Exploring Emergent AI in Video Games

Katie Kutzke

CONTENTS

LIST OF FIGURES

***Abstract*—Video games have traditionally used fairly simple, scripted artificial intelligence. As games are becoming less linear and open-ended, AI practices need to adapt. Multi-agent systems and emergent AI can add extra depth to a game by making characters more autonomous and less predictable. Existing structures and methods can be adapted to create organized and dynamic AI.**

I. INTRODUCTION

Video games nowadays are very advanced pieces of software involving lots of interacting, moving parts. Recent releases boast fast-paced, complex gameplay, wide open worlds for users to explore, and extremely realistic graphics. However, development of artificial intelligence in video games has plateaued over the last ten years or so. Game developers have been challenged to include new and challenging AI because the old methods are not the best fit for newer games.

A multi-agent system is an intelligent system that has many units interacting with each other, such as a video game. Agents are the independent actors in such a system. The term agent is very flexible and can be used in many different ways. However, all agents have the ability to:

- Sense their environment
- Make and act upon goals
- Make decisions based on their observations
- Influence the behavior of other agents

Most modern video games incorporate multi-agent theory in some way, whether it is a strategy game or a first-person action shooter. For example, Grand Theft Auto V (GTA) is an action-adventure game that takes place in a large city. Cities are composed of many different agents, such as people, cars, and wild animals. In GTA, these agents' main purpose is to bring character to the city, making it seem dynamic and lifelike. Another popular multi-agent game genre is real time strategy (RTS). RTS games feature multiple opponents interacting with each other on a map, usually fighting, trading, and other diplomatic actions. Agents in RTS games have many different tasks in order to progress, such as collecting resources, fighting, and strategizing.

While agents have the capability to do many different things, the way developers choose to control them is similar across genres. A game would be chaotic and unmanageable if each agent acted independently all the time. This is why most multi-agent systems have a hierarchical organization of agents. The agents that interact directly with the game map are controlled by higher-level agents called managers. Managers have all the attributes that agents do, but they operate on a larger scale. For example, an agent will observe its direct environment within the game map. The agent manager observes the overall state of the game, sensing the environment, other agents, and actions made by human players. Managers then make decisions and pass them on to low-level agents so they can carry out the needed actions.

The organizational structure of managers and agents is also important for avoiding conflicts and overlap. No agent should be controlled by one manager. If agents are kept simple and focused enough, overlap in goals with other agents will be completely avoided.

One example of distinct manager-agent interaction is Starcraft: Brood War. This is a RTS game that features three different races fighting against each other in a space battle. The primary controller for the computer is the strategy manager. The strategy manager analyzes the current situation overall and prioritizes actions based on that. If an enemy has produced many combat units, the strategy manager will queue production for itself. If the civilization is running low on resources, it will send agents to go find and retrieve them. The strategy manager in Starcraft does not directly interact with units. The

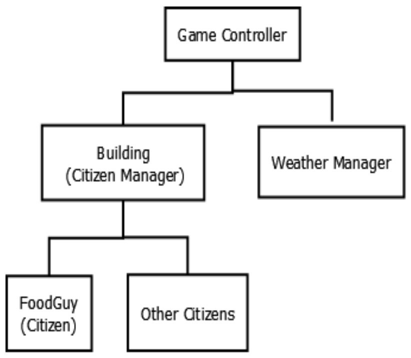

Fig. 1. A simple hierarchical structure showing managers and lower level agents. All figures created by Katie Kutzke

main directive of the strategy manager is to observe the game on a global scale and pass that information to sub-managers and agents. Each sub-manager focuses on a different task–for example, the combat manager manages combat agents and strategy, while the production manager deals with producing new agents [10].

