# Game Design for Classical AI

**Ian Horswill**
Northwestern University
ian@northwestern.edu

**Abstract**

Reasoning using expressive symbolic representations is a central theme of AI research, yet there are surprisingly few deployed games, even within the AIIDE research community, that use this sort of "classical" AI. This is partly due to practical and methodological issues, but also due to fundamental mismatches between current game genres and classical AI systems. I will argue that if we want to build games that leverage high-end classical AI techniques like commonsense reasoning and natural language processing, we will also have to develop new game genres and mechanics that better exploit those capabilities. I will also present a design sketch of a game that explores potential game mechanics for classical AI.

## Introduction

Reasoning and representation using expressive formalisms such as predicate logic (of whatever form) are a central thrust of AI research, and historically its biggest thrust. However there is very little use of expressive knowledge representation in game AI. I cannot, for example, find a single shipped game (as opposed to a research prototype) that even has an implementation of unification (Robinson, 1965), much less the concept and role hierarchies of modern knowledge representation languages such as CycL. Apart from A* and game tree search, there's surprisingly little overlap between contemporary game AI and a typical undergraduate AI course.

This is not to say that classical AI is broken or that game AI is primitive. Rather, the two have evolved in different directions under different pressures. Contemporary game AI and contemporary game genres and mechanics co-evolved. Game AI adapted to best support the genres and mechanics of contemporary games, while genres and mechanics adapted to better leverage contemporary game AI. Not surprisingly, research AI systems would have a very difficult time beating contemporary game AI for first-person shooters. Behavior trees (Isla, 2005) work really well for FPSes, and while Cyc (Cycorp, 1984) and SNLP (McAllester & Rosenblitt, 1991) are "smarter" in certain ways than behavior trees, they're smarter in ways that are difficult to show off in an FPS, while being much more expensive.

Natural language processing systems have a similar problem in game AI. When the RPG genre was developing, game hardware couldn't even run even simple NLP systems; human-authored dialog was the only option. So RPGs evolved to make the best possible use of the strengths of human-authored (fluency, drama, delivery), while trying to minimize its deficiencies (limited player choice, repetition, etc.). Unfortunately, NLP systems make the opposite trade-off: generativity at the cost of fluency and performance. So while, they might have been successful if they could have been used back in the original *Legend of Zelda* (Nintendo Corporation, 1986), current NLP systems can't possibly live up to the expectations for a game like *Mass Effect 3* (BioWare, 2012), because the genre has evolved in such a way as to accentuate NLP's deficiencies and minimize its advantages.

I see no solution to this problem except for us, as AI researchers, to develop new genres and mechanics ourselves, ones that are better matched to the capabilities of our systems. As Hecker (2010) has argued, AI and gameplay are intimately connected, much more so than graphics and gameplay; graphics researchers can work on global illumination without thinking about gameplay, but AI researchers don't have that luxury.

Instead of looking for ways to integrate high-end AI into existing genres, let's instead try to design new mechanics around high-end AI. This is certainly what made successful AI-heavy titles like *The Sims* (Wright, 2000), *Prom Week* (McCoy, Treanor, Samuel, & Reed, 2012), and *Versu* (Evans & Short, 2013) work.

In this paper, I will discuss some of the issues with designing game mechanics for classical AI, and provide a design sketch for a game that explores such mechanics.

## SHRDLU Game Mechanics

Suppose we wanted to design game mechanics around an AI system with roughly the capabilities of SHRDLU (Winograd, 1972). Even though SHRDLU is over 40 years old, it's difficult to find any examples of games that support even its level of AI functionality. Moreover, game hardware is more than powerful enough to run a system like SHRDLU, so it's realistic to think about building gameplay around it.

SHRDLU has two primary capabilities: *problem solving* and *question answering*. That is, given a blocks-world goal, like "build a tower", it can manipulate the world to achieve it. And given a question like "how many tall blocks are there?" it can tell you the answer. So the question is how to center game mechanics around NPC problem solving and question answering, and to do so in a way that is tolerant of the *limitations* of the technology.

