Planning is the Game: Action Planning as a Design Tool and Game Mechanism

Rudolf Kadlec and **Csaba Tóth** and **Martin Černý** and **Roman Barták** and **Cyril Brom**
Charles University in Prague, Faculty of Mathematics and Physics
Malostranske nam. 25, Praha 1, 118 00, Czech Republic
{rudolf.kadlec, toth.csaba.5000, cerny.m}@gmail.com, bartak@ktiml.mff.cuni.cz, brom@ksvi.mff.cuni.cz

Abstract

Recent development in game AI has seen action planning and its derivates being adapted for controlling agents in classical types of games, such as FPSs or RPGs. Complementary, one can seek new types of gameplay elements inspired by planning. We propose and formally define a new game "genre" called anticipation games and demonstrate that planning can be used as their key concept both at design time and run time. In an anticipation game, a human player observes a computer controlled agent or agents, tries to predict their actions and indirectly helps them to achieve their goal. The paper describes an example prototype of an anticipation game we developed. The player helps a burglar steal an artifact from a museum guarded by guard agents. The burglar has incomplete knowledge of the environment and his plan will contain pitfalls. The player has to identify these pitfalls by observing burglar's behavior and change the environment so that the burglar replans and avoids the pitfalls. The game prototype is evaluated in a small-scale human-subject study, which suggests that the anticipation game concept is promising.

Introduction

Planning technologies have attracted a lot of attention in academic community and, more recently, even among game developers. So far, the research has focused mainly on adapting planning techniques to the needs of current game genres, be it at design time or during gameplay. The aim of the game industry and of many of the researchers is to replace or enrich traditional approaches of scripting and reactive planning. Besides trying to tailor planning for current games, we can attempt to create brand new types of gameplay elements that revolve around action planning and would be impossible with reactive decision making only.

Interactive Storytelling (IS) may be considered a step in this direction. In IS, planning is often used as a technology that enables creating or maintaining a coherent story plot. However, the existence of a plan remains hidden to the user. Contrary to this approach we propose a new type of games where the fact that we have a complete plan of agent's actions plays a key role. We call them *anticipation games*. Imagine you play a game where the main agent has a mission that he must accomplish. He creates a plan for this mission but due to incomplete knowledge of the environment there are some *pitfalls* in his plan. The human player has more complete knowledge of the environment, thus when observing execution of the agent's plan, she can anticipate these pitfalls (in further text, agents will be referred to in masculinum, while the player in feminimum). Once she identifies a pitfall, she can modify the environment so that the agent has to re-plan and the new plan avoids this pitfall. The player influences the agent only indirectly through changes of the environment.

Besides using planning as the key element of the gameplay, we also use it to assist the game designers to assure sufficient complexity of the level. In particular we check several properties of a game level with the same planning algorithm that was employed to plan agent's actions. Briefly speaking, the goal is to find a game level where the initial agent's plan contains a given number of pitfalls and these pitfalls can be avoided by players's actions that force the agent to re-plan.

The goal of this paper is to define formally the anticipation games, exemplify it on our implemented prototype, and describe algorithms used at design time and run time.

The rest of the paper continues as follows. In the next section we will detail previous work related to use of both planning and anticipation in games. Then, for explanatory purposes, we will describe game-play of our burglar game. Then we will define formally the class of anticipation games and describe our game from a technical perspective as an instance of the anticipation game. Finally, we will present a small evaluation of the concept of anticipation games, where we use the burglar game as an instrument.

Related Work

Considering planning in the context of computer games, a huge body of work focuses on the use of planners in design and verification of levels, e.g. (Pizzi et al. 2008; Li and Riedl 2010; Porteous and Cavazza 2009). We consider this as an off-line use of planners.

On-line use of planning during actual gameplay is severely limited by the available computing time. With respect to commercial games, the most successful algorithm is the GOAP (Orkin 2003; 2006), a simplified STRIPS (Fikes and Nilsson 1972) like planner. So far 14 game titles have used this planner[1]. Custom HTN planners have also been used in commercial games (Champandard, Verweij, and Straatman 2009). Aside from planners created specially for purposes of games, there were also attempts to use off-the-shelf PDDL compatible planners (Bartheye and Jacopin 2009) or HTN planners (Muñoz-Avila and Hoang 2006) in games directly. PDDL is a modeling language used to describe planning domains in International Planning Competition (IPC) and it became a de-facto standard input language used by many planners.