There are many other modern games that feature multi-agent qualities. The ones that will be discussed in this paper are:

- Bioshock Infinite: An action-adventure puzzle game. The protagonist has to escort Elizabeth around the world while fighting enemies.
- Grand Theft Auto V: An action-adventure game that takes place in a big city. The player can do quests and explore the world while making money and fighting enemies.
- Starcraft: Brood War: A real-time strategy game in which players fight for resources, land, and power. This game takes place on a grid world and has lots of intelligent agents and managers.
- The Sims: A life simulator game in which users can create Sims and set them off to live in a neighborhood with others.
- Sim City (2014): A recreation of the classic city simulator using a multi-agent engine called Glass Box. Citizens live in the city and have homes, go to work, and demand entertainment. The player is an omnipotent city planner who can choose how the town will be laid out.

A. *The Land*

My demonstration, The Land, is a small multi-agent simulator. This simulation is based on something called Sugarscape. Sugarscape features agents that can find food, reproduce, and die. Each game turn, an agent will either eat food, reproduce, or move to a different spot. The agents are very self-centered and only care about their current state and their adjacent spaces. The Land is an expansion on that project, including some more advanced AI techniques. The agents, FoodGuys, explore the land, harvest food, and plant farms. There are agent managers that oversee the condition of the game and change the FoodGuys' goals. The Land also features a dynamic landscape. If the FoodGuys harvest all the food, the land will react and make it rain. If the FoodGuys are using up too much land, the land will cause a drought and the farms will die.

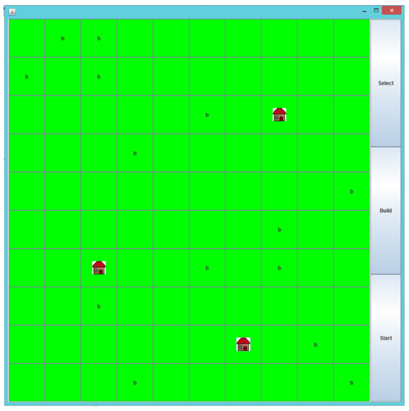

Fig. 2. The Land game board with some buildings and food spots (marked with a b).

II. AI Structures

One of the first games that had sophisticated opponent AI was Space Invaders. Space invaders was the first game that had reactive enemies–they moved around the screen relative to the player and shot back after being attacked. As the game progresses, the difficulty increases [1]. Pac Man is another early adopter of AI. Pac Man's AI uses states rather than immediately reactive enemies. Each ghost enemy has 3 states that lasts for a predetermined number of seconds during each level.

1) Chase state: In this state, the ghosts chase Pac Man around the board. How each ghost chases him is based on a personality algorithm. For example, the red ghost chases after the player directly. To add variety, the orange ghost does not chase after Pac Man directly, but follows a certain spot behind or in front of Pac Man that is determined by how far away they are from each other.
2) Scatter state: In scatter state, the ghosts run away from Pac Man and go back to the center of the map (the ghosts' starting position).
3) Frightened state: This state occurs when Pac Man eats a large dot. When ghosts are frightened, they traverse the map randomly. They turn blue and are able to be eaten by Pac Man, after which they return to the center and wait for the frightened state to wear off.

The strictly organized states and transitions in Pac Man became the standard for scripted AI in many video games.

A. *Scripted AI*

An AI pattern is usually called "scripted" when NPCs have a set number of behaviors that are triggered by certain actions. For example, a trainer in one of the Pokemon games. The trainer starts in a "wait" state in which it waits for the player to walk by. When the trainer sees the player, it enters a battle state. Then, if the computer-controlled trainer wins the battle, it goes back into the wait state. If the player wins, the opposing trainer enters an idle state for the rest of the game. This clear linear progression can be easily modeled with a finite state machine (FSM) as seen in Figure 3.

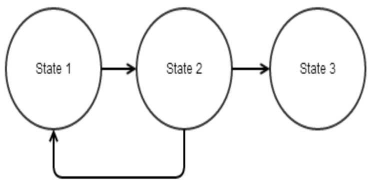

Fig. 3. A finite state machine with 3 states and transitions between them.

