useful for sensor tuning, but also for an off-line study of alternative plans: what plans could be generated from this initial situation?
- **Level identifier** can be useful when processing multiple data files; for instance, it may happen the same NPC name appears across several data files: should we distinguish these occurrences?

The next section presents the measures which we performed on the AIP data.

AI Planning Metrics

AIP Metrics are measures of the kind of planning the AI Engine provides to players when they interact with the game; in that sense, the AI Engine assists the player to enjoy a playing experience. The purpose of these metrics is to help the development team discover/understand/confirm the kind of planning which happens in the game.

Active Concurrent NPCs Game AIP is not as continuous as a web service: the Peak Concurrent Users metric (Seif El-Nasr, Drachen, and Canossa 2013, (4.4.3)) is less crucial here but nevertheless reports on the AIP load: how often shall the planner switch from one NPC to another? Should the planner plan for so many/few NPCs? The answer is the first step into planning culling. Figure 1 shows the whisker boxes for the number of active concurrent NPCs (an NPC is active in-between its first and last call for planning); on average, FEAR and T3 are similar but the planning in FEAR handles up to 15 NPCs (cf. next metric). As in Figure 1 the planning in KZ3 appears to handle more NPCs, Figure 2 shows how these NPCs flow along the playing session, forcing the player to manage several NPCs at a time (5 on average and up to 10 from about 350 to 380 seconds).

Number of Plans per NPC We are looking for maximum and minimum values as another measure of the planning

[2]Let aside the problem of partially ordered set of actions which this format cannot represent; the HTN Planning component of KZ3 and T3 produce, at the more concrete level, totally ordered plans.

[3]For instance, Mathematica uses "_" and ":" as macro-characters for typing and assigning default values, respectively: these characters (and others) had to be discarded.

[4]For instance, the same action identifier had slightly different spelling in a log file: "_" could appear in front of the action identifier and the case of a few characters could change.

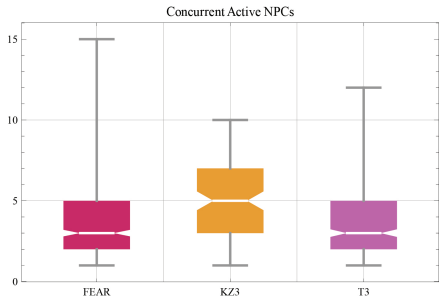

Figure 1: An NPC is active between its first and last call for planning; on the y-axis is the number of active NPCs, regardless of their simultaneous (i.e. same frame) call for planning, as this also depends on the AI update rate of the game; for instance, the AI in KZ3 is updated every 200 milliseconds (van der Beek 2007).

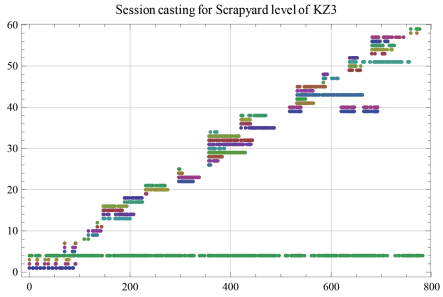

Figure 2: The flow of NPCs for a playing session; the x-axis is the time (seconds) and y-axis represents NPCs as they appear in the game, calling for planning: there is a dot for each plan generated at some point in time for a given NPC. This casting is typical of the three games: successive waves of enemies, a buddy/partner NPC sticking to you most of the time (`Rico` in KZ3, at $y = 4$), and some very similar behaviours (here the drones, for instance at the beginning of the level, but also the two Ship Turrets at $y = 39$ and $y = 40$). Colours are only made to distinguish between lines of dots.

load (cf. Figure 3). For instance, is it normal that 1985 plans are generated for T3's `AICTLaserbeak`? Yes it is; and so it is for the 463 plans of the player's mate `Rico` in KZ3. But the 393 plans for the `Rat02` of FEAR are questionable: indeed, the planner keeps generating plans for `Rat02` far after the player left it in one of FEAR's freaky tunnels; this is why FEAR handles up to 15 NPCs (cf. Previous metric). On the other bound of the range, there are NPCs needing only very few plans: do they worth it? Maximum and minimum values are the second step to planning culling.