### Challenges

One limitation is *fragility*. Classical AI systems, especially simple SHRDLU-like ones, can be very smart, but they can also do stupid, crazy things that can annoy the user and destroy the illusion of intelligence. There are a number of possible strategies for dealing with fragility, including:

- **Build perfect or near-perfect AI**
  Good luck with that.
- **Constrain gameplay**
  enough to allow exhaustive debugging of the AI within the space of possible gameplay. This is doable, but reduces the generativity, and therefore the value, of the AI system.
- **Narrative alibis**
  Use the game narrative to explain away the dysfunctional behavior. For example, make the AI characters be zombies, children, or kooky aliens.
- **Adult supervision**
  Make the gameplay involve keeping the AI out of trouble (as in *The Sims*).
- **Make it the player's problem**
  Incorporate AI debugging into the gameplay. Then failure is expected and is the player's "fault."

A related problem is making the AI *transparent* so that the player understands why it does what it does and has some basis for anticipating its behavior. An NPC's entirely rational behavior might appear random (and stupid) if the player misunderstands its goals and beliefs.

*Third Eye Crime* (Moonshot Games, 2014) has a particularly elegant solution to this problem. It overlays the internal state of the NPCs' search system on the screen so the player can understand what the NPCs are thinking (Isla, 2013). This is explained in the game narrative by making the player character telepathic. Moreover, this allows the designers to create a new game mechanic: unlike traditional stealth games, the player character *has* to be discovered; the gameplay lies in using the player's knowledge of the AI's state to successfully avoid the NPCs after discovery.

A third challenge is *controlling player expectations*. For the foreseeable future, any classical AI system we can put in a game is going to have serious limitations. It will be important to design the gameplay so as to teach the player in as painless a manner as possible exactly what the AI's capabilities and limitations are. For example, any question answering system will involve a tightly constrained grammar and vocabulary. One way or another, the player needs to learn what the grammar and vocabulary are, so feedback above and beyond "I don't understand" is important.

### Game mechanics for AI problem solving

Most AI-based gameplay is centered around the AI solving problems, at least for an inclusive definition of the term. There are several different mechanics that have already been explored for problem solvers. These vary according to the role the AI plays:

- **Adversary**
  Gameplay involves outwitting or otherwise overcoming the AI. Examples include game-tree search for turn-based strategy games (Whitehouse, Cowling, Powley, & Rollason, 2013), as well as NPC AIs using behavior trees (Isla, 2005) or STRIPS planning (Orkin, 2002).
- **Ally**
  The AI assists the player, but is largely autonomous from the player, rather than waiting to take orders. For example, the Elizabeth character in *Bioshock Infinite* (Abercrombie, 2014).
- **Subordinate**
  The player achieves her goals by tasking the AIs: giving them goals and relying on them to achieve them. For example, in squad-based tactical shooters, and RTS games.
- **Ward**
  The NPCs have limited autonomous AI, but require management by the player to keep them out of trouble, as in *The Sims* (Wright, 2000).
- **Puzzle**
  The player is expected to reverse-engineer the behavior of the AI in order to understand how to manipulate it to her ends. For example, in *Prom Week*, the player is intended to learn the game's model of "social physics" well enough to exploit it.

Roles are not mutually exclusive; adversarial AIs are often also puzzles that the player is intended to solve by learning the AIs patterns and exploiting them.

Different roles involve different expectations for the AI's sophistication. Subordinate AI should generally be capable and reliable, while ward and puzzle gameplay often requires the AI to have limitations or blind spots.

## Dialog

Although many games involve dialog, dialog as a game mechanic has been most extensively explored in interactive fiction (Short, 2011) and visual novels (Cavallaro, 2009).[1]

### Question answering

Dialog is most often used in games for providing information (clues, quests, item locations, etc.) to the player. Generative question answering is of obvious use here, and so genres and mechanics that emphasize information gathering are good candidates for this kind of AI.

The most obvious example here would be the detective genre (hard-boiled or otherwise), which has been explored in non-AI-based gameplay in games from *Deadline* (Blank, 1982) to *L.A. Noir* (Team Bondi, 2011). The challenge lies in finding ways of adapting these from the ASK/TELL dialog interfaces of parser-based IF and the dialog trees of titles like *L.A. Noir*, to the broader range of questions that can be asked with a generative system. And again, it will be necessary to design around the limitations of the technology; it's unlikely that a near-term generative question answering system will be able to handle a player input like "Aha! But you said that Colonel Mustard was reading *50 Shades of Grey* when you last saw him, but we now know that his monocle was smashed during the elephant stampede! I put it to you that you are lying! What say you sir?"