The transitions for each state depend on the game. In a first-person shooter, the enemies typically have multiple states, such as pursuing the player, attacking, and fleeing. The transitions in this case could be based on how many health points the enemy has. If it has low health, its strategy will be to flee until it is healthy enough to fight. If it is at full health, it will pursue the player. When it finds the player, it will attack. FSMs are effective for simple decision sequences such as this one, but get very complex as more actions are added [6].

Tree structures are also frequently used to model AI. Trees take the states from an FSM and organize them hierarchically. In a tree, the AI unit starts at the root node and traverses through each available option, checking to see if the conditions to act are met. If they are met, the action is performed [3]. Because of their hierarchical organization, tree structures ensure that certain actions are performed before others (i.e., a child node of the root must be accessed before its child nodes). A tree can be more restrictive than a finite state machine, but is more organized and easier to expand [6].

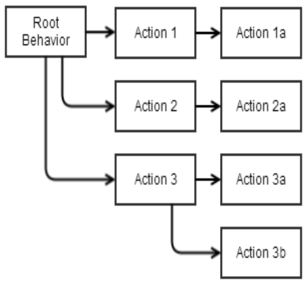

Fig. 4. A tree structure with hierarchical states and chronological layout.

B. Strengths and Weaknesses of Scripted AI

Scripted AI is extremely common in video games because it is relatively easy to implement and get a reliable result. It is easy to plan and control the behavior of in-game characters. Scripted AI is also very easily scaled to large games–the same structure and planning can be used for a variety of NPCs. It is also very effective in simple situations. If an enemy only has to search, attack, and flee, a simple scripted FSM is adequate.

Scripted AI is not the perfect solution to modern video game behaviors. Users often complain that a game is ”too scripted” and that their actions do little to affect how agents behave. For example, the player might be hiding behind a wall, but the agent can see them anyway because they are basing their decisions on how far away the player is. This takes away from the realism and believability of game characters.

Agents that solely rely on 3 or 4 states to act can be very predictable. Once a user figures out what is needed to transition to the next state, they can manipulate that to gain an advantage and easily win the game. If there is little variation in NPC behavior, the game will eventually become boring.

Because of their limited scope, it is hard to scale the difficulty of scripted agents. Rather than making them smarter, lots of developers give agents more strength or health points as the difficulty rises. This works well in some situations, but typically results in the enemy simply taking longer to beat. Alternatively, some scale the difficulty by letting agents ”cheat” and do things that human players cannot. For example, in the Mario Kart franchise (a racing game), opponent cars will miraculously speed up to catch the first place racer, even if they are far behind.

III. Emergent AI

One of the things strictly scripted agents lacks is awareness of the world around them. Even though basic agents can observe their environment, most of their decision-making is based on things that are happening to them. Scripted agents also cannot adapt to new situations. Emergent, or adaptive AI makes agents smarter by letting them base decisions on many factors. This can be done by adding weights to the structures mentioned above. Now, agents are not strictly limited to options available to their current state. They are able to be flexible and seem to adapt their strategies over time.

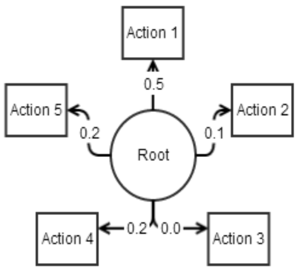

Fig. 5. A weighted AI structure. The weights can change over time to accommodate for changes in the environment or user activity.

One of the main purposes for emergent AI is to create a character that players connect with emotionally [12]. To do this, it makes teammates and characters act as humanlike as possible. Bioshock Infinite features a dynamic computer-controlled companion named Elizabeth. Elizabeth senses the environment, the player’s status, as well as other agents’ conditions. If the player is engaged in combat with a strong enemy, Elizabeth will cheer the player on and heal them, if necessary. If Elizabeth and the player are exploring the world, Elizabeth will notice objects and interact with them or make comments.

