# HTN or State Space - Who Should Do Planning in Your Game?

Martin Černý, Jakub Gemrot
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00, Prague 1, Czech Republic
E-mail: `cernym@gmail.com, gemrot@ksvi.mff.cuni.cz`

**KEYWORDS**
Action Selection, Planning, HTN, Comparison

## ABSTRACT

There is an ongoing discussion in the game AI community, whether to use AI planning for controlling non-player characters (NPCs) in computer games. Recent years have seen implementations of both state space and hierarchical task network (HTN) planning in AAA game titles, each having specific advantages and disadvantages. This paper is concerned with the performance aspect of both technologies and proposes a general methodology for comparing performance of action selection mechanisms. Two case studies comparing NPCs controlled by the JSHOP2 HTN planner and SG-Plan 6 and Metric-FF state space planners in a game-like competitive environment are presented and their results are statistically analyzed. First environment is a puzzle-like scenario with minor combat element and strong competition for limited resources. Second environment features two pairs of cooperating agents in a combat scenario inspired by the Gears of War series. It is concluded that in puzzle-like domains HTN planning is inferior to state space planning, but for the second domain, the results are not conclusive.

## INTRODUCTION

A large number of contemporary games consist of a dynamic, continuous real-time world inhabited by the player and other non-player characters (NPCs). This is especially true for first person shooters (FPS) and role playing games (RPG) which represent a significant share of the game market. Many serious games also have similar properties. Since graphical representation of game worlds has become very close to reality, the perceived intelligence of the NPCs is starting to be of vital importance to maintain believability.

One of the cornerstones of AI design for NPCs in games is the action selection problem — what to do next? The traditional game industry answer has been to use either behaviour trees (Champandard 2007) or finite state machines (Fu and Houlette-Stottler 2004), providing essentially fully preprogrammed responses to various external stimuli. But this is slowly changing, the need for smarter AI and lower design complexity has raised demand for alternative approaches, especially various forms of AI planning. Both STRIPS-style (Fikes and Nilsson 1972) and HTN planning (Ghallab et al. 2004) have already been implemented in multiple AAA game titles (Champandard 2013).

There is a major lesson learnt during the process of adapting AI planning as proposed by academia to controlling NPCs in computer games: while planning is considered a mature technology from the academic point of view, its application to NPC control is far from straightforward and it is often not an appropriate technique. On the other hand, from the number of succesful implementations it seems that planning turned out to be beneficial for game AI — at least in some scenarios. So our first question is: can we determine and classify scenarios where planning is — in some sense — better than reactive techniques? And can we find out when HTN planning is more suitable than STRIPS-style planning?

To answer these questions, it is neccessary to first formulate what "better" actually means. In context of action selection mechanisms (ASMs) for NPCs there are at least three points of view relevant to game AI: first is the *performance perspective* — how good is the AI at achieving its goals in the environment? Or, since computing power is a scarce resource for game AI, what is the tradeoff between performance and resources needed? Second is the *designer perspective* — how difficult is it to design AI using the technique, how much time and knowledge or skills are needed? Third is the *entertainment perspective* — whether it is actually fun to play a game with such AI?

In this paper we propose a general methodology for comparing various ASMs from the performance perspective, but we also briefly touch the designer perspective. The entertainment perspective has been left out, because the perceived entertainment is to a large extent influenced by many factors other than the AI itself which imposes large methodological complications to the research.

To demonstrate the methodology, we present two case studies evaluating HTN and STRIPS-style planning in game-like domains. One of the scenarios is more puzzle-like with indirect competition, while the other is a highly dynamic combat situation incorporating multibody coordination.

The rest of the paper is organized as follows: first, the two planning formalisms studied are presented, then related work is discussed and the methodology for our experiments is given. Next, the individual case studies are described and their results are discussed. The paper ends with conclusions drawn from the results and discussion of future work.

## PLANNING FORMALISMS

In this section we briefly introduce the two AI planning approaches that we have investigated: STRIPS-style planning and HTN planning.

