Impact of Game AI Systems on Player Experience

Daniel Ciugurean
Technical University of Cluj-Napoca
Str. Memorandumului, nr.28
danielciugurean@outlook.com

Bogdan Maxim
Technical University of Cluj-Napoca
Str. Memorandumului, nr.28
Bogdan.Maxim@cs.utcluj.ro

Dorian Gorgan
Computer Science Department, Technical University of Cluj-Napoca
Str. Memorandumului, nr.28
Dorian.gorgan@cs.utcluj.ro

ABSTRACT
In this paper, we explore the impact of the most common building blocks of game AI systems on player experience. We conduct an extensive survey of the most common techniques applied in practice, and then analyze their individual and combined impact on player experience. Next, we argue that a systemic AI that guides the player adversary on a macro level, raises the replay value that keeps the users engaged and enhances the overall player experience. We observe player behavior through a series of playtesting sessions on an AI-based game prototype, with variations on each AI subsystem. Finally, we provide an analysis of our results.

Author Keywords
Artificial Intelligence, AI-based game design, Systemic AI

INTRODUCTION
Given the state-of-the-art in video games with high production values, the AI systems seem like decorations, an afterthought rather than a central component at design time. As a direct consequence of that, with a few exceptions, most of the AI in games is made up of simpleton enemies, with easily exploited behaviors, waiting to be eliminated from the path of the player. Perhaps, the difficulty of making a good game AI lies in the implicit question: "What makes a good game AI?" Even though this question has multiple incomplete answers, for every genre, on closer inspection, a lot of common AI design and implementation techniques emerge. Good AI has its own goals, although such an artificial life system has to be grounded in a world where the player character can have a meaningful experience. The initial version of *S.T.A.L.K.E.R.: Shadow of Chernobyl* had an A-Life system (which had to be toned down near the eventual release), where every single living entity had its own goals, could complete missions and explore the world [7], even complete the final end-game mission without the player's participation. The AI characters could do all the actions a player character would, but the game was simply not fun. The objective of this paper is to identify and analyze through playtesting the most common and well-accepted AI components that have a direct impact on the player experience. The paper is divided into 6 sections and the content in each one is structured as a logical progression of our thought process. In the Related Work section we describe all the relevant documented findings that describe the link between the game AI systems and player behavior. The AI Components section presents a common nomenclature for the building blocks of a game AI architecture. We then develop a customizable AI framework in the Game Implementation section, and then test and analyze our initial assumptions in the Experiment Analysis section. Finally, we open our work for future developments in the Conclusion section.

RELATED WORK
We build upon the classification approach given by Treanor et al. in [1], where the authors provide the necessary terminology to reason about AI-based games. We would like to particularly emphasize the AI modes of operation: *background* and *foreground*. While the authors focus more on the foreground version, our work describes the AI systems' building blocks with a maximum return on player feedback. In [2], the author mentions that the AI system of *Halo* reserved about 15% of the Xbox CPU, which reaffirms our assumptions that the AI component is an integral part of the overall game experience. Bungie conducted playtests during the development of the game and specifically an A/B test with weaker vs tougher enemies. When asked whether the enemies felt intelligent, the number of testers who answered positively jumped from 8% to 43%. Of course, such findings aren't true for every type of game there is, but the idea of a more aggressive AI proved successful in a large number of games. The AI behavior needs to be consistent with the design philosophy of the game. As ID Software found out during the development of Doom 2016, the initial iteration of the AI system was quite aggressive, and would determine the player to play defensively, which was a complete opposite of the initial intention [5]. A more pragmatic approach is to vary the aggression levels with respect to the player. The AI director presented in [3] tracks both the state of the world and of each player in order to vary the stress level of the game, such as spawning more enemies or not. Moreover, the location of enemies and usable resources are determined at map generation, which increases the replay value of the game, and makes it very hard for players to memorize the patterns. Such a system can be tracked to the days of *Pac-Man*, where the element of tension was controlled by a stream of waves, not a constant array of attacks on the player [8]. Player frustration minimization is a clear

heuristic pattern during development time. In *Batman: Arkham Asylum*, the enemy AI will rarely turn around to find its path towards a previously visited location [10]. In the scenario where the player would sneak up behind an enemy, a sudden turn of the Non-Player Character (NPC) would result in the player being detected and possibly even losing the game. A similar off-balancing of odds in player's favor is found in Uncharted 2. In order to ensure that the player has a chance of taking a few shots at the enemy when getting out of cover, the game logic has a *Time To Accurate Cover* value [11]. Thus, starting with a null accuracy, the AI will gradually recover to the normal value.

