

Comparing Approaches to Game AI under Consideration of Gameplay Mechanics

Research project by Thomas Haase (732830)

Summer Semester 2018

Hochschule Darmstadt
Digital Media – Animation and Game

Erstprüfer
Prof. Dr. Martin Leissler

Zweitprüfer
Thomas Nickel

Table of contents

0.1. Abstract (English).....	3
0.2. Abstract (German)	3
1. What is Game AI supposed to do	4
2. Which mechanic suggests which AI structure	6
3. State of the Art	7
4. Related Work	8
5. Comparing Approaches to Game AI under Consideration of Gameplay Mechanics	9
5.1. What is Splinter Cell: Blacklist	9
5.2. What is Mark of the Ninja	9
5.3. Applying the game design patterns.....	10
5.4. What AI Systems are there?	17
5.4.1. General Structure	17
5.4.2. Decision Tree	18
5.4.3. Finite State Machine	19
5.4.4. Behavior Tree	21
5.4.5. Goal-oriented Behavior	23
5.5. Applying AI Structure to Gameplay Mechanics.....	24
6. Evaluation	26
7. Future Outlook	27
8. List of references	28
9. List of figures	29

0.1. Abstract (English)

Often it is not clear to beginner and intermediate AI developer [Artificial Intelligence] what AI architecture to pick for their game. To find indicators this research project compares approaches to Game AI under consideration of gameplay mechanics, specifically stealth games. It has a look at 21 game design patterns, as described by Mengchen Hu in his Master Thesis *Game Design Patterns for Designing Stealth Computer Games*¹, as well as four in the process newly created ones, while considering impact on the AI Structure. It continues to summarize four of the most common AI structures, as described in *Artificial Intelligence for Games*² by Ian Millington, namely Decision Trees, State Machines, Behavior Trees and Goal-oriented Behavior. In an attempt to connect both summaries, four indicators can be found that indicate a relationship between gameplay mechanics and AI structure. The findings are then applied to two examples, *Splinter Cell Blacklist* [2013, Ubisoft] and *Mark of the Ninja* [2012, Klei Entertainment], and suggests State Machines as the AI structure of choice. It is not able to create hard and fast rules to select an AI structure for any given game.

0.2. Abstract (German)

Oftmals ist es für Anfänger und Fortgeschrittene im Bereich KI [Künstliche Intelligenz] für Spiele nicht ersichtlich, welche KI Struktur für ein etwaiges Spiel angebracht ist. Um Anzeichen zu erhalten vergleicht diese Arbeit Ansätze von KI für Spiele unter Berücksichtigung der Spielmechaniken, genauer Stealth Spiele. Dafür werden 21 Game Design Patterns unter Berücksichtigung der KI betrachtet, die Mengchen Hu in seiner Master Thesis *Game Design Patterns for Designing Stealth Computer Games* beschreibt, und erstellt im Zuge dessen vier weitere. Daraufhin werden vier der häufigst genutzten KI Strukturen betrachtet, nämlich Decision Trees, State Machines, Behavior Trees und Goal-oriented Behavior. Im Versuch beide Zusammenfassungen zu verbinden werden vier Anzeichen herausgearbeitet, die einen Zusammenhang zwischen den Spielmechaniken und der KI Strukturen erkennen lassen. Diese Zusammenhänge werden dann an zwei Beispielen erprobt, *Splinter Cell Blacklist* [2013, Ubisoft] und *Mark of the Ninja* [2012, Klei Entertainment]. Anschließend wird Vermutung angestellt, bei beiden Beispielen seien State Machines die KI Struktur der Wahl. Es kann keine sichere Regelung zur Verwendung einer KI Struktur für jedes Spiel gefunden werden.

¹ Cf. Hu, Mengchen. *Game Design Patterns for Designing Stealth Computer Games*. Malmö Högskola, Faculty of Technology and Society, 2014

² Cf. Millington, Ian. *Artificial Intelligence for Games*. 2nd ed., Morgan Kaufmann Publishers, 2009

1. What is Game AI supposed to do

In today's pop culture there are a plethora of divinations about humanity's brutal ending by the hands of their own artificially crafted intelligences. The bringers of doom range from gun wielding robotics, like *The Terminator* [1984, James Cameron], to overly protective administrators, as in *I, Robot* [2004, Alex Proyas].

Headlines like "*AlphaZero: DeepMind's AI becomes unbeatable in chess*"³ [AI – Artificial Intelligence] further fan the flames of fear about this seemingly magical, impossible to grasp concept of artificial intelligence. However, *AlphaZero* is not perfect, or even intelligent.

Machine Learning is powerful. It seems to be the most capable set of algorithms there is. So why should not everyone use Machine Learning for any Game AI?

To take this into the absurd: What if we took *Inky*, *Blinky*, *Pinky* and *Clyde* and gave them the ability to learn from their mistakes? Thinking of a perfectly aligned special force hunting down *Pac-Man* [1980, Namco], while this would be glee for any cruel and/or evil designer or AI programmer, it also would be the core of frustration for anyone trying to enjoy this classic game of the pebble eating hockey puck. Obviously that is not what Game AI is supposed to do. But then, what is Game AI supposed to do?

First off, there is a problem in defining Game AI. In academia there are three different approaches to AI itself. You can look at AI from a philosophical standpoint, asking questions like "What is consciousness?". In psychology AI is used to propose, test and (dis-)prove how the human mind actually works. In engineering algorithms are utilized to get "the job done".⁴ Game AI comes closest to the latter. Processing time and memory are vital for any smoothly running game. And as game programming is riddled with hacks and heuristics to get smaller and bigger benefits towards performance, so is Game AI.

This opens up to the problem of defining Game AI: When a hack is used to simulate intelligence, even as simple as an if-then-else-statement, can it even be considered as intelligence? With this premise, looking at two enemy-types in *Super Mario World* [1990, Shigeru Miyamoto], *Goomba* and *Koopa*, there already can be made a distinction. While a *Goomba* runs off any cliff it comes across, showing no more decision making than a rock "deciding to fall towards the earth", a *Koopa* will turn around as soon there is no ground to

³Abix. "AlphaZero: DeepMind's AI becomes unbeatable in chess" *Steemit*, <https://steemit.com/technology/@abix/alphazero-deepmind-s-ai-becomes-unbeatable-in-chess>
Accessed 27 May 2018

⁴ Cf. Millington, Ian. p.4

walk upon. Does this mean a *Koopa* is more intelligent as the *Goomba*? Is the *Koopa* even considered intelligent?⁵

Although these inquiries might still be open to philosophical debate, we do not actually have to bother with them. The principal question for a practical approach to Game AI is "What is Game AI supposed to do?"

Even though it is the principal question, it is a rather broad one. Each answer might differ from another, given the context of the game. A visitor in a theme park simulator is (hopefully) meant to do entirely different things than a chainsaw-wielding ogre in a shooter game. To get probable, usable answers we have to narrow the question down, like so:

"What should a decision making process be able to decide within a game with given gameplay mechanics and in what technical way?"

Notice, that Game AI in general encompasses movement, pathfinding, as well as decision making. For this paper, though, decision making is the most interesting field in terms of observable behavior.

There are several technical approaches to Game AI to be considered when designing and producing a game. Choices in the design may limit the possibilities in choice for the underlying code structure for the AI, and vice versa. Each design pattern for Game AI performs well in a different aspect. Performance, scalability, flexibility, simplicity, authorial control and some specific use cases are to be considered when picking a system design pattern.