### STRIPS-style Planning

STRIPS-style planning defines a planning domain in terms of states and actions. Actions are defined by their preconditions and effects. Preconditions specify states in which the action is applicable. Effects describe how the world state changes once the action is performed. A planning problem consists of a domain, an initial state and a goal condition. A solution to the problem is a sequence of actions that is applicable to the initial state (all preconditions of actions are met at the time of their execution) and results in a state that meets the goal condition.

In the most widely used planning formalism - PDDL (Fox and Long 2003), the state consists of a truth assignment to a fixed set of logical atoms. However such a representation is often unneccessarily large as many pairs of atoms are mutually exclusive (e.g. `at(home)` and `at(garden)`). For this reason many PDDL planners internally represent state by values of a set of discrete state variables whch are automatically inferred from the original domain definition (e. g. Metric-FF (Hoffmann 2003)).

There are multiple ways to solve STRIPS-style planning problems. Currently, the fastest planners perform *state-space planning* — a heuristic search over the state space starting from the initial state. A nice property of this approach is that complete state information is always available for checking action preconditions and thus even complicated preconditions can be easily handled.

The most known planner used in games, GOAP (Orkin 2006) works with state-variables and performs a state-space search.

### HTN Planning

HTN planning takes a rather different approach. A HTN planning domain consists of two types of tasks: primitive tasks, which correspond to actions in STRIPS-style planning and composite tasks. A composite task corresponds to a higher-level action and provides several decompositions — recipes how to accomplish the task with lower-level actions. Decompositions may be accompanied with conditions for their applicability. A planning problem in HTN is to gradually decompose a given task into a (partially ordered) sequence of primitive tasks such that all the decomposition conditions are held and the resulting action sequence is applicable to a given initial state.

## RELATED WORK

AI planning has been involved in multiple commercial games. For a thorough review of both STRIPS-style nad HTN planning in commercial games see (Champandard 2013).

Some comparison studies of planning and reactive techniques have been made (Hoang et al. 2005, Long 2007, Cartier 2011). In general, planning techniques are found to be useful from the performance perspective if the environment is not too fast paced. In rapidly changing environments the results are less conclusive. Also only one of the studies involved statistical analysis of the results to distinguish between possible random coincidence and "real" difference.

A simple comparison of PDDL planners and reactive techniques in various dynamic conditions in a 3D game-like environment has been performed in our previous work (Cerny et al. 2013). It showed that, unsurprisingly, planning is advantageous only when the environment is small or it is not very dynamic or if it is very hostile. However the work tries to pinpoint the exact transition points and has let me test a statistical methodology to evaluate similar experiments.

We are not aware of any direct comparison of HTN and STRIPS-style planning.

Some research has also been done in comparing usability of agent programming languages to develop virtual agents (Gemrot et al. 2013). The methodology devised in the paper and previous work of the authors is inspirative for comparing algorithms from the designer point of view.

## COMPARISON METHODOLOGY

In this section we present a general methodology for comparing ASMs. We mostly focus on comparisons from the performance perspective, but a short discussion of designer persepctive evaluation is also given.

For the purpose of computer games, it does not make much sense to evaluate the ASM by solving isolated problems. Thus we propose to incrementally create a broad set of different game-like environments, primarily multi-agent, to test the ASMs. Our previous comparison was done in full-fledged 3D game engine (Cerny et al. 2013), but creating complex scenarios in 3D continuous environments showed to be very time consuming. Thus for further research we have decided to simplify the environments to 2D simulations with game-inspired mechanics but running in discrete space and time. After sound results have been established for those simpler environments a confirmative evaluation should be done in fully 3D and continuous environments to prove that the results do generalize.

The agents carry out actions in the environment and are given a real-valued reward signal at every time step. The performance of the agents is measured in two ways: the total reward accumulated during the simulation and the rank — whether it was better than other agents. Those data will be statistically rigorously analysed to determine significance of perceived differences.

Environment simulation is fully dynamic, i. e., it never waits for the agent to make a decision. One of the interesting questions is how does the performance of various algorithms change with varying speed of the simulation — that is, with different amount of processing power available to the agents. In the first part of the research, only fully observable domains will be considered, but some experiments with partial observability will also be run.

The action selection algorithms are wrapped up in *controllers*. A controller is responsible for running the algorithm, performing meta-reasoning about the deliberation process (e. g. interrupting a planner if the world has changed signicifantly since the start of planning) and gathering the results of the algorithm run.