AI COMPONENTS

In this section we discuss some existing common building blocks of game AI systems, which have been applied successfully in industry. We do not give any directions on how to build such systems other than the overall architecture. Our goal is to observe the individual impact of each such system on the end user's experience. Moreover, the overall combined impact of multiple subsets as well as the entire set of components is further analyzed.

Component	Role	Example(s)
AI Director	Promotes replayability	Left 4 Dead
Macro Systemic AI	Complex simulation of different AI entities, given a specific set of strictly enforced shared world rules	Zelda: Breath of the Wild
Movement	Visual feedback which communicates the range of the AI's world navigation capabilities	Uncharted 4: A Thief's End
Perception	Recognizes nearby entities of importance	Alien: Isolation
Behavior	The internal cognition model of the AI. Varies from the ability to learn to the ability to reason and plan	Halo, F.E.A.R.
Feedback	An audio-visual mode of communication to the player of the internal AI state	Splinter Cell: Conviction
Micro Systemic AI	Allows unpredictability in a rule-based world	Watch Dogs 2

Table 1. An overview of common game AI components, their roles and relevant game examples.

Background AI

AI Director (Macro AI)

Introduced in the 2008 team-based FPS *Left 4 Dead*, the AI director diminished the overall reliance on manual, micro-managed level progression of the player. The main goal of such a system is to promote replayability, which constitutes an integral part in player experience, given limited content and its speed of development. Such a system is quite extensible, given a more extensive vocabulary of possibilities and better approximation of player stress value. A similar technique has been applied in more recent games such as *Far Cry 4* [6].

Macro Systemic AI

Given a set of game systems, explicitly designed with external side effects in mind, the resulting graph connections between the system nodes create the overall systemic AI, where a system can influence another, considering that there is a link between the two. Whereas the AI director takes the role of guiding the player into new and interesting situations, the Systemic architecture is put into play after a player action has occurred. A chain reaction of events comes into play, where all the entities in the game apply their awareness of the existing game systems onto the game world. Such a system gives rise to emergent gameplay, which generates unexpected (yet controlled by the rules of the world) and memorable situations for the player.

Unpredictability

While predictability is essential at the micro (action) level, because the expected immediate result of a player action should be deterministic, it reduces the replay value dramatically at the macro level. Any predictable macro action, such as enemy placement, triggers/scripts results in player memorization and effective "cookbook" strategies. Predictability may be fine in a fully scripted and controlled game, but such experiences have a low replay value from a gameplay standpoint. Unpredictability is more of a property of the AI, rather than a component.

Foreground AI

Movement

The movement component of a game AI system manages the actor position in the environment, as well as its collision and animation. The way the AI controlled character moves has a tremendous impact on player perception of its intelligence. An efficient pathfinding system is critical in such a case, although the strict following of the direction vectors isn't. A straight line movement from point A to B is perceived as robotic, even more so than sharp movements at corners: players are much more forgiving of the latter. A curved navigation path with allowed slight deviations is much more natural and fluid, even though it's not the most efficient path by cost value. The way the NPC navigates the world elicits a certain reaction from the player. The NPC being unable to go around an obstacle to reach the player may be funny at first, but detrimental to the overall experience. Such a limitation breaks both immersion and the challenging aspect, since the player can and will easily exploit it.

Vision and other perception sensors

The vision logic of the AI controlled actor is responsible for its line of sight, field-of-view (cone of vision) and its relative position to the player actor. The AI vision space should have a collision model, eliminating the possibility of seeing the player through the walls, while simultaneously recognizing nearby objects of importance. The vision model is often the most critiqued and difficult to optimize with respect to gameplay, especially in stealth games. The alien in *Alien: Isolation* is blind, so it can only perceive the player through proximity and player-generated sounds.