Another way to create adaptive agents is by using smart objects. The most notable game that uses smart objects is The Sims franchise. The Sims features many different activities, from eating and using the bathroom to riding roller coasters and falling in love. Rather than programming each agent to be able to perform every single action, Sims (agents) have needs and base their actions on fulfilling those needs. If a Sim is hungry, it will seek out food. To get some food a Sim might have a refrigerator, a grill, or a restaurant available to them. Newer games in the franchise give Sims personalities for more in-depth decision making. So, if the Sim has the lazy trait, maybe he will order a pizza. Smart objects are very easy to expand. Instead of rewriting the entire AI for a Sim to be able to adjust to one new situation, developers can simply make new items and assign the correct needs to them.

A. *Strengths and Weaknesses of Emergent AI*

Emergent AI makes NPCs smarter and more interesting. By giving them a wider variety of actions to perform, agents seem more lifelike and interesting. Adaptive agents are much less predictable than their scripted counterparts, and seem to act more appropriately to outside stimuli. Adaptive agents have the ability to perform differently in certain situations than the developer may have anticipated. This leads to more interesting gameplay.

Real world environments have many stimuli. If an emergent system is to be reliable, agents need to be balanced so they aren't distracted or overstimulated. It is important that agents are reacting to the right things so they can stay on task. For example, Bioshock's Elizabeth is helpful most of the time, but occasionally she will get off track during a battle. Instead of helping the player, she might go discover something interesting (and totally irrelevant to the situation) outside a window.

Emergent AI, like scripted AI, is hard to scale for difficulty. There is a fine line between knowing enough about the environment and being omniscient. If an agent is too limited, then it can't perform well. If an agent knows too much, then it is much less fun for the player. There needs to be a balance to keep the game challenging but still enjoyable.

Performance can also take a hit if the AI isn't optimized properly, or if there is just too much going on at one time. Scripted AI does not take a lot of recalculating, but because emergent agents need to readjust their strategies every step of the game, it has the potential to take a lot of computing power. The recent release of Sim City (2013) uses a multi-agent framework to make the city come to life. Previous renditions of the game were based on statistics–that is, the animations happening in the game weren't necessarily occurring in real time. In the new game, there are lots of calculations going on at one time that put a greater load on the CPU /citeSimCityPerformance.

IV. Future Trends

While the development of video game AI has seemed to plateau in the last 10 years, recent user demand is pushing games to be more realistic and immersive. Open-world, sandbox games let user s progress at their own pace, making many different decisions along the way. However, most AI now is best suited for linear games that follow a predefined script. As the sandbox genre becomes more popular, AI techniques must adapt and grow to keep users interested. Open-world games provide many side quests and characters to interact with, along with a few main storylines. However, most NPCs follow some sort of script: where to be at a certain time, what to say if spoken to, maybe some idle animations to make them look busy. In a sandbox game, these scripts should not be so finely tuned. Players could speak to this character before the main plot leads them there, resulting in irrelevant or inappropriately timed dialogue. On the other hand, players could completely forget to speak to this character, possibly missing some key plot points. Some games have attempted to make their NPCs more dynamic, but there is still a lot of room for improvement. NPCs could have social lives, interacting with fellow computer-controlled characters [11]. They could organize their own missions rather than standing around waiting, or walking in a set loop. Making characters more autonomous and less scripted will bring some much-needed life to video games.

Developers are also working to make AI behavior seem more humanlike. The goal of artificial intelligence in games has always been to make the opponent seem as much like a human as possible. We are sure to see more dynamic, emotional AI in the future after the successes of Elizabeth and Ellie in Bioshock Infinite and The Last of Us. The Sims 4, which will be released in Fall 2014, lets Sims have traits that influence what they do. Another expansion from the old games is the ability for Sims to multitask, emulating real human behavior even more [2].

Another potential future use of emergent AI is the use in virtual or augmented reality. Augmented reality is the use of computer-generated objects overlaid into the real world. For example, Google Glass has the potential to use the glass as a screen, overlaying hotspots or maps with symbols and images. Emergent AI could provide users with an interactive virtual guide that gets to know the user and the area. This companion could show users interesting places, restaurants that they might like, or simply be entertainment.