### Affinity, trust building, and rapport

Dialog can also be used to build relationships with characters. Character relationships are often the central goals of visual novels, with more sophisticated games having affinity systems for tracking the NPC's attitudes toward the player character. The player has an explicit goal of developing rapport with one or more NPCs, often in order to have a romantic relationship, but sometimes in order that they will confide in the PC. Similar affinity systems, with different goals, can be found in systems like *Façade* (Mateas & Stern, 2005), *Prom Week* (McCoy et al., 2012) and *Versu* (Evans & Short, 2014).

[1] Although visual novels, save perhaps for kinetic novels, are technically interactive fiction, I'm treating these separately, since they evolved independently and have very different themes, tropes, mechanics, and demographics.

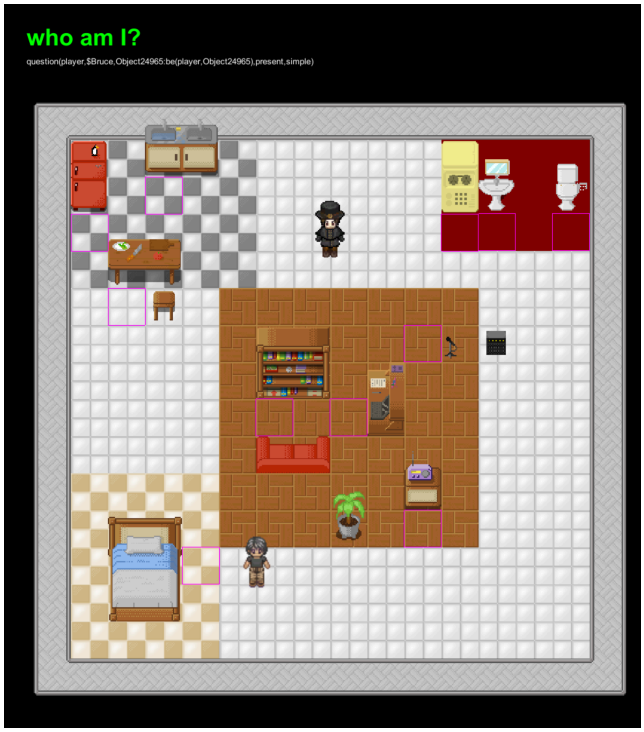


**Figure 1: Screenshot of debug level**

## Gameplay Sketch

I'm currently working on a game that explores mechanics for classical AI (see figure). A mystery thriller, its plot is based on the premise that the CIA's mind control experiments of the 50s, 60s, and 70s were successful; it's working title, *MKULTRA*, is the name of the best known of those programs (United States Congress, 1977). The mystery and detection aspects of the game will follow Laws' (2013) structure of scenes with core clues and peripheral clues. The game's core mechanics will primarily involve dialog and mind control. The game is visually styled as an old-school tile-based RPG (hopefully with better art than that above), but with gameplay closer to a text adventure.

## Dialog

The player is positioned as a kind of super-ego for the player character (PC). The player primarily interacts by typing English text, which is treated as part of the PC's internal dialog.

The player character AI's response to player input depends on the type of speech act and the context. Questions directed to the PC are answered by the PC, and are thought of as part of the PC's stream of consciousness: "Who's that?" "Oh yea, that's Bennie; he owns this bar.

He's kind of a jerk." This provides a mechanism for delivering backstory to the player, without resorting to narrative info-dumps or NPCs whose primary purpose is to be walking encyclopedias.

Most other inputs, be they imperatives or dialog directed to other characters, are treated as *advice* for the PC's action selection mechanism (see below). The player does not directly pilot the PC, although she has considerable control over its actions.

**User interface issues**

Game interfaces based on typed natural language face a number of difficulties. One major issue is that the system will inevitably understand only a tiny fraction of the player's grammar and lexicon. The player needs to learn this subset as quickly and painlessly as possible, lest the gameplay devolve into an endless series of "I don't understand" messages as the user plays "hunt the verb." *MKULTRA* attempts to mitigate this issue by using a bidirectional grammar. This allows the system to parse partial inputs from the user and generate randomized completions that form valid utterances given the system's lexicon and grammar.