It should not only match the game's mechanics, but also its budget and the production team's know-how. Acquiring new skills along the way, for a game developer, is fun. But it takes time and resources, which might be crucial to a successful game. Similarly, setting up and maintaining a complicated, performant algorithm is time consuming. As Occam's Razor states:

*"Entia non sunt multiplicanda praeter necessitatem."*⁶

Or in other words, the optimal solution is not the strongest one, but the most efficient one.

When Dave Mark is asked for which AI architecture should be used, he gives this answer:

*"It depends."*⁷

At first glance this answer is confusing to the naive developer. But the certain uncertainty in

⁵ Thompson, Tommy. "AI 101 – Part 4: 'Game' AI – A unique challenge". *AI and Games*. <https://aiandgames.com/ai-101-part-4/> Accessed 7 Jun. 2018

⁶ Mark, Dave. *Behavioral Mathematics for Game AI.*, Charles River Media, 2009, p.56

⁷ Mark, Dave. "AI Architectures: A Culinary Guide". *Intrinsic Algorithm*, 1 Nov. 2012 <http://intrinsicalgorithm.com/IAonAI/2012/11/ai-architectures-a-culinary-guide-gdmag-article/> Accessed 7 Jun. 2018

this statement stands to reason, when one considers the vast number of game mechanics and possibilities to combine them. Even within a given genre, a single game mechanic can not only drastically shift the game's experience, but also the need for a specific AI architecture. So, what does AI structure depend on? It seems to be vital to compare approaches to Game AI under the consideration of game mechanics.

2. Which mechanic suggests which AI structure

It would be nice to have an indicator, more than a rule of thumb, to shove beginners in the field of Game AI in the right direction, or to re-orient intermediates. As fads come and go, there is great temptation to just pick the most popular model at the time. But it might be ill-fitted for the needs of a particular soon-to-be game. Creating a particular behavior could be infinitely harder using one architecture, when another would be more suited for the job.

To do this, this paper looks for connections between gameplay of a specific genre and its AI structure. The optimal outcome would mean these connections to be re-usable for any other game. The genre chosen for this endeavor is commonly known as stealth games. Concerning AI, it is very suitable for consideration, as the characters' behavior and the player have a special relationship:

The player will typically observe the AI enemies, make a plan to overcome them, and then execute that plan to complete any given level. The AI character in turn will try to detect the player, obstructing his way to victory. Although the game holds all the information, it is fundamental, what information is passed to the non-omniscient AI characters. Not only is their perception of the player crucial. Their behavior has to be interesting to watch, as the player will look at them for a longer time on average than in other action games. Also, the player has to be able to anticipate the AI characters' behavior, so he can make a dependable plan.

To find any connection, this paper will describe game mechanics in stealth games. This will be done with game design patterns comprised in a master thesis by Mengchen Hu.⁸ The usage of game design patterns is debatable for using them in context of designing games. But for the sake of re-usability it fits the needs of analyzing games to create common nomenclature. Similarly to Hu's endeavor concerning the player's experience and level design, the game design patterns will be revised for potential involvement of Game AI.

⁸ Cf. Hu, Mengchen. pp.36-61

It is necessary to have a repertoire of AI structures to be able to compare them to each other. That is why most common structures in Game AI are comprised and analyzed for general purpose, advantages and limitations.

This paper will then look at two example stealth games, *Splinter Cell Blacklist*⁹ and *Mark of the Ninja*¹⁰, comparing approaches to stealth Game AI under consideration of stealth gameplay mechanics.

3. State of the Art

Beginners can be overwhelmed by the choice of AI structures. Or worse, they might not know of any structure, so they use ad hoc rules for simple behavior. This might be enough for hobbyists, but can be excruciating for an earnest attempt on sophisticated behavior. The other way around can be equally misplaced, when a fashionable, but ill-fitted AI structure is used.

There are quite some sources to draw from, when entering the field of Game AI programming. “The AI Programmer’s Bookshelf”¹¹ is a good place to start a beginner’s journey to this discipline of programming. Next to the analysis of the algorithms, there is a lot of information to be collected from post mortems and anecdotal reports of AI hacks, tricks and work ethics to be found all over the internet, or comprised into books.

Experienced Game AI developers tackle the problem on a case-by-case basis. In discussions about any given problem, there often is no consensus on which AI does the job best. Instead reasonable points can be made for some structures with annotations of drawbacks.

Successful developers already do the right job and have no need for an abundant list of gameplay mechanics, game AI structures, or their correlation. Dedicated AI developers grow with experience at the hand of their productions, or by those of others in the form of post mortems. That knowledge is shared annually at the Game Developers Conference AI Summit, where interested developers can “Join top game AI programmers for panels and lectures, in addition to conversations, debates and rants on how game AI can move forward.”¹² Also, GDC Vault offers “in-depth design, technical and inspirational talks and slides from the influencers of the game development industry, taken from over 20 years of worldwide Game Developers Conferences.”¹³

⁹ *Splinter Cell Blacklist*. Ubisoft, <https://www.ubisoft.com/de-de/game/splinter-cell-blacklist> Accessed 7 Jun 2018

¹⁰ *Mark of the Ninja*. Klei Entertainment, <https://www.klei.com/games/mark-ninja> Accessed 7 Jun.2018

¹¹ Orkin, Jeff. *The AI Programmer’s Bookshelf*. MIT Media Lab, <http://alumni.media.mit.edu/~jorkin/aibooks.html> Accessed 7 Jun. 2018

¹² *Game AI Summit*. <http://www.gdconf.com/conference/ai.html> Accessed 7 Jun. 2018

¹³ *GDC Vault*. Game Developer Conference, <http://gdcvault.com/> Accessed 7 Jun. 2018

4. Related Work

In his Master Thesis of spring 2014, Mengchen Hu created a list of gameplay mechanics, that are found in stealth games, or that are specific to the said games.¹⁴ He uses a template by Hullet and Whitehead¹⁵ to create game design patterns. His angle to the problem is finding mechanics, which affect the game's experience as well as level design in stealth games. He admits that his collection is incomplete.¹⁶ Hu recognizes that AI takes part in the game's experience and assigns some of his newfound game design patterns in a sub-category for "*Creatures and behavior/AI*".¹⁷ But he does not further follow up as to why or how it is crucial to the Game AI and its development. His concluded work enumerates 21 game design patterns, which are to be examined under prospect of Game AI. Most of these patterns have attributes, that should be considered for Game AI. In what manner they affect the AI will be established in detail with concrete examples later in chapter 5.3.

Artificial Intelligence for Games by Ian Millington and John Funge comprises a collection of commonly used AI architectures. Next to describing the algorithm, it provides insight with examples, pseudo-code, and optional extensions.¹⁸ Furthermore they explain their own process when designing the AI architecture for a game. They also give a broad genre-by-genre inspection, substantiated by their own experience.¹⁹

Dave Mark provides *A Culinary Guide*²⁰ in order to simplify a suggestion for AI structures, in metaphor to Mexican food, as it is "*the same stuff...just in different shapes*".