Planning Speed is our final measure of the use of the AIP resource. This value is a function of time on the contrary of the two previous metrics. We define the speed of planning per NPC as the number of plans divided by the duration (in seconds) where plans were generated for a given NPC. Figure 6 shows planning has been going faster, first from FEAR to KZ3 and then from KZ3 to T3. We may wonder

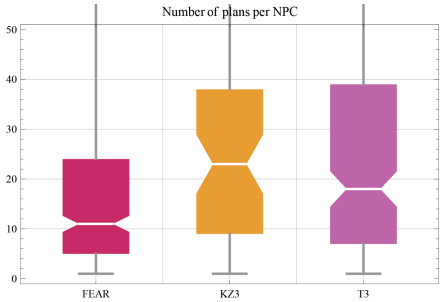

Figure 3: The maximum values, removed from this graphic for readability reasons, are the following: 393 plans generated for NPC `Rat02` in FEAR (left), 463 plans for NPC `Rico` in KZ3 (center), and 1985 for NPC `AICTLaserbeak` in T3 (right).

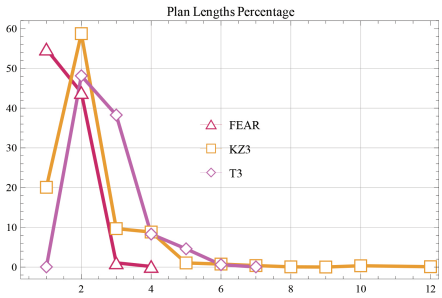

Figure 4: Plan length (number of actions in a plan) on the x-axis; for example, about 38% of the plans logged in the T3 session files are of length 3. The first AIP pattern here is that (al)most (all) of GOAP plans are short and HTN planning produces the longest plans; the second is that, the longest plans are for snipers (KZ3 and T3) and patrolling purposes (FEAR), and for the three games, the shortest plans are for turrets, drones and animals.

whether this result has to do with the AI update rate (200 milliseconds (van der Beek 2007)) of KZ3: the number of Concurrent Active NPCs metrics shows KZ3 handles more NPCs on average than the other two games. But the 75% quartile value of the planning speed is 0.8 for KZ3 and 2.6 for T3, that is more than 3 times, whereas the 75% quartile value of the number of Concurrent Active NPCs is 7 NPCs for KZ3 and 5 for T3: for 75% of the values, planning thus is three times faster in T3 than in KZ3 although T3 handles only $\frac{5}{7}$ (71%) less NPCs. The question raised here is where the planning speed should go? More plans per NPCs or else more NPCs on screen?

Plan Lengths We can expect that the longer the plan the longer the search for it. This also why, probably, GOAP's plans are the shortest (cf. Figure 4) as GOAP implements a classical search procedure in a space of states; indeed, the HTN Planning components of both KZ3 and T3 allows for several actions to be included at once in a solution plan, thus eventually generating longer plans. Note that for these two games, longest plans are generated for snipers and for the three games, shorter plans are built for animals, drones and

turrets.

Actions (numbers, frequency, category, and usage) There are 55 actions logged in the session files from FEAR, 44 and 137 in the session files of KZ3 and T3, respectively. We can expect that not all the available actions have been recorded during these sessions. As each game is a First Person-Shooter, each logged action roughly fits into one of the following category: defensive (lowering danger), offensive (attacking the player), and intermediate (patrolling, surveillance and animation purposes). The following table gives the number of actions of each category for the three games:

	Defensive	Offensive	Intermediate
FEAR	11	16	28
KZ3	5	6	33
T3	18	12	107

Distinguishing between offensive and defensive actions may be controversial. But despite mistakes when categorizing fighting actions, the number of actions designed for fighting purposes is less than the number of non-fighting actions (about a third for both KZ3 and T3) although we here deal with First Person-Shooters.