Each time the player completes a word, the system solves for a completion. If no completion exists, their input is displayed in red and the player will be unable to type further until they delete back to a point from which completion is possible. If a completion does exist, their input turns green and the completion is displayed in grey italics. The player can then hit return to accept that input, or continue typing.

Another issue is the mismatch between typing speed and normal human conversation. In real-time games supporting typed English input, NPCs often time out before the player has finished typing. The NPC then gets further confused when the player hits return, since their utterance is no longer relevant. I do not have a good solution for this. *MKULTRA*'s underlying game world runs in continuous time, so strict turn-based input is not practical. The current system pauses the game simulation whenever the player starts typing, but it might be preferable to have it slow the simulation rather than stop it.

**Conversation with NPCs**

This is the least-well worked out part of the design. From a gameplay standpoint, NPC conversation will be used primarily for information gathering. From a mechanical standpoint the dialog system be a hybrid of a question answering system and a standard IF topics-and-quips system (Short, 2011), but with fancier indexing so that the player can ask questions in a more open-ended manner.

The intent is that NPCs choose answers using the general goal-directed problem solving mechanism used for other purposes, and that they therefore be able to lie or evade. This would then allow the use of a trust-building mechanic where the player builds trust with the NPC (or simply deceives or mind controls them) so as to obtain information the character would not otherwise give.

Given the setting of the game, it is tempting to populate it with a large number of paranoid characters (schizophrenics, cult members, white supremacists, *etc*.), both because these would provide a clear basis for distrust/trust mechanics, and because it's relatively easy to write AI characters that humans accept as paranoid (Colby, 1975).

## Advice, willpower, and ego depletion

Characters use utility-based action selection, similar to *Versu* (Evans & Short, 2014); one set of rules propose actions while another scores them for utility. Advice to the PC is not automatically followed, but rather increases the utility of the action. If that utility is insufficient to make it the maximum utility action, the advice is ignored. The player will be able to increase the utility further by adding exclamation points to their command to simulate using more willpower. However, willpower is a limited resource, and when it is exhausted, the PC will run open loop until its willpower has recharged. (This is a real psychological phenomenon in humans, referred to as ego depletion (Vohs et al., 2008)).

## Mind control

The player will also gradually develop the ability to control the minds of NPCs. The player will not be able to directly joystick or otherwise control the actions of the NPC as in *Stubbs the Zombie* (Wideload, 2005), but rather they will be able to inject *beliefs* into the heads of NPCs. Players must then back-solve for what beliefs will lead to the desired NPC behavior, so this is an AI-as-subordinate-and-puzzle mechanic. It effectively makes AI debugging the problem of the player and provides a ready narrative alibi for dysfunctional behavior: *of course they're stupid, they're mind controlled zombies!*

The player will inject beliefs by speaking in a separate, fictitious language (explained within the narrative of the game as being some kind of divine or infernal language) of which the player has only limited knowledge. This provides a mechanism for limiting the player to saying things that the AI will be able to act on reasonably. It also provides a mechanism for leveling up the player by slowly feeding her bits of grammar and lexicon. It is also tempting to make the learning of the language a puzzle in itself, such as in *The Gostak* (Muckenhoupt, 2001).

**Example puzzle**

Suppose the PC needs to get past a guard NPC to obtain access to a computer in the room being guarded. The player can solve the problem by injecting two beliefs in the guard NPC:

- It's time for a shift change
- The PC is another guard

The guard NPC walks away and PC can enter the room without interference.

While this solves the problem, it has the disadvantage that someone walking past the room later will notice that it's unguarded. So the player could do better (e.g. win an achievement or avoid problems later in the game) by injecting these beliefs instead:

- The PC is another guard
- The NPC has to pee really badly

Then the NPC would ask the PC to take over while he went to the bathroom. While the NPC is gone, the PC accesses the computer, and then goes back outside. The guard returns, and the intrusion is never detected.

**Crafting telepathic items**

One attractive property of the mechanic of belief injection through spoken commands is that it can combine with items to achieve interesting effects. A loudspeaker could allow mind control to act as an area effect weapon (a weapons of mass delusion). A tape recorder could be used as a "belief grenade," allowing time delay or greater standoff distance. Combining the tape recorder with a triggering mechanism allows the creation of mental landmines or time bombs, or to allow the player character inject beliefs into herself.