The Game AI Pro-series²¹, edited by Steve Rabin, gives some insight on the more intricate, specific tasks of AI developers, namely perception models. Especially the articles *How to Catch a Ninja – NPC Awareness in a 2D Platformer*²² by Brook Miles and *Modeling Perception and Awareness in Tom Clancy's Splinter Cell Blacklist*²³ by Martin Walsh, give insight to what is considered important in stealth games. Instead of picking an AI structure, they show what information is fed to the same.

¹⁴ Hu, Mengchen. p.18

¹⁵ Hu, Mengchen. pp.36-61

¹⁶ Hu, Mengchen. p.62

¹⁷ Hu, Mengchen. p.71

¹⁸ Millington, Ian. p.293

¹⁹ Millington, Ian. p.807

²⁰ Mark, Dave. *AI Architectures: A Culinary Guide*

²¹ Rabin, Steve. *Game AI Pro*. <http://www.gameai.pro/> Accessed 7 Jun. 2018

²² Miles, Brook. "How to Catch a Ninja: NPC Awareness in a 2D Stealth Platformer". *Game AI Pro*, edited by Steve Rabin, vol. 1, Sept. 2013, p.413

²³ Walsh, Martin. "Modeling Perception and Awareness in Tom Clancy's Splinter Cell Blacklist". *Game AI Pro*, edited by Steve Rabin, vol. 2, Apr. 2015, p.313

5. Comparing Approaches to Game AI under Consideration of Gameplay Mechanics

5.1. What is Splinter Cell: Blacklist

In *Splinter Cell Blacklist* the main character, *Sam Fisher*, is a super special agent fighting a group of terrorists, with various tools for operating silently as well as aggressively in the open. It was developed by *Ubisoft Toronto* and released by publisher *Ubisoft* in August 2013.²⁴ The player can chose between three paths of engaging the mission, when choosing his loadout at the beginning of the game: The *Ghost* players want to remain undetected, *Assault* players are armored and will engage conflict openly, while *Panther* players will combine the two options trying to kill in a silent way. The player can upgrade his equipment with currency achieved by high mission scores.

The iconic colored lights on *Sam Fisher's* night vision goggles and other gear will tell the player whether he is hidden in shadow. Guards patrol the perimeter, talk to each other, or pretend to be lifelike in other ways, interacting with their surroundings. They react to unusual noise or circumstances, will seek out the player once detected and plan group attacks.

5.2. What is Mark of the Ninja

Mark of the Ninja is a 2D sidescrolling platformer game with various mechanics that make it to be considered a stealth game. It was developed by *Klei Entertainment*, published by *Microsoft Studios* and released in September 2012 after being in development for 16 months.²⁵ In the game the “[...] player, as the *Ninja*, sneaks through levels keeping to the shadows, crawling through vents, and ambushing unsuspecting guards.”²⁶ The player can hide in shadows or behind dedicated objects, kill guards silently and hide their bodies, or avoid them altogether. The *Ninja* is comparably weak when fighting more than one guard at the same time. The modern guards attack a detected *Ninja* with rifles until they lose track of him. After a short time they resume to their original behavior. The player can chose *Ninja's* abilities to vary in his style of play.

²⁴ “Splinter Cell Blacklist”. *Steam Store Page*,

https://store.steampowered.com/app/235600/Tom_Clancys_Splinter_Cell_Blacklist/ Accessed 9 Jun. 2018

²⁵ Cheng, Jamie. “Classic Postmortem: Klei Entertainment's Mark of the Ninja”. *Gamasutra*, 7 Sept. 2017, https://www.gamasutra.com/view/news/305166/Classic_Postmortem_Klei_Entertainments_Mark_of_the_Ninja.php Accessed 9 Jun. 2018

²⁶ Brook, Miles. p.413

5.3. Applying the game design patterns

The following listing contains the original 21 patterns²⁷ with only a short description that will suffice for now. To prevent confusion, names are kept as in the original, even when some other naming seems more appropriate. New patterns are to be created, when appropriate, under the same four categories²⁸ as in the original master thesis: Name, Description, Consequence, Using the Pattern.

Ability and Equipment	The player gets tools to overcome obstacles in a multitude of ways. Depending on the setting, these include Night Vision, Weapons, Teleportation or Invisibility, either by mundane or magical means.
Affecting AI	AI characters can react to varying loadouts with different behavior, e.g. take cover against heavy weaponry or charge against light ones.

Achievement	The player can get optional rewards for doing a task, a level or the game in a specified way. This adds to the replayability of the game, aswell as suggestion on alternative ways to play.
Affecting AI	AI characters that are involved in an achievement, must be predictable to some degree that makes planning for the achievement possible.

Action Posture	The player avatar indicates and translates to the player by animation, what the avatar is doing at the moment. E.g. a crouched avatar placing his feet carefully indicates sneaking. Associated with the Posture are positive effects (harder to detect) and negative effects (slower).
Affecting AI	Actions of AI characters must be readable to show the player what they are doing, so he can plan accordingly. A guard should show whether he is panicked and startled, or determined and aggressive. Also it can be fun to watch while the player waits for an event.

²⁷ Hu, Mengchen. pp.36-61

²⁸ Hu, Mengchen. p.35

Alert	An alarm is triggered, when an enemy detects the player avatar. This can either result in the player losing the game directly, or in enemies acting aggressively against the player until his demise or escape. The alarm state resets after certain conditions are met.
Affecting AI	The AI structure must be able to interrupt its current behavior to react to an Alert. E.g. State Machines will show their capability, while Behavior Trees have certain limitations and workarounds.

Background Story	The narrative of the game world and the role of the player in it sets a baseline in the player's experience. Architecture and decorations should match that narrative.
Affecting AI	The AI character's behavior must fit the narrative, reacting to its role inside the game world. When behavior of character types differ, it is sensible to use data-driven structures, like Behavior Trees, to increment production efficiency.

Collection Factor	Collectables in the world encourage the player to look out for them and give him additional short-term goals. The player can either use these collectables in conjunction of his abilities and equipment, like ammunition or health packs, deepening the narrative, like voice logs, or for the sake of achievements.
Affecting AI	AI characters could be able to notice objects that are suddenly amiss. They would then need to keep track of their surrounding items in detail.

Difficulty Selection	Players of different levels of skill have different needs and expectations in the game. The choice in difficulty is made either at start of the game or during playtime. Difficulty levels may change the number and behavior of enemies, weapon damage and tools for the player.
Affecting AI	Should the number in enemies change, the AI structure should be able to handle varying sizes of squads, when group behavior is necessary. The calculation of detection, e.g. vision cones, can change aswell.

Enemy Type	Having more than one kind of enemy with varied abilities calls for varying solutions. Some may be heavily armored, others could see further or in the dark.
Affecting AI	Variations in enemy types call for differing behavior. Depending on the number of different behaviors needed, it could again be sensible to use a data-driven system to allow designers to set up behaviors.

Field of Vision	Indicators, often seen as a cone-shape in the minimap, show the perception range of an enemy. The player can use this to avoid being detected. The size of the perception can depend on the character, their equipment, the game environment (rain) and difficulty settings.
Affecting AI	The Field of Vision can either be binary, or build up to complete detection in an analog way, simulating reaction time. In a simpler production a cone-shaped vision will suffice, but sophisticated production use a multitude of volumes to detect the player sensibly.