In accord with IPC challenges, benchmark tasks motivated by needs of FPS games have been proposed recently (Vassos and Papakonstantinou 2011).

The main difference of our game prototype compared to the above-mentioned systems is that 1) we use planning in *both* level design and gameplay and 2) planning is not merely a decision-making algorithm hidden from the player, it forms a key part of the gameplay.

A different body of work related to planning in games comes from the IS field. Although the game prototype we propose is not directly related to IS, it may be extended with IS techniques. Those possiblities along with relevant references will be discussed in Future Work section.

The gameplay element of anticipating autonomuous agent actions and indirectly influencing them is found in many games. In From Dust (UbiSoft 2011), one of the game elements is that the player sees the planned movement of members of his tribe and alters the landscape so that they may safely travel to their location. In Frozen Synapse (Mode 7 2011) the player has to anticipate enemy movement and create plans that are then resolved simultaneously with enemy plans. Quite a few other strategy games employ anticipation and indirect commands at various levels, for example Black And White (Lionhead Studios 2001) or The Settlers (Blue Byte Software 1996).

A very different kind of anticipation is present in The Incredible Machine (Sierra Entertainment 1992) and its sequels. The player tries to alter a given setup of devices, items and animals so that upon simulating the system according to a modified laws of physics it evolves to a specified goal condition (e. g. a ball reaches a designated place).

In all of the above mentioned games, the player anticipates either a very simple behaviour or the actions of another player - which are very hard to guess on the other hand. Incorporating planning allows the agents in our game to exhibit more complicated behaviour which might be more fun/challenging to try to foresee and thus the anticipation gameplay element can be brought to a new level and play a more central role.

Burglar Game Description

For explanatory purposes, we will detail mechanics of our game prototype first. Anticipation games in general will be defined later.

The overall situation in all game levels is that a *burglar* – a computer controlled agent – tries to steal a valuable artifact from a secured museum. Apart from the burglar, there is one more class of active agents in the level, the *guards*. The game world consists of interconnected *rooms* of different sizes and shapes. The rooms can contain one of these objects: *cameras*; *containers*, that can hold *keys*; *artifact*, that is the target for the burglar; *sleeping guards*, which can be tied down and the burglar can use their uniforms to sneak under the cameras; and finally *doors* between rooms. If the burglar or a guard has a proper key, they can both lock and unlock a door or a container.

The burglar is caught if he gets into the same room with a patrolling guard or with an active camera. In the rest of the paper such places will be called *trap rooms*. The complication is that the burglar knows only some of these dangers.

The human *player* observes the game world from a bird's eye view. The player's goal is to change small details of the environment, such as locking the doors and containers or even disabling cameras, to prevent the burglar getting caught on his mission. The player wins when the burglar runs away with the artifact, she looses if the burglar gets caught or has no valid plans to reach his goal. The levels are designed in a way that without help of the player the burglar will surely be caught.

While the player can alter state of any object in the game, each interaction costs her a price in action points expense of which she should keep minimal. The player can also spend action points to take a look at the visualized plans of any agent's future actions, the more actions the player sees the more she pays. It is important to highlight that though classical path planning is part of the problem, action planning is very important there as the burglar needs to plan actions such as opening the doors, stealing the uniform etc.

At the beginning of the game the burglar is always positioned at the entrance to the game area, and that is also the place where he has to return. Initially, the burglar knows the layout of the map, the exact location of the artifact and positions of some traps. Based on this incomplete knowledge, the burglar makes a plan how to steal the artifact and then escape from the museum. However it is guaranteed by the design process that his plan will contain pitfalls. There will always be trap rooms on his way and it is the player's task to make the burglar avoid them. For instance, there are cameras in predefined rooms. When the player finds out that the burglar will enter a room with a camera, she can lock a door on the burglar's path so that he has to change his route and misses the trap room.

The burglar and the guards have their own belief-base about the environment, which they use to plan future actions. Their knowledge may of course be wrong. The belief-base is updated whenever the agent finds an inconsistency with the world state. This may trigger re-planning and thus it may cause change of the actual plan. Through the game, it is up to the player to predict the actions of the agents. However if she decides to spend action points to take a look at the visualized plan of an agent, the view also highlights where the belief base of the agent differs from the real world state.