V. Reflection

This project was mostly inspired by the work I did in Yu Zhang's 230 class last spring. An AI simulation called Sugarscape was the main focus of our project. Sugarscape is a multi-agent system featuring agents that can eat food, reproduce, and die. This project was a great success for my group, so I wanted to further pursue that area of research. My first steps to do that were to do research at St. John's in the summer. However, that experience turned out to be less of a triumph than I initially expected. Instead of focusing on the AI aspect of multi-agent systems, our summer projects were supposed to be more about graph theory and statistical modeling of multi-agent situations. I do not have much of a math background, nor a real interest in in-depth mathy applications, so finding an interesting and doable topic was a struggle. I did however learn a lot more about multi-agent systems and how they work. Because of my interest in video games, I wanted to do a project that applied what I learned during the summer to something I actually enjoy.

Besides the Sugarscape idea, I used coursework from Computer Theory and Algorithms to complete this project. Computer Theory helped in understanding and creating the structures in AI, namely the finite state machines and trees. I used knowledge from Algorithms to explore and implement pathfinding methods in my demo project. Otherwise, I mostly used programming experience that I have used in every class since CS150. The Land was coded in Java using the Swing libraries, which were used in CS160, CS200 and CS230. To organize the agents and managers, knowledge from CS200, data structures, was utilized.

On the other hand, researching this project has helped me use my previously isolated course experience into one product. Rather than only knowing the theory of state machines, I actually programmed some into an integrated system. It has been interesting making a program that uses concepts from all my classes, rather than focusing on one or two course-specific topics.

VI. Conclusion

As non-linear, open-world games become more popular, developers' AI techniques must change. Traditional, purely scripted techniques are no longer appropriate for such large scale, living worlds. It is very limiting in games where the player has a lot of freedom, and can make NPC behavior stiff and predictable. Using concepts from emergent AI and multi-agent systems, video game worlds can become more lifelike and dynamic.

Much of AI is giving human players a convincing illusion that they are not playing against a computer. While many behaviors are obviously not human, building AI so it is both logical and imperfect makes it seem less robotic, yet still fair. Structuring the agents and managers in an organized way enables agents to pursue common goals rather than running around at random. Keeping AI organized into agents and managers is also important for expandability. While The Land only has two types of agent and one manager, adding more is not difficult if the hierarchical structure is kept intact.

Emergent AI, where the agents can adapt to new situations, is a growing idea in game development. Having many stimuli to react to rather than only the character's personal state greatly enhances its realism and flexibility. There will be many more dynamic, adaptive characters in video games in the near future. Learning about adaptive, emergent AI also has implications for non-gaming topics. For example, having AI that can accurately react to a human's actions could be beneficial for social networking and virtual reality.

References

[1] Electronic games magazine, May 1982.

[2] Megan Farokhmanesh. Why multitasking in the sims 4 makes characters feel 'more believable'.

[3] Bjoern Knafla. Introduction to behavior trees, March 2011.

[4] Eugene F. Grant Lardner and Rex. The new yorker.

[5] Patrick Lester. A* pathfinding for beginners, 2005.

[6] Dave Mark. Ai architectures: A culinary guide, November 2012.

[7] Sergio Ocio and Jose Antonio Lopez Brugos. Multi-agent systems and sandbox games, 2009.

[8] Jamey Pittman. The pacman dossier, June 2011.

[9] Steve Rabin. Artificial intelligence: Agents, architecture, and techniques.

[10] Firas Sadafi and Damien Ernst. Organization in ai design for real-time strategy games, 2013.

[11] Edge staff. The future of videogame ai.

[12] et al. Umarov, Iksander. Believable and effective ai agents in virtual worlds: Current state and future perspectives.

[13] Steven Walton. Sim city performance, benchmarked.

[14] Georgios N Yannakakis. Game ai revisited, 2012.