Behavior

A concurrent execution of behaviors represents the intention of the AI system. The frequency with which these behaviors are implemented is either static or learned, as is the case for the game *Alien: Isolation*. A lot of the enemy's behaviors are locked at first, and are unlocked as the player progresses, to give the illusion of it becoming smarter. It also takes the player's actions into account, which encourages creativity from the player's perspective. *Metal Gear Solid 5* includes a system which adapts to the player's playstyle [9], such as the enemies wearing helmets after too many headshots and other methods, which prevent the player from exploiting the same tactics throughout the game. The actions that determine the behavior can be implemented in either a state machine or a behavior tree, popularized by *Halo 2*, and the default template in Unreal Engine 4. The GOAP (Goal Oriented Action Planning) system of *F.E.A.R.* [4] has only two states: **Goto** and **Animate**. A planning system over a sequence of such states resulted in one of the most memorable and lauded AI-based game experiences, even at the time of this writing.

AI feedback

An observable AI character gives the players a few hints about its intentions. One of the most common ways of doing that is through sound cues: the player makes a loud noise, which triggers the enemy AI to talk generic lines like "What was that?" etc. *Splinter Cell: Conviction* displays a silhouette of the player, which communicates the last location of the player character known to the AI. Such audio-visual cues are necessary in order to give the player minimal clues about what the AI is thinking. Therefore, the player can build his own internal model of the behavior, an approximation of the real system. This, in turn, allows him to plan ahead and use this knowledge to his own advantage. Distinct personalities such as those found in *Pac-Man* for the ghosts, *Civilization IV* or *Total War* campaign AIs, makes them feel smarter and more engaging, thus having a positive feedback on the overall experience for the player.

Micro Systemic AI

This requirement has a high priority in sandbox-type games such as *Zelda: Breath of the Wild*, *S.T.A.L.K.E.R.*, *Far Cry 4* and *Watch Dogs 2* or any game from the immersive-sim genre, such as *Deus Ex*, *Dishonored* or *Prey*. The AI is allowed to take an unpredictable sequence of actions within the game world, constrained by the rules of the nearby activated game systems. For example, in *Zelda: Breath of the Wild*, an enemy may light up a wood stick with fire, so that he could deal more damage to the player. The enemy AI may use a hint system such as that found in *Half-Life 2*, where the NPC can weigh all the available nearby suggestions in order to take the appropriate action. Such emergent behavior is difficult to control at run time and may be difficult to balance in the final game. Therefore, the game world rules need to be strictly consistent – any breakage in the rule hierarchy may discourage the player from further experimentation, who becomes biased towards actions that always work. This can be traced to the well-known problem in reinforcement learning called the explore-exploit dilemma.

GAME IMPLEMENTATION

The aforementioned components of a generic AI system give us a framework for reasoning about their footprint on the overall player experience. We developed an AI-based game prototype, with the intention of testing our assumptions given earlier. Consequently, we could control all the components of the AI system.

Game prototype

Figure 1. A screenshot from the playable prototype demo. The enemy AI cannot shoot the player character, so it finds a new position to attack from.

Description

The prototype puts the player at the center of the action, in a *One on One Deathmatch* inspired game mode. It is a third-person shooter, with no cover mechanics. Since the focus of our writing is on player-NPC interaction, the graphical aspect of the demo is modest.

AI Director

Upon death, both the player character and the AI controlled character can spawn at a randomly chosen location (although manually placed at design time). The AI director can give certain hints to the AI controlled character, such as mentioned in the following implementations:

1. The enemy NPC starts off tabula-rasa upon (re)spawn.
2. The enemy NPC remembers the previous state before the respawn. As such, it can take the possible player location into account when moving through the world space.
3. Depending on the player's success, the AI director may adjust the parameters of the NPC character (such as health, damage etc.) or spawn an additional but weaker enemy. This ties in to the Micro Systemic AI.