On Figure 1 there are two game situations. On the left image, the player locks a door on the burglar's path to prevent him encounter the guard. When the burglar discovers

[1]For the list of commercial games using GOAP see http://web.media.mit.edu/∼jorkin/goap.html, 23.11.2011

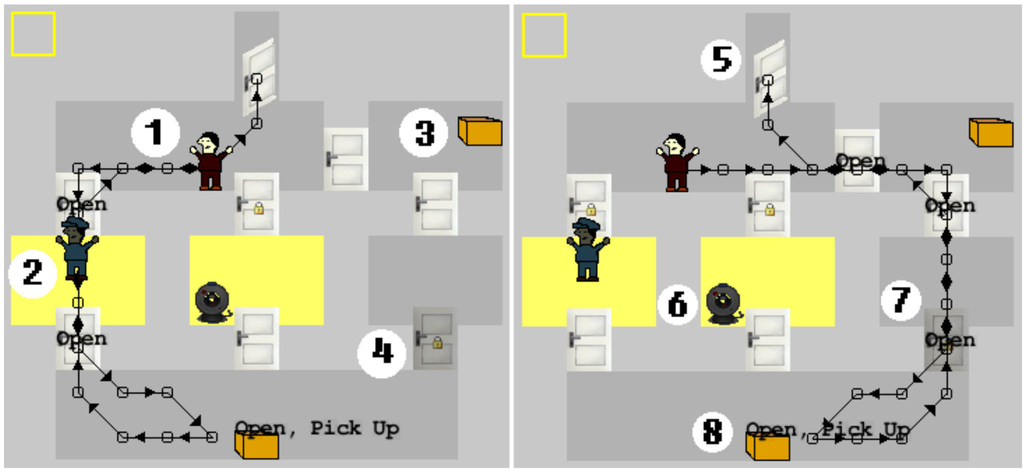

Figure 1: *Locking a door causes the choice of another path, seen from the burglar's perspective. Visible objects are: 1) the burglar, 2) a guard, 3) a closed container, 4) a closed door, marked with a darker colour to symbolize it is not fully consistent with the burglar's belief base, 5) the level entrance, 6) a camera, that is marked with deep dark colour to symbolize that it is completely unknown to the burglar, 7) an intent line showing the future path of the burglar in the game area with arrows to mark his direction, 8) a container holding the artifact, with textual description what the burglar intents to do with it.*

that the door is closed, he re-plans. The right image captures this situation, and also points out that there is another locked door on the burglar's new path, that he is not aware of yet.

The difficulty of a level is given by the number of places where the player has to assist the burglar. The harder the level, the more trap rooms are put to the world, but on the other hand there has to be some sequence of actions (including the player's actions) leading to a successful end. Hence when designing the game level, planning techniques are valuable to verify these properties.

Anticipation Game Definition

We will now abstract our game and we will formally define a class of games that we call anticipation games. We will do so by defining properties of the game level of such games.

Anticipation game is a tuple:

$$\langle S, S^0_{real}, A_{player}, Agents, mainAgent, prohibited(S)\rangle$$

where S is a set of possible world states, $S^0_{real} \in S$ denotes the real world state at the beginning of the level, A_{player} is a set of actions available to the human player, $Agents$ is a set of agents in the level, $mainAgent \in Agents$ is the agent the player should help to, the other are background agents. The predicate $prohibited(S)$ defines which states of the world contain a pitfall and thus are prohibited to the main agent. Each agent is specified by a tuple $\langle S^0_{agent}, A_{agent}, goal(S)\rangle$ and its decision making system (Alg. 1). $S^0_{agent} \in S$ is an initial world state as known by the agent, A_{agent} is a set of his possible actions and $goal(S)$ is a predicate that defines states of the world the agent is trying to achieve. An action is a partial function $a : S' \subset S \to S$. Action is applicable in every state $s \in S'$ and the corresponding function value is the new state after applying such action. In each cycle the agent executes Alg. 1, that is, he executes one action from its plan P. P is a sequence of actions $a_i \in A_{agent}$, that is $P = a_1, a_2, ...a_n$. Then he observes the new state of the world and updates his belief base according to it. In the end he decides whether he should replan or not. We will specify the particular implementation of $updateBeliefBase$ and $shouldReplan$ functions used in our game prototype later as they are not needed in the description of a general anticipation game. We should note that the $updateBeliefBase$ also models the agent's perception.

