5-2019

Developing an Artificial Intelligence for Grant Anon: A Real-Time Strategy Game

Tristan Martin
trmartin@unomaha.edu

Developing an Artificial Intelligence for Grant Anon:

A Real-Time Strategy Game

University Honors Program Thesis/Capstone/Creative Project

University of Nebraska at Omaha

Tristan Martin

May 2019

Harvey Siy, Ph.D

Abstract

Artificial intelligence (AI) is a necessary facet for almost any modern-day game. Whether this be in the form of artificially controlled enemy player(s), obstacles/challenges in the environment, or randomness throughout the game: AI can make games more fun and challenging through added complexity. Through the development of Grant Anon—a real time strategy game where up to four players control competing colleges fighting to collect grants—it became apparent that AI would be necessary due to the difficulty of gathering multiple people in the same room at the same time. The implementation of an artificial intelligence that responds to game stimuli in the way a real player would is not a trivial task, however. To simplify this process, the AI for Grant Anon was broken down into multiple stages and developed in iterations. This way, the AI could progress to be more player-like over time without tackling a huge task at once. Ultimately, this process led to an AI that presented a very difficult (yet not impossible) spin on the game, which accomplished the goal of added complexity.

Table of Contents

Introduction 3

Introduction to Real-Time Strategy (RTS) 3

Introduction to Local / Split Screen Multiplayer 5

Introduction to Artificial Intelligence (AI) 7

Introduction to Grant Anon 9

Game Mechanics Necessary for AI 12

Pathfinding 13

Grant Collection 13

Grid Translation 14

Developing the AI 14

Task 0: Setup 14

Task 1: Move Units to the Grant on Spawn 15

Task 2: Randomly Spawn Either Unit or Building 16

Task 3: Intelligently Move Units Towards Grant 17

Conclusion 17

References 19

Appendix 21

Developing an Artificial Intelligence for Grant Anon: A Real-Time Strategy Game

Grant Anon is a real-time strategy, local multiplayer game in which up to four players compete to collect grants. The player that collects three grants first wins. This project aimed to develop an artificial player that would play the game like a real player, adding additional complexity to the game as well as the ability for a single player to play the game. Each of the below sections elaborates on these core topics, before moving into the development of the game and AI itself.

Introduction to Real-Time Strategy (RTS)

Before understanding what a real-time strategy (RTS) game is, it can be useful to compare it to the age-old turn-based strategy (TBS) games. Turn-based strategy games have been popular well before the advent of computers: featuring classics such as chess and checkers (as well as countless other board games). However, there are plenty of digital examples of TBS games as well, such as the popular Total War series and Civilization series which are both found on PC. In the TBS game genre, players compete against each other to achieve certain goals in a round-robin fashion. This presents an interesting obstacle for players as they must make decisions that will not only benefit them in their turn but also strategize against what the opponent(s) may do in their turn. Ultimately, each player's goal is to achieve one or more victory conditions—conditions that constitute a player winning the game—such as check mate in chess or no more pieces in checkers. One additional distinction that differentiates TBS and RTS games from other genres such as first-person shooters is that players act as a commander of a set of units or resources instead of controlling an individual character.

Unlike TBS games, real-time strategy games are played in real-time, meaning that all players are simultaneously working to complete objectives in order to win the game. While this is the only real distinction between RTS and TBS games, the nature of an RTS game often lends itself to involve one key attribute that is not necessarily found in TBS games: an economy.

Because RTS games are played in real-time, an economy allows players to strategically work towards completing victory conditions in different ways. For example, in an RTS game where each player commands an army, each player may be awarded a certain amount of resources at a set time interval which can be used to buy additional units or buildings that could be used to fight other armies. With an economy, players have to strategically use their resources to pursue victory conditions while deterring other players from doing the same.

Richard Moss notes in his article "Build, Gather, Brawl, Repeat: The History of Real-Time Strategy Games" that the creation of the RTS genre dates back to the Intellivision game Utopia, which was created in 1982. He states that the term 'RTS' wasn't coined, however, until the release of Dune II in 1992 when creator Brett Sperry "coined the term while attempting to describe the game".

Following the release of Utopia in 1982, the RTS genre saw a boom across the board until 2001 where PC sales began to fall every year since (Moss, 2017). Moss states that this is likely due to the inherent complexity and learning curve found in RTS games. Edwin Evans-Thirlwell of PCGamer supports Moss's claim through something he calls 'the MOBA effect' in his article "The Decline, Evolution, and Future of the RTS". Evans-Thirlwell claims that MOBA games, a genre in which players control a single hero unit (as opposed to an entire economy in RTS games), occupies the same space as RTS and has slowly taken players away due to the smaller learning curve found in MOBA games (Evans-Thirlwell, 2016).

Both authors are well-grounded in their hypotheses, as no new RTS games are being released in the near future from AAA game studios (which represent the largest publishers) (Moss, 2017). While this doesn't mean the end for the RTS genre, it is telling of troubling times. Despite this, games such as Starcraft II still show that there is demand in the market, as they still have a sizeable share of the RTS market and remain popular despite being released in 2010 (Evans-Thirlwell, 2016).

Introduction to Local / Split Screen Multiplayer

Local (split screen) multiplayer is a genre of game in which multiple people connect to the same video game system and play together. This differs from online multiplayer, in which each player connects to a different game system and plays together via an internet connection. The first local multiplayer game released was Pong, in which two players controlled a ping-pong paddle and attempted to score on the other player (see Figure 1) (Forehand et al., n.d.).