Every environment has multiple *representations*. A representation is an interface between an environment and a controller. There may easily be multiple different representations (e. g., on a different level of abstraction) of the same environment for the same controller. The representation provides the controller with input data for the action selection algorithm and also provides information for controller's meta reasoning (e. g., what are the possible planning goals and what are their priorites, has the world changed considerably in last $x$ steps, ...). The representation also translates high-level actions from the controlling algorithm (if any) into simple reactive plans that can be executed without deliberation (e. g., move to next room, shoot while the enemy is visible, ...). Thus the algorithms effectively do not need to be aware of the actual environment. The representations need to be carefully designed not to create significant advantage for any particular controller by translating its response into more powerful reactive plans.

An important part of the project output is a freely available codebase of environments and controllers [1]so that anybody can a) rerun the experiments and reproduce the results, b) test a new algorithm against any of the already connected, provided it can handle data in one of the supported formats and c) add a new environment to test the algorithms against as long as relevant representations are added.

The evaluation from a designer perspective is methodologically more complicated. Reliable results are very difficult to obtain without user testing (Gemrot et al. 2013). Due to high associated costs, user testing was not performed as a part of this study and was left for future work. However, personal experience from developing individual representations are an indicator of the actual difficulties

[1]The work-in-progress codebase is available at `http://code.google.com/p/aiste/`

## CASE STUDIES

We have performed two sets of experiments to compare HTN planning using JSHOP2 (Ilghami and Nau 2003) and two PDDL planners: SGPlan 6 (Hsu and Wah 2008) and Metric-FF (Hoffmann 2003).

For the purpose of our experiments we have modified JSHOP2 code slightly. We have added a simplistic branch and bound optimization scheme to the original code and performed some technical adjustments for better integration in our experimental platform. JSHOP2 was chosen because it is in widespread academic use and has been repeatedly used in connection with game AI. JSHOP2 is written in Java.

SGPlan 6 and Metric-FF were chosen because they are in frequent academic use and have scored well at past International Planning Competition (IPC). Both planners use the PDDL language (Fox and Long 2003) as their input. The winner of the latest IPC (held in 2011), LAMA 2011 was based on the Fast Downward platform, which has shown severe obstacles for real-time planning in our previous study (Cerny et al. 2013) and would require different environment representation for efficient planning under severe time constraints, which we could not provide.

Both environments and all supporting code was written in the Java language. All experiments were run on a dedicated computing server with two AMD Opteron 2431 processors (6 cores each, 2.4GHz, 64bit) and 32GB RAM. Experiments were run in parallel while ensuring that each environment and each planner instance can occupy a full processor core without competing with each other.

### Case Study I - Spy vs. Spy

The first domain is inspired by the 1984 computer game SpyVsSpy (Anon. author 2012). There are two agents moving on a grid of rooms. The rooms may contain one or several items, traps, trap removers and weapons. There are multiple types of items, traps and trap removers. The agents have the same goal: gather one instance of each of the item types and reach a goal destination. However, if there is a trap in a room, it has to be disarmed with a corresponding trap remover, otherwise an agent trying to pick up anything in this room dies.

An agent carrying a weapon may attack its opponent if they are in the same room. Such an attack has a 30% probability of killing the other agent. Weapons are

for single use only. Whenever an agent dies, all its belongings are “left on the floor”. Agents are rewarded -50 points for death, 150 for reaching the goal with all items and -1 for each step of the simulation. The simulation ends when one of the agents reaches the goal or after a timeout.
Maps with 15, 30 and 70 rooms were randomly generated, 15 instances per size. The generation process ensured, that in the static case (i. e. if only one agent is present in the environment) the goal state was reachable for any starting position. Simulations were run at three different speeds - 100, 500 and 1000ms delay between steps. For each map and simulation speed, every pair of controllers was run twice. For the second run, the initial starting positions were switched to ensure fair comparison. This made for a total of 810 experiments.