Algorithm 1 One step of an agent's decision making

```
Require: $P$ — plan that is being executed
Require: $G$ — goal pursued by the agent
Require: $S^t_{agent}$ — agent's prior knowledge of the level
1: $action \leftarrow$ getNextActionFromPlan($P$)
2: if $action \neq undefined$ then
3:     $S^{t+1}_{real} \leftarrow$ action($S^t_{real}$)
4: end if
5: $S^{t+1}_{agent} \leftarrow$ updateBeliefBase($S^{t+1}_{real}, S^t_{agent}$)
6: if $action = undefined$ or shouldReplan($S^{t+1}_{agent}, P$) then
7:     $P \leftarrow$ plan($S^{t+1}_{agent}, G$)
8: end if
```

Not all details of the real world state S^t_{real} in time t have to be perceivable by the agent (e.g. objects in different rooms). His internal believed world state S^t_{agent} does not have to correspond to the real world state.

Each level of an anticipation game has to be constructed so that there will be some pitfalls in the agent's initial plan. At the same time, the player should have the possibility to choose some actions whose outcome will make the agent to re-plan and pick a new plan. We formalize this requirement using the following formula:

$$\exists \bar{A} \subseteq A_{player} \exists t : P = plan(S^0_{mainAgent}, G) \wedge$$
$$numFlawsInPlan(P, S^0_{mainAgent}) = 0 \wedge \quad (1)$$
$$numFlawsInPlan(P, S^0_{real}) = n \wedge n \geq 1 \wedge \quad (2)$$
$$\bar{A} \oplus S^0_{real} = S^1_{real} \wedge S^0_{real} \neq S^1_{real} \wedge \quad (3)$$
$$shouldReplan(S^t_{mainAgent}, P) \wedge \quad (4)$$
$$\forall t' < t : \neg shouldReplan(S^{t'}_{mainAgent}, P) \wedge \quad (5)$$
$$P' = plan(S^t_{mainAgent}, G) \wedge \quad (6)$$
$$\neg flawed(P', S^t_{mainAgent}) \wedge \neg flawed(P', S^t_{real}) \quad (7)$$

Where $numFlawsInPlan(P, S^t) = |\{t' \in \mathbb{N}, t' \geq t :$ execution of plan P in state S^t will lead the agent into state $S^{t'}$, such that $prohibited(S^{t'})\}|$. Thus it returns the number of pitfalls in plan P when executed from the state S^t.

The formula requires that there is an initial world state S^0_{real} and its modified version known to the agent S^0_{agent} such that a plan P chosen by the agent seems to be solving the task given the agent's initial knowledge (Cond. 1) but that contains $n \geq 1$ pitfalls in reality (Cond. 2) . Moreover there must be a set of the user's actions $\bar{A}$ application of which on the initial state results in a new different state (Cond. 3), the $\oplus$ operator is used to apply effects of actions on a world state. There must also be some time t when the agent first re-plans due to inconsistency between S^t_{agent} and S^t_{real} (Cond. 4 and 5). Eventually, after re-planning at time t, the agent will create a new plan P' (Cond. 6) that is without pitfalls both in S^t_{agent} and S^t_{real} (Cond. 7). Thus P' could be followed by the agent without the player's intervention and it will result into a successful completion of the level.

Note that the main agent does not know the plans of the background agents. This definition can also lead to creating levels where the initial burglar's plan contains n pitfalls but it can be solved with just one user's action. We would like to overcome these limitations in future work, but there is a simple case, where no further requirements are needed: if the number of pitfalls, that may be resolved by a single action, is bound by a constant k, the minimal number of user actions is at least n/k.

Anticipation Game Level Design

With the anticipation game definition provided in the previous section we can create a generic level design algorithm. Suppose that a designer specifies S^0_{agent} that describes the main agent's initial knowledge of the level. In the end we want the world state with pitfalls, that is, we want S^0_{real} Alg. 2 shows a brute force solution of the game level design problem. First at Line 1 we create a plan solving the level. Then we iterate over all combinations of plan steps where we possibly could place some pitfall (Line 2), the $possiblePitfalls$ function returns such steps. Next we modify the level so that there will really be pitfalls when the agent gets to these steps of the plan. At Line 4 we tell the agent where the pitfalls are (the agent knows the real state of the world S_{real}) and ask him to make a plan avoiding these pitfalls (this requirement is contained in the goal G). If there is such a plan the last step is finding human player's actions that will force the agent to re-plan and pick the plan P', this is done by the $userReplanActions$ function. Note that $possiblePitfalls$, $placePitfalls$ and $userReplanActions$ procedures are game specific.