Hostage	Some mission objective might be to hijack or save a hostage. When the character has to carry the hostage, it often disables other actions and options of the player, like sneaking or jumping. A conscious hostage, however, needs some behavior to act accordingly.
Affecting AI	A rescued, AI controlled hostage has to sensibly avoid enemy detection, so not to frustrate the player when a plan fails due to its behavior.

Light and Shadow	To avoid detection the player can use dark areas to conceal his avatar. He might also be able to change the game environment, e.g. by shooting light sources.
Affecting AI	The shadow system can either be binary or vary in degrees. While raycasts could report peception, a concealed player avatar should still be ignored by the AI.

Mark	The player can mark enemies or locations to track their whereabouts. It can greatly help the player orienting, but reduces difficulty and the player's sense of reality in the game.
Affecting AI	Does not affect the AI.

Neither Damage nor Moving Action	Incapacitating enemies removes them as an obstacle to an arranged plan. Either by means of sleeping darts or strangling, the enemy will neither move, nor will he inflict damage. They might return to consciousness after a given time.
Affecting AI	After regaining consciousness the AI character could recognize and be alarmed about its earlier timeout and therefore react accordingly.

Observation	Prior to making any plan, the player must observe the level's environment and obstacles within. This includes looking for patrol routes, radio checks, or eavesdropping on other critical information. Usually the player has enough time to study, what enemies are doing.
Affecting AI	AI characters should be interesting to watch and highly readable.

Obstacles	Furniture, boxes, walls, trees and other appropriately sized inventory can be used by the player to hide his avatar or incapacitated enemies. Sometimes these obstacles can be moved to a degree.
Affecting AI	Objects breaking the line of sight from the AI character to the player should be considered when determining the detection of the player.

Option	Some choices, e.g. whether to kill or let live a perpetrator, can differ the plot of a story to varying degrees. Not only a singular event, but also repeated ways of the player acting, can have short-, mid-, or long-term consequences on the playthrough.
Affecting AI	Recent player activity and choices in the plotline, whether aggressive or peaceful, could change the AI character's opinion of the player. They should react in a suitable manner, e.g. scared, triumphant.

Patrol	Enemies may walk predefined, or otherwise obvious, routes. It limits the player's space to move in, but gives him the opportunity to use gaps for his plans or ambush an unsuspecting enemy.
Affecting AI	Pathfinding and waypoint systems are a core aspect of Game AI. The number of AI characters is possibly limited by the performance of a chosen algorithm, or vice versa.

Sound	Another way for enemies to detect the player are unusual sound sources, like suspicious footsteps, shattering glass or gasping hostages. The player can either avoid making sounds or use them to goad enemies.
Affecting AI	Varying sound sources can have different levels of importance. An AI character should interrupt its current investigations when a more important sound is perceived. The sound model should consider level geometry for the traversal of sound.

Statistics and Assessment	After a level, mission or stage, a score screen can show statistics to the player. Completion time, accuracy ratings and kill count informs the player of his performance to a varying degree of his satisfaction.
Affecting AI	According to the player's performance, Difficulty Settings can be adjusted automatically, potentially changing the AI characters behavior.

Trap	These inanimate objects react to the avatar, depending on their shape and intention. They can hurt the avatar, hamper his progress, or act as another way to detect the player by raising attention to his location
Affecting AI	The AI characters should avoid traps, which they are aware of. They could also react to recently disabled traps.

View of the Player	The way a player conceives the game can be through the avatar's eye, over his shoulder, bird's eye view from the top or the side. Each of them induce a different experience.
Affecting AI:	Limitations of the player's perception can hide enemies from him. To avoid frustration, unnoticed AI characters can have their perception constrained.

Additionally there are some aspects that to be found, which are essential concerning AI:

Pattern name: Unreachable Locations to AI

Description: Stealth games often utilize elevation. Whether it's climbing a rooftop or crawling through ventilation shafts, the AI must react accordingly.

Consequence: Unreachable Locations to the AI create space that is owned by the player, to give him moments of respite. Also, they open vertical paths for the player to plan attacks and avoidance in an extra dimension. It is not obvious how the AI should react.

Using the pattern: In *Splinter Cell Blacklist* some terrain is unreachable to the AI. Nonetheless, they should not idly wait for the player to slit their throats or conveniently snipe them from afar, turning the game into a metaphorical duck hunt. In *Splinter Cell Blacklist* enemies will try to coordinate around the obstacle, find cover and lay down suppression fire, then flush out the player using grenades.

As *Mark of the Ninja* is a vertically 2-dimensional platformer, elevation is an integral part of the game's level design. Jumping, climbing and crawling are the tools of the character to avoid detection in the limited space.

Pattern name: Disguise

Description: The player avatar can be hidden in plain sight, e.g. by dressing in uniform to the enemy characters, by covering it in shrubbery or by hiding in boxes.

Consequence: Even though the AI characters can detect the player, they react differently. They can ignore him outright, or can break the disguise by further investigation.

Using the pattern: Neither *Splinter Cell Blacklist* nor *Mark of the Ninja* utilize this mechanic. Honorable mentions here are the *Hitman*-series [2000, IO Interactive]. *Hitman* is well-known for the main character to be disguised as the recently clobbered victims. The disguise breaks, when the player avatar does something atypical from the behavior of the impersonated character.

Pattern name: Goad

Description: The player can lure AI enemies away or towards specific spots, either as a diversion or to set up an ambush. He can do this with tools, bodies or signals.

Consequence: The AI has to react to any in a sensible way, so they do not feel robotic to the player. Different kinds of interest sources can have different importance to the AI character and can result in different behaviors.

Using the pattern: In *Splinter Cell Blacklist* the player can let *Sam Fisher* whistle to attract nearby guards, shoot lights, or use a camera with integrated radio and explosion device. As guards react to fallen comrades, the player can choose to leave bodies in the open to create an ambush situation.

In *Mark of the Ninja* the player can create sound, e.g. with thrown daggers, or leave bodies in the open to be found. However, found bodies penalize the player score.

Pattern name: Escaping

Description: When the player avatar is detected, the player can try to flee the combat situation and return for another try once the alarm rings out.

Consequence: The AI characters can follow the player or search for him to engage in combat again.

Using the pattern: Both *Splinter Cell Blacklist* and *Mark of the Ninja* utilize this concept, as do most of the games considered as stealth games. It penalizes the player for a bad plan or its execution with an engaging minigame. If it just involves running out a timer, it can get boring and repetitive fast. Having a stable, sensible AI hunting the player can make it much more entertaining.

When looking at these 25 game design patterns, four of which were created additionally, it comes to the attention that nearly all of them could possibly involve the AI system, but not necessarily its structure. In some specific cases, a further look into the AI structure is required. Following is a synopsis of the most common AI structures to allow making the interconnection between game design patterns and AI structure apparent.

5.4. What AI Systems are there?

5.4.1. General Structure

All decision making processes act in the same way: Knowledge is fed into the process, the process works its magic and an action is requested.

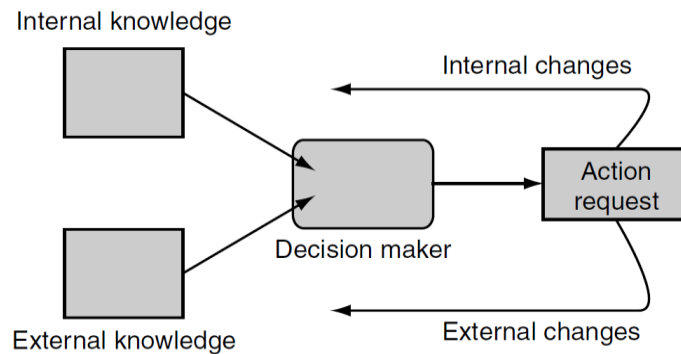


Figure 1 - Decision making schematic

Knowledge can be broken down to external and internal knowledge.