*Results*
Let us first examine mean reward generated by each controller. There are two ways to examine reward. First is to examine the mean reward over all experiment runs, second is to calculate the *win percentage* that is the percentage of runs where the controller gathered greater reward than its opponent. The results for each controller pair are summarized in Table 1. We see that SGPlan is better than JSHOP2 and JSHOP2 is in turn better than Metric-FF and that the differences are huge. Indeed, the null hypothesis that two controllers are equally likely to gather greater reward in a single run has p-value $< 10^{-15}$ (tested using multiple comparisons of means with Tukey contrasts (Hothorn et al. 2008) over an ANOVA fit with a first order generalized linear model).
It is of interest, that when averaging over runs in a particular map size or a particular step delay the ordering is exactly the same and mean rewards and win percentage are roughly similar (less than 10% difference). Also the statistical significance of the differences is still high ($p < 10^{-5}$ in all cases).
From the designer perspective JSHOP2 was, in our personal experience, much more difficult to handle than PDDL planners. Developing PDDL representation for SGPlan and Metric-FF was very straightforward - in a discretized and nearly deterministic environment it is simple to describe individual actions. Developing HTN representation for JSHOP2 on the other hand was much more difficult. A direct decomposition resulted in performance inferior by a large margin to both PDDL planners. Involving domain-specific heuristics for ordering possible variable bindings and external A* path finding were neccessary for JSHOP2 to be competitive.
Domain-specific heuristics allowed JSHOP2 to find relatively good plans, but the planning time was still too high (432ms for maps of size 15, 647ms for maps of size 30 and 493ms for maps of size 70). The numbers should however be interpreted carefully: only succesful planning runs (those that found a plan) are included in the statistic, because there are often plenty unsuccesful runs which finish very quickly. However, succesful planning runs do not include runs which were terminated by the end of the simulation or radical change in the environment state and those interrupted runs tend to be longer than average (they ran long enough to be interrupted). This explains the decrease in succesful planning time for JSHOP2 in the largest domains. See the online appendix to this paper (Cerny 2013) for full planning statistics data.
The percentage of steps idle (steps agent had no plan and couldn’t do anything) is a good indicator of planning performance. Metric-FF had huge issues even in the smallest domain (87% idle) while the other two planners had oscillated around 30% steps idle. The variance of JSHOP2 planning times was remarkably high (sd over 79% of the mean planning time for all map sizes), while SGPlan 6 has relatively stable performance (sd below 20% of the mean planning time for all map sizes).

*Discussion*
In Spy vs. Spy scenario, PDDL planning is more beneficial than HTN planning with JSHOP2. PDDL domain was easier to design than the corresponding JSHOP2 representation, yet with a correct planner, better performance was achieved. This is most likely due to relatively static and puzzle-like nature of the environment, but implementation details of JSHOP2 and the fact that it is in Java could have also played a role.

## Case Study II - Cover Game

The second environment is called Cover Game. It is a combat scenario inspired by the Gears of War game series (Epic Games Inc. 2013) and also takes elements of gameplay from XCOM: Enemy Unknown (Firaxis Games 2013). In the scenario two pairs of cooperating agents fight each other with ranged weapons on a square grid map with lots of cover spots. Each pair of agents is controlled by a single controller, issuing actions for both bodies. In each turn, the agent may either move up to three squares, shoot at a visible enemy, suppress a visible enemy with fire (at most once per 2 turns) and enter full cover mode, if already in cover (full cover mode significantly increases cover bonus until the agent moves or fires). The probability of a succesful shooting is influenced by the distance to the target and its cover status. Agents that are a target of supression suffer a large aim penalty, but are not hurt. The damage done by a single shot is non-deterministic, on average 3 hits are neccesary to eradicate a target. A killed agent is respawned at one of the predefined locations. The team is rewarded +1 point for each kill and -1 for each death (i.e. it is a zero-sum game) and the game ends after a specific amount of turns. The agents have unlimited ammo and start to heal slowly if they are not hit for two turns.