Algorithm 2 Anticipation game level design

```
Require: $S^0_{agent}$ — initial world state by the designer
Require: $n$ — number of required pitfalls in the level
1: $P \leftarrow$ plan($S^0_{agent}, G$)
2: for all $pitfalls \subseteq$ possiblePitfalls($P$) $\wedge |pitfalls| = n$ do
3:     $S_{real} \leftarrow$ placePitfalls($S^0_{agent}, P, pitfalls$)
4:     if $\exists P' : P' =$ plan($S_{real}, G$) and $\exists \bar{A} : \bar{A} =$ userReplanActions($S_{real}, P, P'$) then
5:         return $\langle S_{real}, \bar{A} \rangle$
6:     end if
7: end for
8: return $null$
```

Anticipation Game Instance

In the previous sections we introduced a formal definition of anticipation games. The decision-making and game-level-design algorithms were also described in an abstract way. Now we will describe our prototype game in terms of the previous definitions. A more detailed description can be found in the related thesis[2] (Toth 2012).

Burglar's and Guards' Planning. In the on-line phase the burglar and the guards can perform the following actions: *approach, open, close, lock, unlock, enter* and *operate*. All these actions are atomic and take exactly one time unit to execute. Actions related to game objects can be performed only when the agent stands right next to the object. From the planner's point of view there is no difference between room sizes, or distances between the objects.

All the agents have the same planning domain. The only real difference between the two types of agents is in their goals. The guards' goals consist of a list of rooms that they need to visit, while the burglar has a single goal room and an artifact to gather. The guards are repeating the same plan again and again - once a guard visits all the rooms he starts again - unless the change of environment, such as locked doors, forces him to re-plan. If the guard has no plan to achieve his goal, he remains still.

The $updateBeliefBase$ function from Alg. 1 is implemented in a way that the burglar gets information about presence of all objects in his current room. However he recognizes some details of the objects only when he tries to use them (e.g. he realizes that the doors are locked only when he tries to open them).

The $shouldReplan$ function returns true only if the agent comes upon an instruction in his plan he is unable to execute (e.g. he finds a locked container that was supposed to be

[2]downloadable from http://burglar-game.googlecode.com/files/thesis.pdf [23.07.2012.]

opened, without the key to open it). An alternative approach would be to re-plan each time the agent finds an inconsistency of his internal believed world state with the real world state. However this can cause much higher frequency of re-planning, thus leading to less predictable behavior. We tested this approach but finally we decided to use the first method where the agent re-plans only when the inconsistency causes a failure of his plan. Nevertheless our *posthoc* evaluation has shown that the latter approach is closer to human behavior, which opens the door for future work.

When implementing the $plan$ function that utilizes external PDDL planners we made the following observation. When the burglar knows that there is a pitfall in e.g. $Room3$, then the goal G should contain negative predicate $\neg visitedRoom(Room3)$. However such negative predicate significantly slows down all the tested planners. It is much more convenient to emulate this requirement by removing the $Room3$ from the planning domain and running the planner on this modified domain.

Level Design. In our design process, the designer specifies the map layout and possibly adds some objects to it. That becomes the main agent's prior knowledge S^0_{agent}. The map layout remains fixed, the pitfalls that are placed on the burglar's initial path by the $placePitfalls$ function are *cameras* or *guards*. The general Alg. 2 can be simplified because there are always some user actions $\bar{A}$ that are to be found by the function $userReplanActions$: each pitfall is in a room and the player can lock the doors to this room. Thus for each pitfall there is a player's action that will force the burglar to re-plan. The only open question is placing the pitfalls in such a way that there still exists a plan for the burglar. We use some domain specific information there. For example we do not try to place pitfalls in rooms that the burglar is unable to avoid, like graph chokepoints, or rooms that hold objects necessary for completion of the burglar's mission.

Note that generating plans for agents and the level building is more complex than path finding. It is not enough to find the shortest route, the acting agents may have to pick up items, use objects, lock and unlock doors; while doing this, the agents change the world state. In addition, to control whether there exists a valid solution for a level, the program has to take into account both the agents' and the player's possible actions.

Even though the algorithm still has time complexity exponentional in the number of rooms and objects, it proved to be usable on our small testing domains. Moreover it could be executed offline since it is run only at design time.