Figure 8 presents how many times actions are used. Despite a huge variation in the total number of action occurrences, the highest action occurrence is about the same for the three games (about 2400); for FEAR and KZ3 this "highest" action has to do with the game engine mechanics whereas it is a movement action (`FlyTo`) for T3. But the noticeable result here is that half of the available actions are poorly used. Several actions occur only 1 or 2 times in all the sessions and it looks reasonable to wonder whether they should be implemented at all. A final remark, illustrated by Figure 5: action costs used as search heuristics in GOAP do not affect the number of occurrences of actions when playing FEAR.

Planning Frequency as a Function of Time How often is the AI planner called? That is, what is the time-pressure on the planning component? Figure 7 shows the planning frequency in KZ3, both from an absolute and an NPC perspective. Such a graphic is very similar for the three games. The unexpected result, although this is rarely the case here (i.e. the frequency is close to 0%), is that some NPC appear to take their time between two calls to the AI Planner; this is for instance the case for Ship Turrets in KZ3 (cf. horizontal lines at $y = 39$ and $y = 40$ of Figure 2.

The next section presents four patterns we discovered from assembling AI Planning data and interpreting metrics.

AI Planning Patterns

AIP Patterns are combined results from AIP measures. For instance, the highest action usage for one game is not a pattern; the fact that it is about the same value for the three games is a pattern for the use of AIP in a First-Person Shooter. The previous sections suggested a few patterns already, but these can be interpreted as extensions of metrics. Here are structures built from the interpretation of metrics.

Hyperbolic Planning Frequency Following the Planning Frequency metric of the previous section, we first ask whether its hyperbolic shape of Figure 7 is a constant for First-Person Shooters. Indeed, for the three games we studied, the time pressure is high for small amount of time and then decreases regularly down to very low frequencies for long intervals between two calls for the AI Planner: 529.75, 80.52, and 22.19 seconds in FEAR, KZ3, and T3, respectively. Again, there are questions for extreme values of an hyperbolic curve: When the planning frequency is high and the plan is the same (e.g. rats in FEAR), is AIP the best component to achieve the behaviour of this NPC? When the frequency is very low, is this NPC really needed? In FEAR, there is also the case of NPC `Delta01` which appears in Level 9 right after a dogfight in a lobby (Black 2005), and keeps calling the AI Planner at about 2 and 4 seconds for animation purposes.

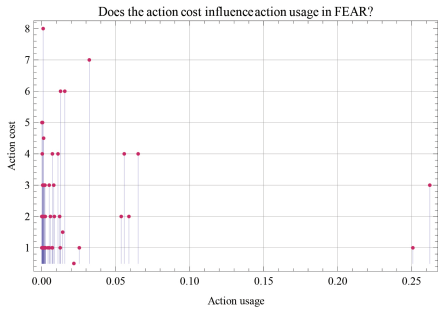

Figure 5: x-axis represents the action usage (cf. Figure 8) and y-axis is the action cost; the 55 (red) dots of this graphic represent the 55 actions logged in the FEAR sessions. We can observe that despite (relatively) low costs for the two highest action usage, there is no clear influence of the action cost over the action usage.

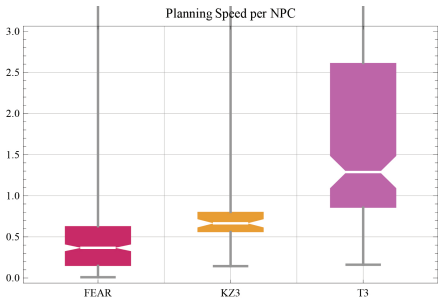

Figure 6: The y-axis represents the Planning Speed per NPC, i.e. the number of plans per second per NPC; the maximum planning speed is 8.5 for FEAR (left), 3.4 for KZ3 (center), and 59.9 for T3 (right). The median value roughly doubles from FEAR to KZ3 and then from KZ3 to T3: AIP has been going faster since 2005.