For example, suppose a malevolent organization has constructed a device that broadcasts suicidal thoughts to anyone who comes near it, and the player needs to deactivate it. One particularly amoral solution would be to choose some conveniently disposable NPC, give them explosives, and convince them to approach the device. When the NPC reaches the device, they decide to commit suicide, but since they're already holding explosives, the most convenient way to do so would be to detonate the explosives, thereby taking the device with them.

## Mind reading

Finally, it's necessary to provide feedback to the player about the states of the various NPC AIs. In real life, humans rely on non-verbal behavior such as facial expressions. Unfortunately, this is not practical for a game with RPGMaker-style graphics. While it's tempting to have the characters display English language thought balloons showing their internal thoughts, this is more information than we want the player to have, and more information than the player could assimilate anyway.

An interesting alternative is to use aural displays to simulate telepathy. Mood and other long term state information will be communicated using drones that change pitch, volume, or timbre. In addition, the system can play a short, staccato sound event (e.g. a click or chirp) each time the character considers an option in its decision cycle. Ideally, these would be modulated based on the utility, valence, or other attribute of option. This would allow the player to hear the NPC thinking, even if they don't know the propositional content of the thoughts.

This mechanic is most directly useful for judging whether an NPC is lying – if they make a lot of noise before answering, they're thinking hard about the answer. But puzzles are also possible. An invisible character could be located by the sound of their thoughts or a character impersonating another character (or possibly one who is being mind controlled) might be found out through a change in the sound of their thoughts. It's also an interesting mechanism for debugging during development.

## Implementation status

The initial version of the system infrastructure (world simulation, NLP, problem solver) should be completed by mid-August. Then implementation effort will shift to generating content using placeholder art. My goal is to have a playable demo level ready by October and an alpha release of the game by Spring 2015. The system is open-source to encourage modding, particularly for educational and research purposes.

The system is built on Unity3D (Unity Technologies, 2004). The basic tile system with locomotion code (steering behaviors and path planner) is written in C#, as is the Prolog interpreter and the glue code between it and the rest of the game.

The AI is written in Prolog (Clocksin & Mellish, 2003). The interpreter is mostly ISO compliant, with the addition of the freeze/2 and dif/2 predicates for constraint handling. It also contains a number of additions that are useful for game programming: an implementation of eremic logic (Evans, 2010), which has better semantics for state changes than conventional Prolog, the ability to selectively randomize the order of clause execution, and relatively transparent interoperation with C# code.

The English parser-generator handles single-clause sentences or single clauses wrapped in modal verbs. It can parse and generate the standard tenses, aspects, and moods, although tense and aspect are not currently used in any interesting way. It uses a definite-clause grammar (Pereira & Shieber, 1987), with Montague's PTQ semantics (Montague, 1973) to handle quantifiers, although there are not currently any quantifiers in the system's lexicon – I don't yet have a use case for them.

DCGs offer a number of advantages. It's relatively easy to make them bidirectional, so the same code base can both parse and generate. They macro-expand directly into

Prolog code, so they're very easy to implement, and they can also be salted with raw Prolog code to execute during the parsing process. Finally, it's straightforward to implement a quip system by treating quips as additional character-specific grammatical productions. They can be thought of as generalizations of the slotted string mechanisms used elsewhere (Evans & Short, 2014; McCoy et al., 2012; Montfort, 2007; Nelson, 2006).

Problem solving is performed using a number of mechanisms. A rudimentary reactive planner (Bonasso, Firby, Gat, & Kortenkamp, 1997; Mateas & Stern, 2002) provides basic support for event handling and utility-based action selection. Within this framework, different mechanisms can be used for proposing and scoring: a grammar-like mechanism is used for proposing actions in ritual exchanges like greetings and partings; the player interface component proposes whatever action the player last proposed and scores it accordingly. Although I have a more traditional reactive planner implemented, I intend to replace it with a trivial subset of NASL. The appeal of this is that it would allow the use of Sibun's (1992) incremental, local discourse planner.