Implementation

The game has been implemented as a Java application, it uses an external game engine Slick[3] and an external planner SGPlan 5.22 (Chen, Wah, and Hsu 2006). SGPlan can be replaced with other planners capable of solving problems in PDDL 2.2. We use Planning4J[4] interface, that works with several different planners. The Navigation library is used to smooth movement of the agents. While the planner uses high-level actions such as *enter a room* or *approach an object*, the particular path between two points in the environment is planned using the classical A* algorithm. The game prototype is downloadable and open-source[5].

Domain Size and Performance

The planning domain of the game agents has 12 predicates describing properties of the environment, 10 different operators and 20 types of objects. We created a level with 28 rooms to test performance of the planners. When translated to the burglar's PDDL planning problem this level had 74 PDDL objects and 167 initial facts. The exact number of facts and objects may vary based on the actual belief base of the agent, but the one used in this example had a near flawless knowledge of the world. The resulting plan for the problem above contained 78 grounded actions. On the current test configuration (Intel Core i7 2GHz, 2GB RAM), the time required to create such a plan is about 300 ms including the initial construction of the problem description in PDDL from the internal representation in Java and parsing the returned instruction list. Several PDDL planners have been tested on this level, FF 2.3 (Hoffmann 2001), Metric-FF (Hoffmann 2003), SGPlan 5.22 and MIPS-XXL (Edelkamp and Jabbar 2008) all ended within half a second, however Blackbox (Kautz and Selman 1999), HSP (Bonnet and Geffner 1998), LPG (Gerevini, Saetti, and Serina 2006), LPRPG (Coles et al. 2008), Marvin (Coles and Smith 2007) and MaxPlan (Xing, Chen, and Zhang 2006) ended with error or reported that the problem is unsolvable.

The hardest game level so far was a 10 by 10 room maze with all neighboring rooms interconnected, the problem definition contained 290 PDDL objects and 912 initial facts. In this level only FF 2.3, SGPlan 5.22 and Metric-FF found solution in 8 seconds limit. MIPS-XXL needed nearly 50 seconds and the other planers failed.

Evaluation

By experimental evaluation we wanted to obtain the following information concerning the gameplay of an anticipation game. First, whether the players can play the game in a way we expected (Task 1). Second, what strategy would humans use for re-planning (Task 2). This was done with a study of human players trying our game prototype. The evaluation was performed with 20 college students, 13 of them studied computer science. There were 16 men and 4 women between 20 and 32 years old.

In Task 1 we had four questions, we tested if the players can: Q1) predict burglar's path, Q2) identify the pitfalls, Q3) pick the action that will make the burglar re-plan and avoid the pitfall and Q4) predict its new plan. In Task 2 we modified the game so that the players controlled directly the burglar and they had access only to the information the burglar has. We wanted to know if they would re-plan when an action from the initial plan fails (Strategy 1), or when a new

[3]Slick homepage: http://slick.cokeandcode.com [18.5.2012]

[4]Planning4J homepage: http://code.google.com/p/planning4j/ [17.5.2012]

[5]Homepage of the game is http://burglar-game.googlecode.com [23.07.2012]

shorter plan emerges because of a new knowledge (S2); or if they will explore the environment after noticing the change but before re-planning (S3).

Method. In the first part of the evaluation the participants were introduced to the key concepts of the game by playing 3 tutorial levels[6] that demonstrated all the concepts needed to solve the test levels. If they made some mistake they were allowed to play the level again until they solved it.

In Task 1 they were presented with three previously unseen levels A, B and C printed on a paper[7] showing the same information as on a computer screen. Then they were asked to draw how they would solve the level with minimum penalty points. After drawing the solution they could run the simulation on a computer. Then they were asked to rate its difficulty using a five point Likert item ranging from 1-*easy* to 5-*difficult*. After completing levels A, B and C they rated overall enjoyment of the game. Answers to Q1-4 were obtained by measuring percentage
[8]

of participants that had correctly drawn the plans (Q1 and Q4) and marked pitfalls (Q2) and objects whose state has to be changed (Q3) (Fig.).

In Task 2 we let the participants directly control the burglar in two different levels to find out how they would behave if they were the burglar. Both levels were designed in a way that there were some initially unknown objects, doors or even whole rooms that make it possible to create a new shorter plan if they were perceived by the human player (corresponds to S2). In the second level there were also many possibilities for exploratory behavior not following any plan (S3). Fig. shows these two levels.