Table 1: Overall Results for Spy vs. Spy. for every controller pair. The win percentage and mean reward along with standard deviation (in brackets) is reported.

| **Controller** | **Opponent** | **% Win** | **Reward** | |
|---|---|---|---|---|
| | | | **Controller** | **Opponent** |
| SGPlan 6 | Metric-FF | 92 | 113.10 (46) | -24.45 (47) |
| SGPlan 6 | JSHOP2 | 86 | 92.91 (70) | -21.13 (53) |
| JSHOP2 | Metric-FF | 87 | 92.94 (56) | -28.73 (64) |

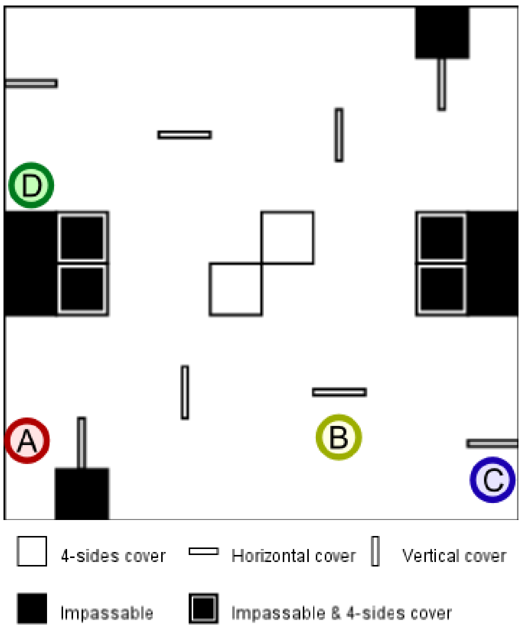

Figure 1: A sample scenario in Cover Game. 4-sided cover spots shield all four neighbouring tiles, while horizontal and vertical cover spots shield only the neighbouring tiles in the correct direction. Cover spots provide 180° shelter, e.g. agent B (yellow) is not covered from agent A (red), but agent C (blue) is. Agent D (green) cannot shoot at any other agent, neither can any other agent shoot at him because the line of fire is blocked by an impassable square.

Three manually created maps of different sizes were tested. The smallest one (10x10 tiles, called "Simple") is shown in Figure 1. There was also a larger map (22x20 tiles, called "Irregular", because it is not symmetric) and a very large map (27x37 tiles) inspired by the "Security" multiplayer map from Gears of War II.

Several different representations of the environment were developed. Two variants of low-level PDDL representation, where single action in the plan corresponds directly to a single action in the environment were tried. One forced the planner to alternate actions for one agent with actions for the other agent (this will be referred to as "LowAlt") and one which did not ("Low"). The idea was that forcing the planner to alternate would make it more difficult to find a goal, but the plan should complete faster, because of maximal parallelism.

A similarly low-level JSHOP2 representation was not tested in the final runs because even representation capturing only movement and finding cover had difficulties finding a single plan in a small environment within several seconds. Instead more high-level representations for both JSHOP2 and PDDL were created where the actions correspond to assigning "roles" to individual agents. The set of roles were the same for both representations and the domains were designed so that very similar role assignments are found by both representations. PDDL planners were thus tested with three representations. Since we are mostly interested with comparing techniques, matches between two instance of the same controller with different representations were excluded.

For each pair of controllers and representations, 4 rounds 200 simulation steps each were run at 100, 500, 1000 and 2000ms per step, resulting in total of 720 experiments.

*Results*

Results were analyzed with the same methods as in Case Study I, so refer there for details. A significant amount of draws were met, so the summary of draw percentages was added. The results are presented in Table 2. The message is clear, all of the role representations are significantly (all p< $10^{-4}$) better than the low-level ones, regardless of controller. The differences between controllers with the same representation level are however insignificant (all p> 0.7).

The same pattern of results, although not always statistically significant, is repeated in all map sizes and step delays. This spares us strong conclusions, except for the fact that low-level representations are not practical in highly dynamic and non-deterministic domains.

However, looking at planning statistics reveals some interesting points. Most notably JSHOP2 plans very quickly (below 10ms). It was not enough to give him statistically significant performance advantage probably because the PDDL planners planned only slightly above 100ms which was the shortest step delay. The most likely cause is that PDDL planners are started in a separate system process and thus had the overhead of starting the process, while JSHOP2 ran in the same virtual machine as did the environment. See the online appendix to this paper (Cerny 2013) for actual numbers.