## Related Work

Although most game AI code can be thought of as a combination of finite-state control and combinational logic, there are also examples of the use of more powerful representations, the best-known examples being *Façade* (Mateas & Stern, 2005), *Prom Week* (McCoy et al., 2012), and the various pieces built on the Versu platform (Evans & Short, 2013). These systems use generative AI internally for character control, but provide fixed options for user input, such as menus of possible actions. In the case of *Façade*, players type English text that the system categorizes as one of a fixed set of atomic speech acts.

The opposite case can be found in parser-based interactive fiction (Jackson-Mead & Wheeler, 2011). They support generativity in player input, but have little or no AI per se. More recent IF systems use declarative methods. Inform 7 (Nelson, 2006), although it doesn't directly implement any character AI, does represent world state and behavior declaratively in terms of an internal logical form (facts + rules). And Versu, which very much does implement character AI, uses its own modal logic called eremic logic (Evans, 2010).

Various research IF systems have also been built using more expressive logics. Zafeiropoulos (2008) used an object-oriented version of Prolog to implement a traditional text adventure engine. Koller *et al*. (2004) built an IF engine based on *SHIQ* description logic together with a dependency parser and a Tree-Adjoining Grammar generator.

A number of games use propositional declarative representations like rule systems (Evans, 2009) or STRIPS planners (Orkin, 2002). These systems either don't allow variable binding or use some variant of deictic representation (P. Agre & Chapman, 1987; P. E. Agre, 1988), meaning that variable binding is moved outside the inference engine, allowing inference to be propositional and thereby more efficient (Zubek, 2015). Although not a game per se, it's worth noting that Chapman (1990) used a similar architecture in an AI player for a *Joust* clone that could take advice from a human.

## Conclusion

Classical AI has seen surprisingly little use in contemporary game AI. I believe this is due to a combination of system building/integration issues and, more importantly, a mismatch between classical AI and contemporary game mechanics that makes classical AI ill-suited to current games. While I hope that some of the tools being built for *MKULTRA* will help with the former, I believe the latter can only be solved by the research community developing not only new technologies, but also new mechanics and genres that properly leverage them.

## Acknowledgements

I would like to thank the members of the AI Game Programmers Guild for helpful feedback on the industry state of the art. I'd also like to thank Rob Zubek, Richard Evans, Andrew Fray, Eileen Hollinger, and James Ryan for kind words of encouragement.

## References

Abercrombie, J. (2014). Bringing BioShock Infinite's Elizabeth to Life: An AI Development Postmortem. In *Game Developer's Conference*. San Francisco, CA.

Agre, P., & Chapman, D. (1987). PENGI: An implementation of a theory of activity. *Sixth National Conference on Artificial Intelligence (AAAI-87)*. Seattle, WA: AAAI Press.

Agre, P. E. (1988). *The dynamic structure of everyday life*. Cambridge, MA: MIT Artificial Intelligence Laboratory.

BioWare. (2012). Mass Effect 3.

Blank, M. (1982). Deadline. Infocom.

Bonasso, P., Firby, R. J., Gat, E., & Kortenkamp, D. (1997). Experiences with an Architecture for Intelligent Reactive Agents. *Journal of Theoretical and Experimental Artificial Intelligence*, *9*(2-3).

Cavallaro, D. (2009). *Anime and the Visual Novel: Narrative Structure, Design and Play at the Crossroads of Animation and Computer Games*. Jefferson, NC: McFarland.

Chapman, D. (1990). *Vision, Instruction, and Action*. Massachusetts Institute of Technology.

Clocksin, W. F., & Mellish, C. S. (2003). *Programming in Prolog: Using the ISO Standard* (5th ed.). New York, NY: Springer.

Colby, K. M. (1975). *Artificial Paranoia*. Oxford: Pergamon Press, Ltd.

Cycorp. (1984). Cyc.

Evans, R. (2009). AI Challenges in Sims 3. In *Artificial Intelligence and Interactive Digital Entertainment*. Stanford, CA: AAAI Press.

Evans, R. (2010). Introducing Exclusion Logic as a Deontic Logic. In *Deontic Logic in Computer Science, Proceedings of the 10th International Conference, DEON 2010, Lecture Notes in Computer Science Volume 6181* (pp. 179–195). Fiesole, Italy: Springer.

Evans, R., & Short, E. (2013). Versu. San Francisco, CA: Linden Lab.

Evans, R., & Short, E. (2014). Versu - A Simulationist Storytelling System. *IEEE Transactions on Computational Intelligence and AI in Games*, *6*(2), 113–130.