Movement

The controlled actor (either by the player or by the AI) has a few movement capabilities: normal run, sprint, crouch movement and jump. As such, the player can switch rapidly between them, which generates a good amount of test cases. Four versions of the movement component were implemented:

1. The AI moves in a straight line to the destination. The enemy AI sees the player and immediately goes to the player, without any path deviations. The friendly AI follows the player character at a very close distance, and stops whenever the player stops.
2. The AI is allowed to deviate from the shortest path vector, following a curved path instead. The enemy AI, upon seeing the player, gets in the range of an allowed radius from the player and stops.
3. The enemy AI is capable of moving around the player in a strafing motion, trying to flank the player from different angles. This version brings an improvement on the AI behavior in the **IDLE** and **SEARCHING** states. Thus, it doesn't just choose a random location in the immediate radius, but tries to make a smoother motion.
4. This implementation is relevant to the following behavior of the friendly AI. In order to avoid rapid state flutter between the movement capabilities, we introduce an element of hysteresis, which delays the state change and determines a smoother transition between movement types.

The default version of pathfinding from Unreal Engine 4 was used for all implementations.

Vision

The vision logic is the only perception logic that we implemented. The AI has a field of view of 110 degrees, with a center axis placed in the head. The enemy AI constantly shooting a line-trace in order to determine if the player character is in its field-of-view (FOV). The following versions have been tested:

1. The vision logic has no collision model. If the player is in the FOV of the AI character, he is considered to be detected, and the AI can reason about his position.
2. The vision logic has a collision model. A line-trace tests whether the first hit is the player character or a generic collision-aware world entity. Thus, the AI cannot see through walls, and can only reason about the position of the player if he is actually visible.

Behavior

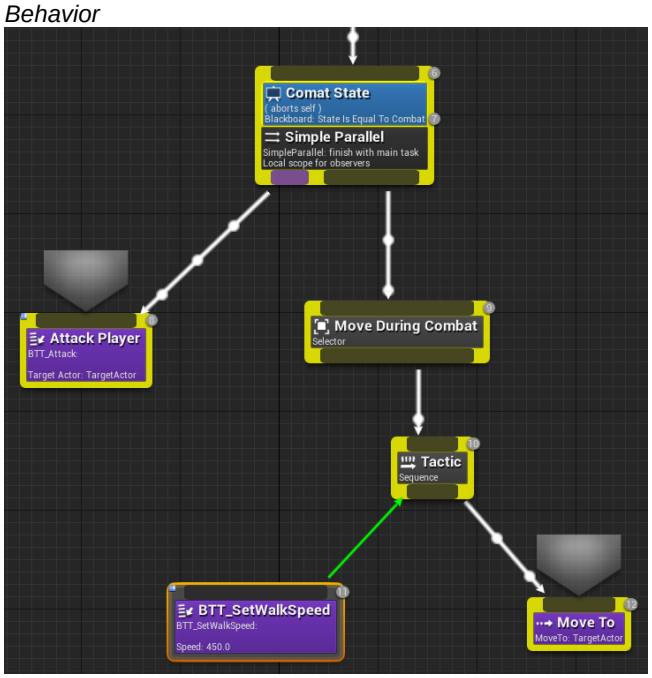

Figure 2. Behavior subtree for the movement component during the combat state. It allows the AI character to move and shoot simultaneously. Note: not the complete version.

We used behavior trees (the default mode in Unreal Engine 4) in order to describe the AI logic of both the enemy and friendly AI. This particular component received the most attention during development, since this is where the intelligence of the AI system is most perceived.

1. The enemy AI moves closer to the player, until it is in a close enough range to shoot. It stops, and attacks the player indefinitely. If the player distances himself, the AI stops shooting, moves closer and resumes action. It has no concept of memory, so, if the player gets out of the AI's FOV, the enemy NPC gets back to the **SEARCHING** state.
2. The enemy AI can move and shoot, two distinct behaviors which run in a concurrent manner. When placed within the player's shooting radius, the AI stands still and attacks the player.
3. The enemy reaches the player and starts flanking and shooting along the way. The shooting accuracy may suffer, but the NPC is harder to hit,

which gives him more options to work with. It can take nearby obstacles crouch for cover, using the navigation mesh and hint-nodes as an information database.