Table 2: Overall Results for CoverGame for every controller pair. The win and draw percentages and mean reward along with standard deviation (in brackets) are reported.

| Controller | Opponent | % Win | % Draw | Reward | Significant |
|---|---|---|---|---|---|
| SGPlan 6 - Roles | Metric-FF - Low | 79 | 6 | 4.42 (5) | yes |
| SGPlan 6 - Roles | Metric-FF - LowAlt | 69 | 6 | 3.92 (6) | yes |
| SGPlan 6 - Roles | Metric-FF - Roles | 50 | 6 | 0.56 (6) | no |
| SGPlan 6 - Roles | JSHOP2 - Roles | 48 | 10 | 0.33 (5) | no |
| JSHOP2 - Roles | Metric-FF - Low | 79 | 10 | 5.21 (6) | yes |
| JSHOP2 - Roles | SGPlan 6 - Low | 77 | 12 | 4.65 (5) | yes |
| JSHOP2 - Roles | Metric-FF - LowAlt | 75 | 12 | 4.12 (4) | yes |
| JSHOP2 - Roles | SGPlan6 - LowAlt | 71 | 19 | 4.29 (5) | yes |
| JSHOP2 - Roles | Metric-FF - Roles | 48 | 10 | 0.27 (5) | no |
| Metric-FF - Roles | SGPlan 6 - Low | 77 | 8 | 5.54 (7) | yes |
| Metric-FF - Roles | SGPlan6 - LowAlt | 77 | 12 | 4.58 (6) | yes |
| SGPlan 6 - Low | Metric-FF - Low | 42 | 21 | -0.06 (4) | no |
| SGPlan 6 - Low | Metric-FF - LowAlt | 41 | 29 | 0.37 (4) | no |
| SGPlan6 - LowAlt | Metric-FF - LowAlt | 38 | 29 | -0.08 (3) | no |
| SGPlan6 - LowAlt | Metric-FF - Low | 35 | 38 | 0.58 (4) | no |

From our personal experience, implementing the high-level role representations for both PDDL and HTN planners was of roughly the same difficulty. But the design of the low level domain was quite cumbersome, as the mechanics of the domain such as probability to hit are not easily represented as logical atoms.

*Discussion*

The experiment could not determine, whether HTN or PDDL planning is better, because the level of representation seemed to be the sole important factor. An interesting point is that the abstract domains with roles were only slightly more complicated than a simple behavior tree, but planners still needed a lot of time to evaluate it. For PDDL planners it is impossible to determine whether this was simply an overhead of starting a new system process, but JSHOP2 yields itself to more detailed analysis. 5ms is still a lot for a behavior tree evaluation, and translating the problem instance to JSHOP2 formalism is not included in this time. Also no string operations are performed during the translation or inside JSHOP2, as all atoms are encoded as integers. Profiling the JSHOP2 code showed, that over 90% of the time is spent working with variable bindings - although the domain would work exactly the same if variable binding changed to parameter passing. This could be of importance in practical HTN applications — if variable binding is not strictly neccessary, do not use it.

## CONCLUSIONS AND FUTURE WORK

We have outlined a methodology for comparing action selection mechanisms and performed two case studies followin the methodology. The first case study showed than in puzzle-like environments, PDDL planning seems to be better than HTN planning. Results for highly dynamic CoverGame domain were not conclusive due to low statistical significance. But still the methodology showed to be feasible to apply and provide statistically supported results about various action selection mechanisms.

We have also noted that variable binding is an important performance factor in JSHOP2, even if the domain is so simple, that it is not needed. This should be noted for practical deployments of HTN planning.

An interesting point is that performance of both HTN and PDDL planners seemed to suffer similar penalty when scaled to larger domains or when under more time pressure.

As for designer perspective, our personal experience has shown that in a puzzle-like world, designing a HTN implementation requires much more effort than its PDDL counterpart.

Next steps consist of both extending the set of environments and connecting other action selection mechanisms such as reinforcement learning implementations and off-the-shelf probabilistic planners to the environments already designed. More thorough experimentation with the two domains presented in this paper should also yield further insights into HTN and PDDL planning applicability.