Fixed Plans are plans which occur with the same sequence of actions. KZ3 and T3 use HTN Planning and fixed plans can be understood as HTNs; thus, it is not so surprising that we find fixed plans in their session files. But it happens that the FEAR session files also contain fixed plans. There are, respectively, 10, 11, and 46 fixed sequences of actions in

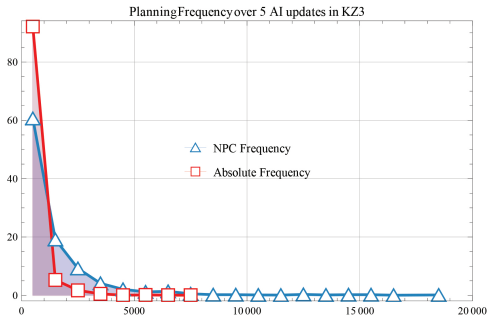

Figure 7: The x-axis represents time in milliseconds. How often is the AI planner called? For instance, the second red square, starting from the left, tells us the AI Planner is called every 1.5 seconds about 5% of the time; the second blue triangle tells us that for a given NPC, the AI Planner is called every 1.5 seconds about 20% of the time.

FEAR, KZ3, and T3. But most of these fixed plans occur only once; the following table gathers, from FEAR, KZ3 and T3, in that order, the three fixed plans occurring the most together with their frequency (6679, 2349, and 8751 plans were logged in the FEAR, KZ3, and T3 session files, respectively):

Fixed Plan's sequence of actions	Frequency
{`GotoNodeOfType`, `AttackFromView`}	$\frac{13}{6679}$
{`SurveyArea`, `GotoTarget`, `InspectDisturbance`}	$\frac{51}{6679}$
{`GotoNode`, `UseSmartObjectNode`}	$\frac{2355}{6679}$
{ `remember.activeplan`, `staggerfire`}	$\frac{1}{87}$
{ `remember.activeplan`, `lapapeekatentity`}	$\frac{4}{261}$
{ `remember.activeplan`, `scanwaypointlistlapa`}	$\frac{115}{2349}$
{`StartStrafing`, `StrafingIdle`}	$\frac{199}{8751}$
{`Asleep`, `Idle`}	$\frac{626}{8751}$
{`ChooseNewAttackPoint`, `FlyTo`}	$\frac{1838}{8751}$

We will not discuss the content of these plans but question their existence: instead of searching for these plans, shouldn't we instead put these fixed plans in some hashing table, indexed by the initial and goal situation? The answer is a step into the architecture of the AI Engine which can not only search for plans but also fetch plans.

Pareto principle for Game AIP If we sum 20% of the highest numeric values of the action usage, we get roughly 80% of the cumulative action usage for all three games: 86%, 80%, and 82% for FEAR, KZ3, and T3, respectively. That is, 20% of the actions of a First-Person Shooter own 80% of the occurrences in the session files. The main argument against this pattern here is that the Pareto principle is valid when the data set is huge (T3 illustrates Figure 9 because it has the highest number of action occurrences: 19187). In this case, the obvious question is again that of the amount of development effort that should be spent on the remaining 80% of available actions which (only) impact 20% of the occurrences. In a dual spirit, the 20% actions which are used the most should be checked and optimized, in particular when their purpose is animation.

In the next section, we gather the questions of this paper and briefly discuss how answers can be turned into decisions for Game AIP.

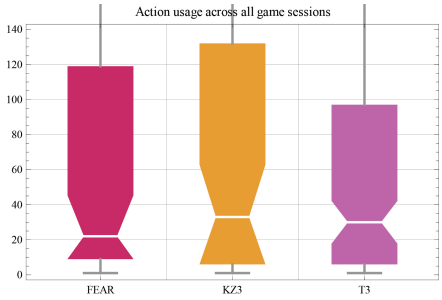

Figure 8: The y-axis is the number of occurrences of a given action identifier across all sessions; there are 9781 action occurrences in the FEAR session files, 6371 for KZ3, and 19187 for T3. The maximum number of occurrences for one action, removed from this graphics for readability reasons, are the following: 2566 (26%) for action `UseSmartObjectNode` in FEAR (left), 2355 (37%) for action `RememberActivePlan` in KZ3 (center), and 2305 (12%) for action `FlyTo` in T3 (right); it is remarkable these maxima are so close. However, we believe the most important values are the median values: 50% of all available actions occur only at most 22 times in FEAR, 33 times in KZ3, and 30 times in T3; it is again remarkable these values are so close, but with such a little use in the game, how many of these actions really worth the development effort?