External knowledge is information about the game environment, like positions of characters, level layout, the state of a lever or direction of noise and is identical for any algorithm.

Internal knowledge is the character's internal state or thought process, like health, ultimate goals and recent actions. Internal information depends on the algorithm and is shaped by it.

The action then changes that knowledge, either externally or internally or both.

External changes include pulling a lever, firing a weapon or moving to a room.

Internal changes include a characters opinion of the player, its emotional state or a shift in its ultimate goals.²⁹

There are several ways to get the magic done. Following are some of the most common practices, as presented by Ian Millington and John Funge.

The following recollection is primarily a shortend version with the information we need to determine our approach. For a detailed insight, a read of the original *Artificial Intelligence for Games* is recommendable. It also includes pseudo-code to the interested reader and beginner- and intermediate level AI architect, as well as professionals.

²⁹ Cf. Millington, Ian. p.294

5.4.2. Decision Tree

Decision Trees³⁰ are fast, easily implemented and simple to understand. They have the advantage of being very modular and easy to create. Characters usually refer directly to the global game state, rather than having a representation of what they personally know.

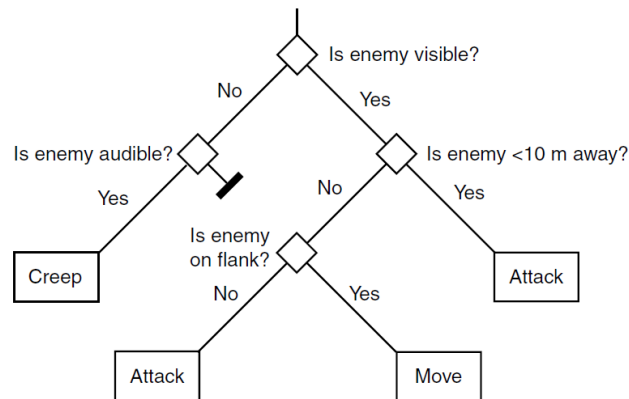


Figure 2 - A decision tree

Decision Trees (Figure 2) are any number of connected decisions. The starting decision is also called root. On each decision point a value is checked. When Boolean logic is required, the tree represents this. Each decision point has a set of ongoing options to chose from. The preferable binary tree only has two options in a set. The algorithm traverses along the tree until it arrives at a leaf, an action that is carried out immediatly. An action can be placed at the end of multiple branches.

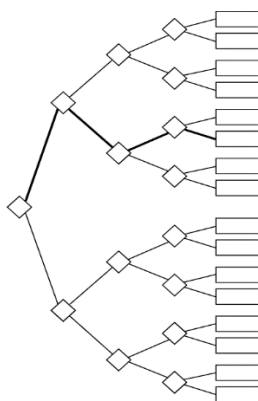


Figure 3 -
Wide decision tree
with decisions

Decision Trees are efficient, because decisions are usually very simple. Also the number of decisions that are considered is much smaller than the actual number of decisions in the tree. Thereby wide trees are even more efficient than deep trees.

Figure 3 shows a wide decision tree with 15 decisions and 16 actions. On each traversal only four checks are needed to be made.

Decision Trees do not require any translation of knowledge used by the rest of the game, because any decision works with primitive data types. This makes it simple to implement, but can cause difficult-to-find bugs. In an ongoing production changes in `get()`-functions can destroy a decision's logic in some languages, like C or C++.

³⁰ Cf. Millington, Ian. pp.294-305

Considering performance, Decision Trees take no memory and in time they perform linearly to the nodes visited, assuming each decision taking a constant amount of time and the tree being balanced, meaning every branch having about the same number of leaves.

“Structuring the tree for maximum performance is a black art. Since decision trees are very fast anyway, it is rarely important to squeeze out every ounce of speed. Use these general guidelines: Balance the tree, but make commonly used branches shorter than rarely used ones and put the most expensive decisions late.”³¹

Next to the advantages already mentioned, the primary Decision Trees are convenient for their minimal set-up.

The major drawback, however, is that it gets unwieldy past the most basic behaviors.³²

5.4.3. Finite State Machine

Oftentimes characters act in a limited set of ways, doing the same action until some event or influence makes them change. The same behavior can be done with Decision Trees, but in most cases State Machines (Figure 4) are easier.³³

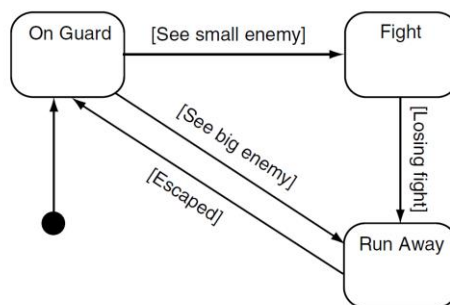


Figure 4 - A simple state machine

In a State Machine each character occupies one state, with an initial state. Actions and behaviors are associated with each state. States are connected with transitions, leading from one state to another. Transitions are associated with one or more conditions, which trigger when they are met. Triggered conditions then fire the transition. The separation of trigger and fire allows transitions to have their own actions. Each iteration, e.g. each frame, an update()-function checks for triggered transitions and, depending on the algorithm, fires the first found.

³¹ Millington, Ian. p.305

³² Mark, Dave. “AI Architectures: A Culinary Guide”

³³ Cf. Millington, Ian. pp.309-331

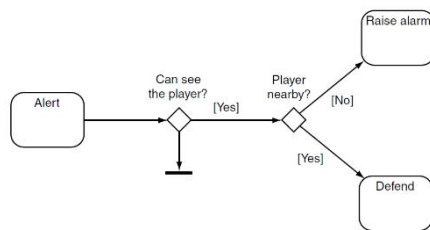


Figure 5 - State Machine with Decision Tree Transitions

In Decision Trees the same set of decisions is always used, so every action is reachable. In State Machines only transitions from the current state are considered, so not every action is reachable. This gives more control over the AI characters' behavior. It is even possible to use Decision Trees as transitions of State Machines.

State Machines are usually implemented by interfaces and generic transitions. These are defined in a data-file and read into the game at runtime. This allows for a GUI [Graphical User Interface] for designers to work with.

Hard-coded State Machines are easy to write, but notoriously difficult to maintain. The main drawbacks are that programmers need to write AI behaviors for all characters. Also the game needs to recompile on each change of behavior, which, in large projects, can take minutes to hours. More complex structures, like Hierarchical State Machines, are difficult to coordinate.