4. The enemy AI is responsive to player actions. It can turn around if shot from behind or observes a shot taken in his direction (without being hit), and searches the possible position from where the player might have attacked.

The complete implementation of the behavior tree has about 25 nodes. The major decision branches are weighted, which directly impact the randomness aspect of the system. These come into play when the systemic links are activated, which is presented later.

AI feedback

The feedback loop is an important component in rewarding the player for his accomplishments. We present a few versions of audio-visual cues that give the user the necessary feedback:

1. Upon player respawn, the enemy AI taunts him. Similarly, the player character says a few words that confirm a winning battle.
2. Every controllable action such as sprinting or jumping has an associated sound cue, which informs the player about the AI's actions, if placed in a close enough range.
3. Visual body damage system. The controlled actor may slow down if shot in the legs, drop the gun if shot in the arm or take increased damage if shot in the head. Each one may trigger a certain animation, which gives additional visual feedback about the AI state.

Systemic AI

The systemic links are tied in to all the aforementioned subsystems. The AI is allowed to improvise a bit (where given a set of weighted possibilities, it can randomly choose any version, with a small probability). Below, we mention a list of actions an AI character can choose from:

1. If the health value is below a certain threshold, the AI can flee the player, and find the nearest cover.
2. If the gun is dropped, the AI can choose to either pick it up again, flee or attack the player without a gun, albeit with inferior damage.
3. Given multiple AI enemies, they can agree on a flanking strategy, so that the player can be attacked from behind.

The systemic behavior can either be turned *on*/*off*, or stripped of some complexity. Through our playtesting sessions, described in the next chapter, we chose the first version, because the combinatorial complexity became unwieldy. However, all the available systemic possibilities were included in the activated version.

EXPERIMENT ANALYSIS

The number of implemented components has a direct influence on the speed of user evaluation. Given the combinatorial explosion of possibilities, we start off with the worst iteration of each component, and then choose the best individual one to get to the next level in the combinatorial structure. Our approach to playtesting is similar to that conducted at Valve [12]. We used both traditional methods such as *direct observation, verbal reports* and *Q&A sessions*, and technical approaches such as *design experiments* and *surveys*. The prototype was tested by 10 people, with varying levels of expertise in video games, ranging from beginner to competitive.

Even though traditional methods do interfere with player behavior, it did give us enough feedback to reason about the way the player interacts with the game. During the Q&A sessions, the players showed significant preference towards the more advanced implementations. In the case of the most rudimentary implementation of the vision system (the one that allowed the AI to see through walls), the players felt that the AI was cheating, which downgraded the experience for them, even if all the other components (given the most advanced implementations) were received positively. This increases the certainty in one of our hypothesis, which states that there is no possibility to include AI shortcuts or cheating into the vision system, especially in a game where player planning is a gameplay feature. The Q&A sessions revealed that the feedback given by the AI character is highly related to player interaction, but not to the perceived behavioral cleverness of the enemy AI. The extension of the behavior component with systemic elements such as environment awareness has given us the most positive feedback during these sessions. Enemy communication during combat had a similar strong impression on the players. Further development with both visual and audio cues is required here (such as calling other members by unique names), our hypothesis being that natural information propagation within the shared knowledge base of the enemy would further improve the player's perception of the game AI's ability.

We also conducted a survey session, where the players were asked to answer a few questions with forced choices, in order to get less biased responses. Even though the rating system diminishes the finer details of the experience, which in turn constrains the search space, we can map user experiences to bucketed values. These are the questions that were asked in the survey (all answers are on a scale from 1 to 9, where 1=bad and 9=good):

1. How would you rank the AI movement behavior?
2. How well could you read what the AI was thinking?

3. How did you perceive the AI's non-combat behavior?
4. How would you rank the overall AI behavior? (This question was asked twice for the demo where we tested weaker vs stronger AI in terms of health and damage)
5. How well did the AI react to your actions?
6. How well did the game system react to your play style?
7. How would you rank the AI behavior in the multiple enemy demo?