Hecker, C. (2010). *Game AI is Game Desgin*. Stanford, CA.

Isla, D. (2005). Handling Complexity in the Halo 2 AI. *Game Developer's Conference 2005*. San Francisco, CA, USA: CMP, Inc.

Isla, D. (2013). Third Eye Crime: Building a Stealth Game Around Occupancy Maps. In *Proceedings of the Ninth Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-13)*. Boston, MA: AAAI Press.

Jackson-Mead, K., & Wheeler, J. R. (Eds.). (2011). *IF Theory Reader*. Boston, MA: > Transcript On Press.

Koller, A., Debusmann, R., Gabsdill, M., & Striegnitz, K. (2004). Put my galakmid coin into the dispenser and kick it: Computational Linguistics and Theorem Proving in a Computer Game. *Journal of Logic, Language and Information*, *13*(2), 187–206.

Laws, R. (2013). *The Esoterrorists* (2nd ed.). London: Pelgrane Press.

Mateas, M., & Stern, A. (2002). A Behavior Language for Story-Based Agents. *IEEE Intelligent Systems*, *17*(4), 39–47.

Mateas, M., & Stern, A. (2005). Façade.

McAllester, D., & Rosenblitt, D. (1991). Systematic nonlinear planning. In *Proceedings of the ninth National conference on Artificial intelligence (AAAI-91)* (pp. 634–639).

McCoy, J., Treanor, M., Samuel, B., & Reed, A. A. (2012). Prom Week. Santa Cruz, California: Expressive Inteligence Studio at UC Santa Cruz.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In P. Suppes, J. Moravcsik, & J. Hintikka (Eds.), *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics* (pp. 221–242). Dordrect, NL.

Montfort, N. (2007). *Generating Narrative Variation in Interactive Fiction*. University of Pennsylvania.

Moonshot Games. (2014). *Third Eye Crime*.

Muckenhoupt, C. (2001). The Gostak. IFDB.

Nelson, G. (2006). Inform 7.

Nintendo Corporation. (1986). The Legend of Zelda.

Orkin, J. (2002). Applying Goal-Oriented Planning for Games. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*. Stamford, CT: Cengage Learning.

Pereira, F. C. N., & Shieber, S. (1987). *Prolog and Natural Language Analysis*. Brookline, MA: Microtome Publishing.

Robinson, J. A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, *12*(1), 23–41.

Short, E. (2011). NPC Dialog Systems. (K. Jackson-Mead & J. R. Wheeler, Eds.)*IF Theory Reader*. Boston, MA: > Transcript On Press.

Sibun, P. (1992). *Locally Organized Text Generation*. University of Massachusetts, Amherst.

Team Bondi. (2011). L.A. Noir. New York: Rockstar Games.

United States Congress. (1977). *Project MKUltra, the Central Intelligence Agency's Program of Research into Behavioral Modification. Joint Hearing before the Select Committee on Intelligence and the Subcommittee on Health and Scientific Research of the Committee on Human Resources, Unit*. Washington, DC.

Unity Technologies. (2004). Unity 3D. San Francisco, CA.

Vohs, K. D., Baumeister, R. F., Schmeichel, B. J., Twenge, J. M., Nelson, N. M., & Tice, D. M. (2008). Making choices impairs subsequent self-control: A limited resource account of decision making, self-regulation, and active initiative. *Journal of Personality and Social Psychology*, *94*, 883–898.

Whitehouse, D., Cowling, P., Powley, E., & Rollason, J. (2013). Integrating MCTS with Knowledge-Based Methods to Create Engaging Play in a Commercial Mobile Game. In *Proceedings of the Ninth Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-13)*. Boston, MA: AAAI Press.

Wideload. (2005). Stubbs the Zombie in Rebel Without a Pulse.

Winograd, T. (1972). *Understanding Natural Language*. Academic Press.

Wright, W. (2000). The Sims. MAXIS/Electronic Arts.

Zafeiropoulous, V. (2008). *Adventure Games Implementation Under The Prolog Language*. Mälardalen University.

Zubek, R. (2015). Production Rules Implementation in 1849. In S. Rabin (Ed.), *Game AI Pro 2*. Natick, MA: A K Peters/CRC Press.