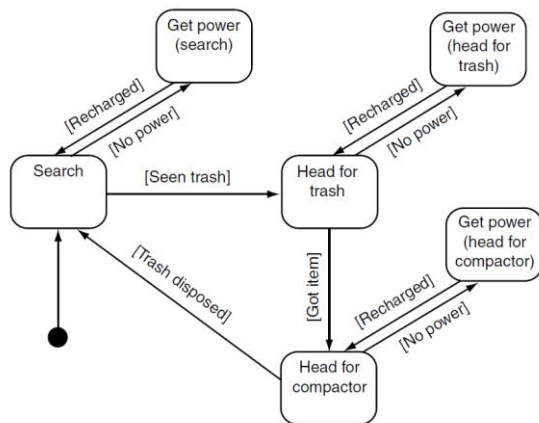


Figure 6 - Alarm mechanism in a standard state machine

Hierarchical State Machines allow alarm behaviors, meaning some important event interrupts normal behavior. Representing this in a normal State Machine leads to a doubling in number of states (Figure 6). To fix this, the states are separated into several State Machines, arranged in a hierarchy (Figure 7). The next State Machine down is only considered, when the higher-level State Machine is not responding to an alarm.

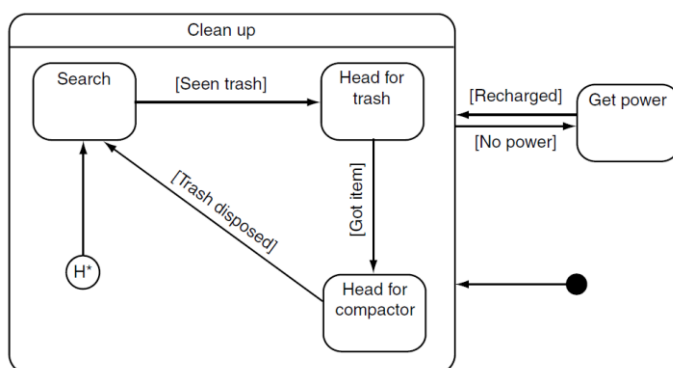


Figure 7 - Hierarchical alarm mechanism

Overall, State Machines are easy to understand and to build, but transitions between states get hard to manage with more behaviors.

5.4.4. Behavior Tree

Behavior Trees³⁴ are seen as a synthesis of State Machines, scheduling, planning and action execution. It is especially easy to understand and create for non-programmers. Basically it is a depth-first search and it will execute any node, which it arrives at.

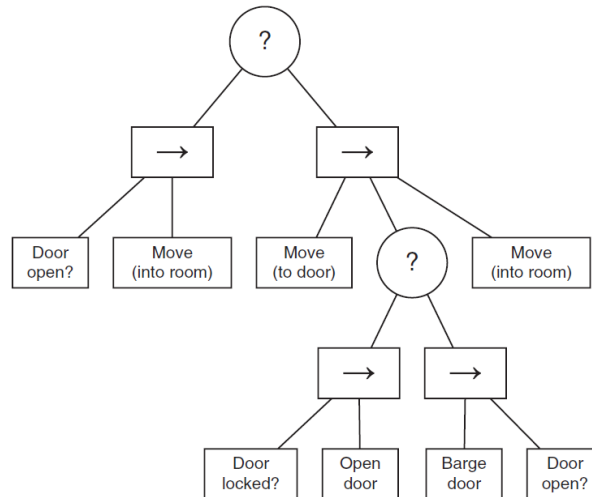
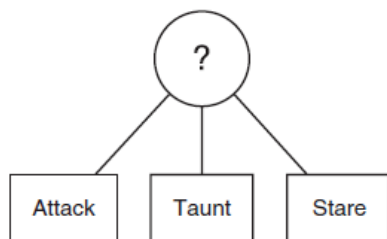


Figure 8 - Behavior Tree for a minimally acceptable enemy

Behavior Trees really shine when coupled with a GUI for designers. Complex sub-trees can be composited into higher-level trees. A designer needs not to worry about the details of a sub-tree. A single, naive behavior is enough to get *something* working. A roughed out behavior can be revisited and refined easily.

Main building block is a Task. Those can be Conditions, which check some property of the game, actions that alter the state of the game, and Composites. The latter make up most of the branches and keep track of the child tasks. Its behavior is based on the behavior of its children and it decides whether to continue through the children or to stop. The most common examples are the Selector and the Sequence.



A Selector node (Figure 9) tries the leftmost node. If it fails, it tries the next node to the right. When all children report failure, the Selector reports failure to the parent node.

Figure 9 - Selector node

³⁴ Millington, Ian. pp.334-371

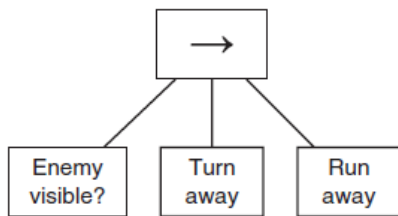


Figure 10 - Sequence node

A Sequence node (Figure 10) tries the leftmost node, if it succeeds, it tries the next node to the right. When one child reports failure, the Sequence reports failure to the parent node.

This can be interpreted as Reactive Planning: Try something, and if it fails, fall back to some different behavior. This is not a very sophisticated form of planning, but grants a good boost in believability.

As actions may take time to complete, the Behavior Tree needs to be able to run multiple behaviors at the same time. Also it would be highly wasteful to search a huge tree from the top. Therefore Behavior Trees are run on multiple threads (Coroutines), which sleep while an action proceeds.

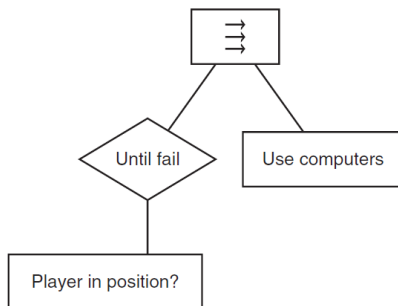


Figure 11 - Using Parallel to keep track of conditions

A Parallel (Figure 11) composite runs all child tasks until one fails. This allows action combinations.

A Decorator composite can provide continuing behavior.

Also this enables higher level Behavior Trees to control group behaviors. (Figure 12)

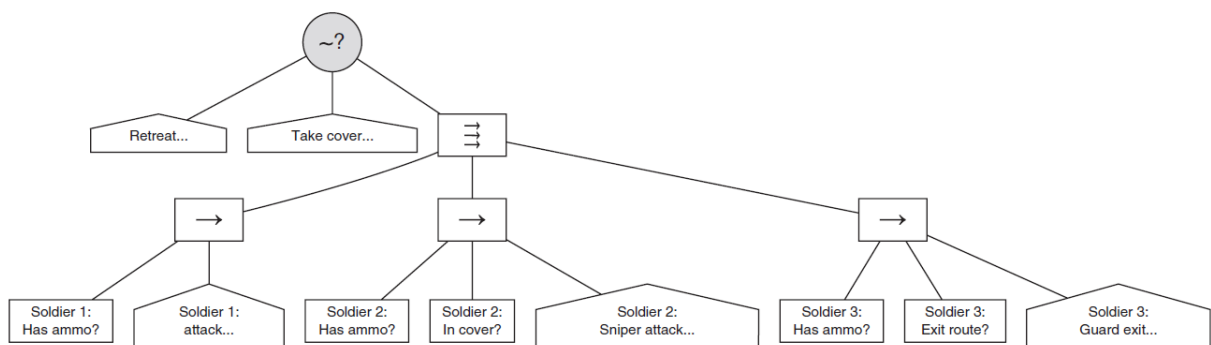


Figure 12 - Using Parallel to implement group behavior

A Limitation of Behavior Trees becomes obvious with the alarm-like behavior of State Machines. Responding to external events in general is much harder. An implementation is possible, but cumbersome, and leads to unintuitive trees. It is possible to integrate Behavior Trees into State Machines, with each alarm-state having its own tree. This combines the best of both worlds, but it also adds a considerable extra burden on the AI authors.