Participants were allowed to ask the experimenter questions about mechanics of the game both during the tutorials and in the testing phase. Each participant had about one hour to complete the whole procedure.

Results. Results of Task 1 are summarized in the Table .

Level	Q1	Q2	Q3	Q4	Avg. dif. rating
A	100%	95%	85%	90%	3
B	85%	95%	75%	80%	4
C	83%	72%	89%	78%	2.6

Table 1: *Percents of players that were successful in tasks designed to answer Q1-4 and avg. rating of difficulty. Level C was played by only 18 participants.*

In Task 2 we found that in the first level (Fig. left) all participants re-planned as soon as they realized that there is a shorter plan (when they enter the room with a sleeping guard

[6]Levels 1, 2 and 4 as can be found in the downloadable distribution of the game.

[7]The levels can be obtained on http://burglar-game.googlecode.com/files/AIIDE-12_test_levels.zip [23.07.2012.]

[8]We compared the plan drawn by a participant with the actual plan chosen by the burglar. If the plan diverged in an insignificant detail (e. g. a different, but still minimal path through a set of empty rooms), we treated the plan as guessed.

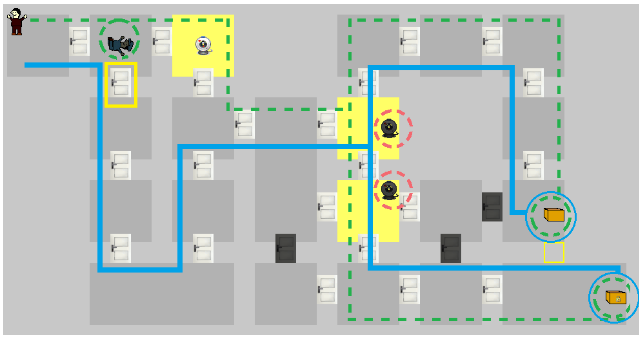

Figure 2: *One of the evaluation levels with the solution drawn by one of the participants. Blue solid line shows the burglar's initial plan as drawn by the participant (Q1). Red dashed circles mark pitfalls on the selected path (Q2). Yellow square marks door that should be closed (Q3). Green dashed line shows the final path (Q4) where the burglar disguises himself as a guard, taking advantage of the previously unknown sleeping guard.*

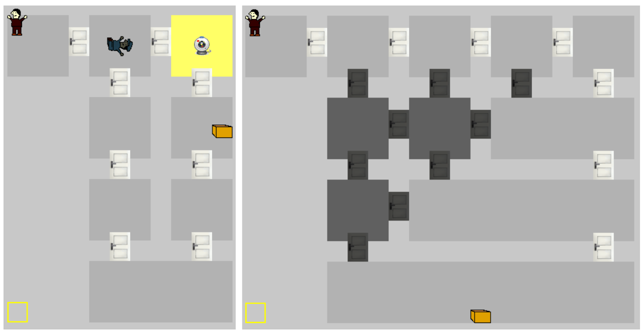

Figure 3: *Two situations where human participants directly controlled the burglar. Dark objects, doors and rooms were initially unknown to the player.*

they can take his uniform and sneak under the camera without avoiding this room), thus following S2. In the second level (Fig. right) 12 players decided to explore previously unknown rooms that could even contain potential pitfalls, this corresponds to S3. The other 8 followed the plan based on the initial knowledge of the level. They re-planned only when they were sure that the new path will be safe given their updated belief base — S2.

Discussion. We see that even after a brief period of training the participants were able to play the game quite well. They were able to guess the burglar's plan, identify the pitfalls, fix them and predict the new plan (see Table). Thus the concept of the game seems to be viable. It is also positive that most players rated the game as entertaining when the average rating was 0.5 on the scale -2 ... +2.

The experiments where the burglar was controlled by humans show us that humans follow strategy S2 and in some situations S3, none of the participants used S1 implemented in our *shouldReplan* procedure. This posses a question whether the burglar should not re-plan as a human. Future research is needed to investigate this question.