Regarding question 4, we confirm the results presented in [2], that stronger AI are perceived as smarter than weaker AI, as suggested by the survey statistics. The stronger AI was given the same variables as the player character, which made the game fair in terms of overall in-game capabilities. Given the sample size, we consider a simple scoring system, which is the average of the individual per-question scores. When ranking the simpler implementations, the testers leaned more towards lower scores, with an average of 2.4 – a stark contrast to a score of 7.2 for the most advanced implementations. A more rigorous method of analysis is required, but a lot of the ones we have considered are quite intrusive, such as psychological analysis, eye tracking, EEG etc. Given our choice of analysis methods, the general agreement is that a combination of the most advanced implementations received the most positive feedback. However, the goal of the AI does not resolve around winning against the player, but rather to provide a meaningful challenge, even if it means downgrading the underlying logic. As such, the presented results are still strongly linked to the context of the research, and not very general.

CONCLUSION

The process of developing a game is a difficult one. In production, the final iteration of the game usually becomes fun a month before the eventual release. The process is made even more difficult by the fact that the vision of the game is not clear until much later into the production process. However, we have shown that an AI-based game design provides a central focus point for the overall player experience, which speeds up the subsequent iteration cycles. We have formalized the common AI components, which have been applied successfully in production, then tested and analyzed their combined impact through a series of both traditional and technical methods. Further work may include more numeric methods of analysis, such as statistical methods, in order to discover patterns in user behavior, although such a method would require a much bigger sample size.

ACKNOWLEDGMENTS

This research has been carried out in the Computer Graphics and Interactive Systems Laboratory (CGIS) of the Computer Science Department, in The Technical University of Cluj Napoca.

REFERENCES

1. Treanor, M., Zook, A., Eladhari, M.P., Togelius, J., Smith, G., Cook, M., Thompson, T., Magerko, B., Levine, J. and Smith, A. (2015), *AI-Based Game Design Patterns* Foundation of Digital Games Conference, June 2015.
2. *The Illusion of Intelligence: The Integration of AI and Level Design in Halo*, March 2002, available at: http://halo.bungie.org/misc/gdc.2002.haloai/talk.html (May 2018).
3. Booth, Michael, The AI systems of left 4 dead. *Artificial Intelligence and Interactive Digital Entertainment Conference at Stanford* 2009.
4. Orkin, Jeff. Three states and a plan: the AI of FEAR. *Game Developers Conference*. Vol. 2006.
5. *Make me think, make me move: New Doom's deceptively simple design*, April 6th 2016, available at: https://www.gamasutra.com/view/news/295254/Make_me_think_make_me_move_New_Dooms_deceptively_simple_design.php (May 2018).
6. *The definition of artificial insanity*, May 17th 2017, available at: https://aiandgames.com/the-definition-of-artificial-insanity/ (May 2018).
7. *A-Life, Emergent AI and S.T.A.L.K.E.R.: An Interview with Dmitriy Iassenev*, February 25th 2008, available at: http://aigamedev.com/open/interviews/stalker-alife/ (May 2018).
8. *The Pac-Man Dossier*, February 23rd 2009, available at: https://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1 (May 2018).
9. *MGSV: Phantom Pain Enemies Response System, Defense, Vehicles Guide*, September 5th 2015, available at: https://segmentnext.com/2015/09/05/mgsv-phantom-pain-enemies-response-system-defense-vehicles-guide/ (May 2018).
10. *Arkham Intelligence*, May 5th 2014, available at: https://aiandgames.com/arkham-intelligence/ (May 2018).
11. *The Secrets of Enemy AI in Uncharted 2*, November 3rd 2010, available at: https://www.gamasutra.com/view/feature/134566/the_secrets_of_enemy_ai_in_.php?print=1 (May 2018).
12. Ambinder, Mike, Valve's approach to playtesting: The application of empiricism. *Game Developers Conference* 2009.