5.4.5. Goal-oriented Behavior

Goal-oriented Behavior³⁵ presents the character with a suite of possible actions and has it choose the one that best meets its immediate needs, also called goals or motives. There may be hundreds of goals, with any number of them currently active. Each is associated with a level of importance, its insistence, represented by a number. Goals with high insistence influence the character's behavior more strongly. The character will try to fulfill the goal or reduce its insistence by picking an appropriate action.

Actions are commonly generated by objects in the world. Every possible action is added to a list and then rated against each goal for its particular impact. An action may be a sequence of actions, with the fulfillment of a goal several steps away.

In a Simple Selection the most pressing goal is chosen then an action, that fulfills it or decreases its insistence the most. This approach is simple, fast and can lead to surprisingly sensible results. However it does not take into account any side effects or timing of the action. The Overall Utility approach utilizes an energy metric, e.g. discontentment. The energy metric can be named and used in a number of ways. In this case high insistences in goals lead to high discontentment of the character. It is often sensible to use some function (squared will suffice) to assign high insistence values a higher importance. Now the character tries to choose an action that reduces its overall discontentment the most. A prediction for discontentment is made for all actions. The action with the least discontentment predicted is then chosen.

Actions may take time to complete. They may also be far away, which case heuristics should be used, as pathfinding takes a lot of resources for computing tens or hundreds of paths. Also insistence may change over time, either set by a designer or in a recency-weighted average of recent changes, e.g. damage, which accomplishes some degree of learning. The prediction of discontentment can encompass this.

Another approach for scoring actions is the Multi-layered Weighting Process. It acts like a filter, through which utility information passes to produce an action score. It can take any number of relevant information, but it needs translation by utility functions.³⁶ Utility-based AI is a capable AI structure in its own right and will be looked at in detail in the practical counterpart to this paper, the Bachelor Thesis titled *Imitation of Player Behavior in Sandbox Games using Utility-based Rational Agents*.

³⁵ Millington, Ian. pp.401-426

³⁶ Mark, Dave. *Behavioral Mathematics for Game AI*, p.346

Goal-oriented Action Planning can be seen as a sub-category of Goal-oriented Behavior. One action may enable or disable several others. Therefore actions must be planned in sequences. Each possible sequence has a world model of the current game state and changes made in the process. The algorithm takes several repetitions to score all sequences of all possible combinations of actions.

A naive approach is immensely wasteful, as the number of combinations increase exponentially with each step looking into the future. When feasible optimal results can be sacrificed for good results by using heuristics, using A* to find the best approach or picking only a single goal, as in the Simple Selection. Also the depth of search for looking into future states can be limited.

The *Smelly* approach can further save on calculations. Each action in the game world emits a signal with a distance fall-off and considering level-geometry.

Goal-oriented Behavior is highly advantageous as it creates lesser workload for AI developers. It scales well with new, unpredictable content, like user generated content or addons. When a new object is added to the game, the actions are fed into the AI system, which figures out how to use them best.

However it is heavy on processing and should only be used with a limited number of characters. Another weakness is the loss of direct control over the AI and it is harder to debug weird-looking behavior.

5.5. Applying AI Structure to Gameplay Mechanics

At this point there is no clear or obvious way to pinpoint any given AI structure to any combination of gameplay mechanics. However, Ian Millington describes a method to approach the Game AI, considering the gameplay mechanics³⁷:

*"[...] we tend to work from a set of behaviors gleaned from the design document, trying to work out the simplest set of technologies that will support them"*³⁸

³⁷ Cf. Millington, Ian. pp.807-812

³⁸ Millington, Ian. p.807

He points out that „*Some decisions are a little more tricky - in particular, the decision making architecture causes headaches*“ and that „*[...] there are no hard or fast rules for selecting a decision making system.*“ He plans the outset for some degree of flexibility and expects the important stuff to be run by a State Machine. There is no report or post mortem available on what AI structure is used for either example, *Splinter Cell Blacklist* or *Mark of the Ninja*. As already stated, there is no hard or fast rules to outright choose a perfect system. However, there are some indicators to what the AI structure should be able to do:

The Alert behavior, common to stealth games, can be found in both examples. For this purpose Hierarchical State Machines were already deemed most proper in 5.4.3.

There are varying Enemy Types in both games as well. The enemies in *Splinter Cell Blacklist* vary by their weaponry and equipment. This indicates that enemies could react in a multitude of ways, which are not always predictable prior to the game. Given the budget and size of the game it is considerable to pick Behavior Trees with a GUI to allow dedicated AI designers to revise the behavior over a lot of playtesting. Alternatively, using the more complex Goal-oriented Action Planning to have the AI react to every possible, unpredictable situation might also be adequate, especially in heated combat situations. The limited count of AI characters allow this heavy processing.

The two kinds of enemies in *Mark of the Ninja*, guards and watch dogs, allow the studio with comparably low budget to commit to two similar State Machines. As the enemies can't climb and are therefore restricted in their movement, no complicated planning is necessary.

In both examples some group behavior can be observed. With the information collected up to this point Behavior Trees seem to be the only option and should sufficiently be exploited in this regard. At this point it is worth mentioning that the AI can be structured in tiers, meaning a strategical layer of AI coordinating the behavior of individual AI characters. There are more similar coordination structures to be found in *Artificial Intelligence for Games* in chapter 6.³⁹

³⁹ Millington, Ian. pp.559-575

While there is no information on how exactly this is done in *Splinter Cell Blacklist*, in *Mark of the Ninja* a sensory manager assigns roles, when multiple AI characters notice the same interest source.⁴⁰ The separation in sentry, investigator and bystander allows for some lightweight group behavior and suggests that these are different states in the same State Machine.

With this information it is plausible to assume that *Splinter Cell Blacklist* and *Mark of the Ninja* both use State Machines, possibly in a hierarchical sense, to satisfy the need for alarm behavior, with an additional coordinating layer to create group behavior. It is also likely for *Mark of the Ninja* to stick to State Machines for all of the limited AI behaviors. Given the budget, *Splinter Cell Blacklist* possibly incorporates Behavior Trees nested in the states, but there is no evidence to that.

6. Evaluation

This paper had a look at game design patterns, common AI structures and the interconnection between them. It was trying to pinpoint those with examples. Only four indicators could be found by the hands of the given information, which are specific to the examples and not conclusive. It became clear that there are indeed no hard or fast rules for choosing an appropriate AI structure.

The problem seems to be that any rules, that could be created, would either be too broad or too specific. An assumption of game design patterns being reliable could also be faulty. There is no way of knowing, which gameplay mechanic or design pattern is used in conjunction with another one. The possibilities to combine them grow exponentially. Also, the classification of a game to a genre is not clear cut and it is imaginable for a game to have a gameplay mechanic outside of its typical genre to create something interesting and new. This, in turn, can possibly lead to entirely different requirements to the AI structure, depending on the significance of the additional feature. For this reason it is understandable, why experienced AI developers tackle each problem on a case-by-case basis. The wording in 5.3. also shows this problem, when the AI characters *should* or *could* or *might* do something, as well as *react accordingly*.