As a part of the research an open-source toolkit for

evaluating various techniques in game-like scenarios is being created. The toolkit will allow researchers from diverse fields to easily apply a novel technique to game-like domains and measure its performance therein. Such toolkit would foster interdisciplinary communication and has the potential to bring a unifying view on AI evaluation in games across the whole field.
From our previous experience in evaluating techniques from the designer perspective, the most promising approach for future work is to conduct user-studies as a part of labs or examination of students of relevant AI courses. The students will try to implement representations of the same environment for the individual algorithms (e.g. a PDDL domain description, a feature set for reinforcement learning). Then it will be measured how succesful agents are with those representations and the students will be asked to both quantitatively and qualitatively describe their desginer experience.

## ACKNOWLEDGEMENTS

This work was partially supported by the student research grant GA UK 559813/2013/A-INF/MFF, by the SVV project number 267 314 and by the Czech Science Foundation under the contract P103/10/1287 (GAČR).

## REFERENCES

Anon. author, 2012. “Spy vs. Spy (1984 video game)”. URL http://en.wikipedia.org/wiki/Spy_vs._Spy_%28computer_game%29.

Cartier J.F., 2011. *Étude Comparative des Planificateurs Appliqués au Domaine des Jeux-Vidéos*. Master’s thesis, Université de Montréal, Québec, Canada.

Cerny M., 2013. “HTN or State Space - Who Should Do Planning in Your Game?: Online appendix”. URL http://popelka.ms.mff.cuni.cz/~cerny/pddl_vs_htn_online_appendix.pdf.

Cerny M.; Bartak R.; Brom C.; and Gemrot J., 2013. “Reactive and Planning Agents in Dynamic Game Environments: An Experimental Study”. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*.

Champandard A., 2007. “Understanding behavior trees”. *AIGameDevcom*. URL http://aigamedev.com/open/article/bt-overview/.

Champandard A., 2013. “Planning in Games: An Overview and Lessons Learned”. *AIGameDevcom*. URL http://aigamedev.com/open/review/planning-in-games/.

Epic Games Inc., 2013. “Gears of War”. URL http://gearsofwar.xbox.com.

Fikes R. and Nilsson N., 1972. “STRIPS: A new approach to the application of theorem proving to problem solving”. *Artificial intelligence*, 2, no. 3-4, 189–208.

Firaxis Games, 2013. “XCOM: Enemy Unknown”. URL http://www.xcom.com/enemyunknown/.

Fox M. and Long D., 2003. “PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains”. *Journal of Artificial Intelligence Research*, 20, 61–124.

Fu D. and Houlette-Stottler R., 2004. “The ultimate guide to FSMs in games”. In *AI Game Programming Wisdom II*, Charles River Media. 283–302.

Gemrot J.; Hlávka Z.; and Brom C., 2013. “Does high-level behavior specification tool make production of virtual agent behaviors better?”. In *Cognitive Agents for Virtual Environments*, Springer, vol. LNCS 7764. 167–183.

Ghallab M.; Nau D.; and Traverso P., 2004. *Automated Planning: Theory and Practice*, Morgan Kaufmann, chap. Hierarchical Task Network Planning. 229–252.

Hoang H.; Lee-Urban S.; and Muñoz-Avila H., 2005. “Hierarchical plan representations for encoding strategic game AI”. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*. 63–68.

Hoffmann J., 2003. “The Metric-FF Planning System: Translating “Ignoring Delete Lists” to Numeric State Variables”. *Articial Intelligence*, 20, 291–341.

Hothorn T.; Bretz F.; and Westfall P., 2008. “Simultaneous inference in general parametric models”. *Biometrical Journal*, 50, no. 3, 346–363.

Hsu C.W. and Wah B.W., 2008. “The SGPlan planning system in IPC-6”. *Proceedings of the Sixth International Planning Competition*, 5–7.

Ilghami O. and Nau D.S., 2003. “A general approach to synthesize problem-specific planners”. Tech. rep., University of Maryland Institute for Advanced Computer Studies.

Long E., 2007. *Enhanced NPC behaviour using goal oriented action planning*. Master’s thesis, School of Computing and Advanced Technologies, University of Abertay Dundee, Dundee, UK.

Orkin J., 2006. “Three states and a plan: the AI of FEAR”. In *Game Developers Conference*. vol. 2006, 1–18.