Future Work

As long as the whole game concept is based on players' ability to predict the burglar's plans, we need the planner not only to produce *some* plans but arguably to produce plans that resemble plans of humans. Current off-the-shelf planners are not optimized for this type of objective, but we can get inspiration from the IS field. For instance there are works that try to extend planning algorithms to account for intentions of agents (Riedl and Young 2010), suspense (Cheong and Young 2008) or emotions (Aylett, Dias, and Paiva 2006). We think that these properties can make the plans more engaging for humans. We can also focus on the re-planning strategy and change it as suggested by our evaluation. Or we can extend the definition to require existence of harmful player's actions that lead to capturing the burglar.

Conclusion

In this paper we defined formally a game genre of anticipation games, which advocates a novel form of exploiting action planning in games: both at design time and run time.We also presented a game prototype and a small-scale evaluation of the anticipation game concept. The game concept can be useful for academy as a research platform and although the game levels can only be of medium complexity given state of the art planners, it can be useful also for industry, e.g. for creating specific game missions.

Acknowledgment

This work was partially supported by the student research grant GA UK 655012/2012/A-INF/MFF, by the SVV project number 265 314 and by the grants GACR 201/09/H057 and P103/10/1287.

References

Aylett, R.; Dias, J.; and Paiva, A. 2006. An affectively-driven planner for synthetic characters. In *Proceedings of ICAPS 2006*, 2–10.

Bartheye, O., and Jacopin, E. 2009. A real-time PDDL-based planning component for video games. In *Proceedings of AIIDE 2009*, 130–135.

Blue Byte Software. 1996. The Settlers II.

Bonnet, B., and Geffner, H. 1998. HSP: Heuristic search planner. *AIPS-98 Planning Competition*.

Champandard, A.; Verweij, T.; and Straatman, R. 2009. Killzone 2 multiplayer bots. In *Game AI Conference 2009*.

Chen, Y.; Wah, B. W.; and Hsu, C. 2006. Temporal planning using subgoal partitioning and resolution in SGPlan. *Articial Intelligence* 26:323–369.

Cheong, Y., and Young, R. 2008. Narrative generation for suspense: Modeling and evaluation. In *Proceedings of ICIDS 2008*, 144–155. Springer.

Coles, A., and Smith, A. 2007. Marvin: A heuristic search planner with online macro-action learning. *Articial Intelligence Research* 28:119–156.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A hybrid relaxed planning graph-LP heuristic for numeric planning domains. In *Proceedings of ICAPS 2008*, 52–59.

Edelkamp, S., and Jabbar, S. 2008. MIPS-XXL: Featuring external shortest path search for sequential optimal plans and external branch-and-bound for optimal net benefit. In *6th. Int. Planning Competition Booklet (ICAPS-08)*.

Fikes, R., and Nilsson, N. 1972. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4):189–208.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *Artificial Intelligence Research* 25:187–231.

Hoffmann, J. 2001. FF: The fast-forward planning system. *Articial Intelligence* 22:57–62.

Hoffmann, J. 2003. The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Articial Intelligence* 20:291–341.

Kautz, H., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *Proceedings of IJCAI 1999*, 318–327.

Li, B., and Riedl, M. 2010. An offline planning approach to game plotline adaptation. In *Proceedings of AIIDE 2010*, 45–50.

Lionhead Studios. 2001. Black & White.

Mode 7. 2011. Frozen Synapse.

Muñoz-Avila, H., and Hoang, H. 2006. Coordinating teams of bots with hierarchical task network planning. *AI Game Programing Wisdom* 3.

Orkin, J. 2003. Applying goal-oriented action planning to games. *AI Game Programming Wisdom* 2(1):217–227.

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

Pizzi, D.; Cavazza, M.; Whittaker, A.; and Lugrin, J. 2008. Automatic generation of game level solutions as storyboards. In *Proceedings of AIIDE 2008*, 96–101.

Porteous, J., and Cavazza, M. 2009. Controlling narrative generation with planning trajectories: the role of constraints. In *Proceedings of Interactive Storytelling*, 234–245. Springer.

Riedl, M., and Young, R. 2010. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):217–268.

Sierra Entertainment. 1992. The Incredible Machine.

Toth, C. 2012. Planning systems in game level design and agent control. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics.

UbiSoft. 2011. From Dust.

Vassos, S., and Papakonstantinou, M. 2011. The SimpleFPS planning domain: A PDDL benchmark for proactive NPCs. In *Workshops at the AIIDE 2011*, 92–97.

Xing, Z.; Chen, Y.; and Zhang, W. 2006. Maxplan: Optimal planning by decomposed satisfiability and backward reduction. In *Proceedings of ICAPS 2006*, 53–56.