⁴⁰Cf. Miles, Brook. pp.417-418

In process of the research Dave Mark outlined, what is actually important to an AI developer: *"The AI architecture is often not the big differentiator in AI. If that were the case, there would be about 6 types of AI... total. Ever. And all of the games that uses the same one would give the player a fairly similar experience. Instead, the architectures are simply tools that the designers and programmers use to create behaviors out of. While the architectures can enable or make more difficult certain types of things, it is how they are used that makes the most difference."*⁴¹

How the architectures are used, can be found in various post mortems, as already mentioned. One specific problem to stealth games are perception models. As already mentioned in 4. the articles *How to Catch a Ninja – NPC Awareness in a 2D Platformer*⁴² by Brook Miles and *Modeling Perception and Awareness in Tom Clancy's Splinter Cell Blacklist*⁴³ by Martin Walsh show the importance of this aspect.

7. Future Outlook

As the partial success of this paper shows that there is no strict evidence to be found, it seems futile to try more of the same within another genre or with other examples. If there was a way to empirically show the connection, a retake on the matter would be plausible. But it is not clear, how this empirical evidence could be achieved.

However, the paper opens up new, more specific questions. The formulation of these questions can be equally hard, as for the same lack of empirical evidence. At this point it is interesting to ask, which type of game or genre has special concerns about what specific behavior, as in stealth games and perception models.

For a practical approach of an AI developer it is recommendable to look at post mortems of similar games to avoid pitfalls, that were already done by other developers.

⁴¹ Mark, Dave. Comment on "Looking for Related Work and feedback concerning 'Approaches to Game AI under Consideration of Gameplay Mechanics'". GameDev.net, 5 Jun. 2018, <https://www.gamedev.net/forums/topic/697130-looking-for-related-work-and-feedback-concerning-approaches-to-game-ai-under-consideration-of-gameplay-mechanics/> Accessed 9 Jun. 2018

⁴² Brook, Miles. p.413

⁴³ Walsh, Martin. p.313

8. List of references

Abix	“AlphaZero: DeepMind's AI becomes unbeatable in chess”. <i>Steemit</i>, https://steemit.com/technology/@abix/alphazero-deepmind-s-ai-becomes-unbeatable-in-chess Accessed 27 May 2018
Brook, Miles	“How to Catch a Ninja: NPC Awareness in a 2D Stealth Platformer”. <i>Game AI Pro</i>, edited by Steve Rabin, vol. 1, Sept. 2013
Cheng, Jamie	“Classic Postmortem: Klei Entertainment's Mark of the Ninja”. <i>Gamasutra</i>, 7 Sept. 2017, https://www.gamasutra.com/view/news/305166/Classic_Postmortem_Klei_Entertainments_Mark_of_the_Ninja.php Accessed 9 Jun. 2018
<i>Game AI Summit</i>	Games Developer Conference, http://www.gdconf.com/conference/ai.html Accessed 7 Jun. 2018
<i>GDC Vault</i>	Game Developer Conference, http://gdcvault.com/ Accessed 7 Jun. 2018
Hu, Mengchen	Game Design Patterns for Designing Stealth Computer Games Malmö Högskola, Faculty of Technology and Society Department of Computer Science, 2014
Mark, Dave	“AI Architectures: A Culinary Guide”. <i>Intrinsic Algorithm</i>, 1 Nov. 2012, http://intrinsicalgorithm.com/IAonAI/2012/11/ai-architectures-a-culinary-guide-gdmag-article/ Accessed 7 Jun. 2018
Mark, Dave	<i>Behavioral Mathematics for Game AI</i>. Charles River Media, 2009, p.56
Mark, Dave	Comment on “Looking for Related Work and feedback concerning ‘Approaches to Game AI under Consideration of Gameplay Mechanics’”. <i>GameDev.net</i>, 5 Jun. 2018, https://www.gamedev.net/forums/topic/697130-looking-for-related-work-and-feedback-concerning-approaches-to-game-ai-under-consideration-of-gameplay-mechanics/ Accessed 9 Jun. 2018
<i>Mark of the Ninja</i>	Klei Entertainment, https://www.klei.com/games/mark-ninja Accessed 7 Jun.2018

Millington, Ian / Funge, John	<i>Artificial Intelligence for Games</i> . 2nd ed., Morgan Kaufmann Publishers, 2009
Orkin, Jeff	The AI Programmer's Bookshelf. MIT Media Lab, http://alumni.media.mit.edu/~jorkin/aibooks.html Accessed 7 Jun. 2018
Rabin, Steve	<i>Game AI Pro</i> , http://www.gameaipro.com/ Accessed 07.06.2018
<i>Splinter Cell Blacklist</i>	Ubisoft, https://www.ubisoft.com/de-de/game/splinter-cell-blacklist Accessed 7 Jun 2018
<i>Splinter Cell Blacklist</i>	<i>Steam Store Page</i> , https://store.steampowered.com/app/235600/Tom_Clancys_Splinter_Cell_Blacklist/ Accessed 9 Jun. 2018
Thompson, Tommy	"AI 101 – Part 4: 'Game' AI – A unique challenge". <i>AI and Games</i> . https://aiandgames.com/ai-101-part-4/ Accessed 7 Jun. 2018
Walsh, Martin	"Modeling Perception and Awareness in Tom Clancy's Splinter Cell Blacklist". <i>Game AI Pro</i> , edited by Steve Rabin, vol. 2, Apr. 2015, p.313

9. List of figures

Figure 1 - Decision making schematic	17
Figure 2 - A decision tree	18
Figure 3 - Wide decision tree with decisions.....	18
Figure 4 - A simple state machine.....	19
Figure 5 - State Machine with Decision Tree Transitions.....	20
Figure 6 - Alarm mechanism in a standard state machine	20
Figure 7 - Hierarchical alarm mechanism	20
Figure 8 - Behavior Tree for a minimally accaptable enemy	21
Figure 9 - Selector node.....	21
Figure 10 - Sequence node	22
Figure 11 - Using Parallel to keep track of conditions	22
Figure 12 - Using Parallel to impement group behavior.....	22

**All above figures are found in Artificial Intelligence for Games by Ian Millington*

Erklärung:

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig erstellt und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe.

Soweit ich auf fremde Materialien, Texte oder Gedankengänge zurückgegriffen habe, enthalten Ausführungen vollständige und eindeutige Verweise auf Urheber und Quellen.

Alle weiteren Inhalte der vorgelegten Arbeit stammen von mir im urheberrechtlichen Sinn, soweit keine Verweise oder Zitate erfolgen.

Mir ist bekannt, dass ein Täuschungsversuch vorliegt, wenn die vorstehende Erklärung sich als unrichtig erweist.

Alzenau, 11.06.2018

Ort, Datum



Unterschrift

Erklärung zur Archivierung:

Bitte zutreffendes ankreuzen:

- ☐ Mit der Archivierung der gedruckten Abschlussarbeit in der Bibliothek
bin ich einverstanden.

- ☐ Mit der Archivierung der gedruckten Abschlussarbeit in der Bibliothek
bin ich nicht einverstanden.

Begründung:

- ☐ Die Arbeit ist gesperrt, da sie in einem Betrieb durchgeführt wurde und ihr Inhalt ausdrücklich durch diesen gesperrt ist. (Vgl. ABPO § 18 (9))
- ☐ Persönliche Gründe

Alzenau, 11.06.2018

Ort, Datum



Unterschrift