AI Planning Intelligence

Along with the AIP metrics and patterns, we formulated, in that order, the following questions:

1. How often shall the planner switch from one NPC to another?
2. Should the planner plan for so many/few NPCs?
3. There are NPCs needing only very few plans: do they worth it?
4. Where should the planning speed go?
5. More plans per NPCs or else more NPCs on screen?
6. How often is the AI planner called?
7. What is the time-pressure on the planning component?
8. Is AIP the best component to achieve the behaviour of this NPC?
9. When the planning frequency is very low for a given NPC, is this NPC really needed?
10. Shouldn't the fixed plans be stored in some hashing table, indexed by the initial and goal situation?

Answers to these questions can support decisions on the following topics, reporting to game design and production during iterations:

- AIP development effort (e.g. questions 3, 4, 5, and 9),
- AIP technical specifications (e.g. questions 6, and 7),
- AI Engine features (e.g. questions 4 – because the AI update rate might influence the planning speed, 8, and 10).

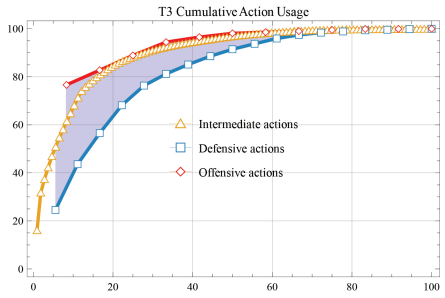

Figure 9: The x-axis represents an action percentage in decreasing order of action usage and the y-axis is the cumulative action usage in T3. The purpose here is to answer the following question: how many actions (in percentage of all actions occurring in the session files) are needed to reach a given percentage of action usage? Whereas offensive actions are based on a few highly used actions (2 actions to reach more than 80% of cumulative usage), there is more diversity in the case of defensive actions (6 actions are needed to reach about 80% of cumulative usage). The results from T3 are shown here because this game has the highest number of action occurrences (19187); 20% of all available actions in T3 own 82% of the cumulative action usage.

Conclusion

What did we learn about AIP in First-Person Shooters? We can expect Game AIP to handle several tens of actions in order to build plans with at most 10 actions and smaller length on average (1 to 5 actions). For instance, GOAP plans are rather short (1 or 2 actions) while HTN Planning builds longer plans (2 to 5 actions); long plans seem dedicated to specific tasks (patrolling, sniping) while the shortest plans are for turrets, animals and drones. Moreover, some specific NPCs and actions use AIP more than others. Rather unexpectedly, most of the AIP time and memory budget in First-Person Shooters is not spent into fighting actions but into actions whose purpose is patrolling and animation. Planning speed is the only metric showing the increase of Game AIP over the years. Finally, alternatives to AIP should be considered for repetitive actions (i.e. plans of length 1) and fixed plans (i.e. plans appearing the same sequence of actions).

Perspectives A study of GOAP in its latest implementation would be very valuable. Indeed, what has happened to GOAP since 2005? Wait. There is *Tomb Raider*. Crystal Dynamics, please, get in touch!

References

Black, F. 2005. *F.E.A.R. PRIMA Official Game Guide*. PRIMA Games.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Hassan, A.; Byrne, B.; Andrews, C.; Niazi, U.; and Murray, W. 2011. *KillZone 3 Official Guide*. Future Press.

Humphreys, T. 2013. Exploring HTN planners through examples. In Rabin, S., ed., *Game AI Pro*. CRC Press. chapter 12, 149–167.

Monolith Productions. 2006. F.E.A.R. public tools.

Orkin, J. 2006. Three States and a Plan: The A.I. of F.E.A.R. In *Proceedings of the Game Developer Conference*, 17 pages.

Seif El-Nasr, M.; Drachen, A.; and Canossa, A. 2013. *Game Analytics – Maximazing the Value of Player Data*. Springer.

van der Beek, J. 2007. Autonomous squand behaviour in a virtual game environment. Master's thesis, Vrije Universiteit of Amsterdam.

Wolfram, S. 1999. *The Mathematica Book*. Cambridge University Press.