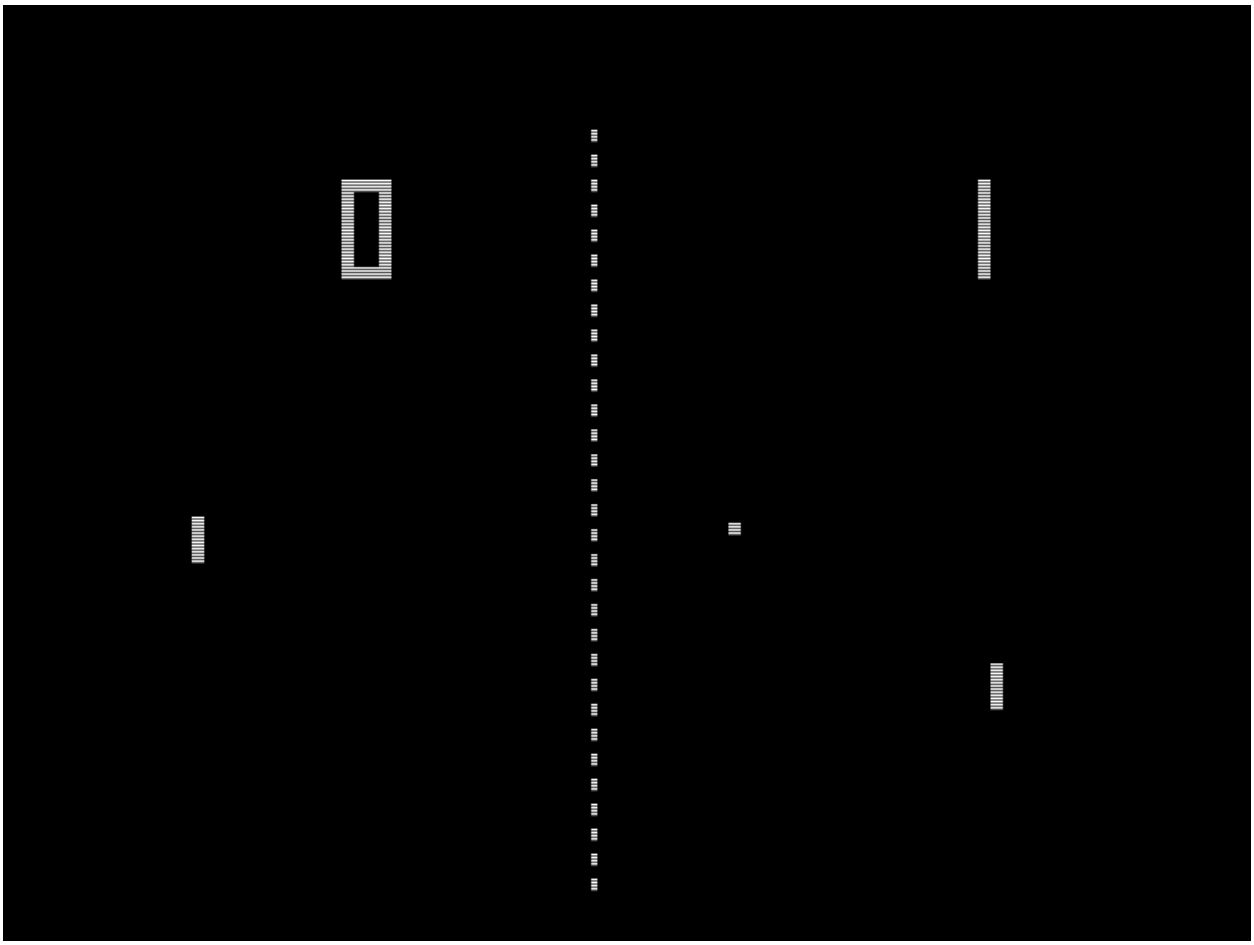

Figure 1. Screenshot of PONG from the Atari Arcade Hits #1 Software Title Released Hasbro Interactive – A Conversion of the Original 1972 Atari Pong [Digital image]. (2006). Retrieved from https://commons.wikimedia.org/wiki/File:Pong.png

Many local multiplayer games have released since Pong in 1972. Unfortunately, a number of reasons have led to the decline of the local multiplayer genre since the early 2000's. The first reason is the increased popularity of online multiplayer, which allows people to play together without having to be in the same physical room (Fuller, n.d.). In fact, it is possible to play against others on entirely different continents, meaning that players can connect from much further distances. On top of this, the number of households that have video game consoles or computers has steadily increased over the past three decades, which supplements the increased

popularity of online multiplayer (Fuller, n.d.). Developers also benefit from not implementing local multiplayer, as they save the cost of development for local multiplayer and can sell more copies if only one player can play per console.

Despite the movement against local multiplayer, Nintendo has shown that there is still demand in the market. Roughly 45% of Nintendo Switch owners (Nintendo's newest console) bought the local multiplayer game Mario Kart 8 Deluxe in just three days following its release (Rosenberg, 2017). This shows that even though local multiplayer is on the decline, there are still a large amount of people willing to invest in quality titles.

Introduction to Artificial Intelligence (AI)

Artificial intelligence is "'the study or design of intelligent agents' where an intelligent agent is a system that perceives its environment and takes actions which maximizes its chances of success" ("Artificial Intelligence", n.d.). To break this down into more concrete terms, AI can sense stimuli in its environment and processes the stimuli to make decisions which maximize its chances for success in given scenarios. Figure 2 is a visual representation of this concept.

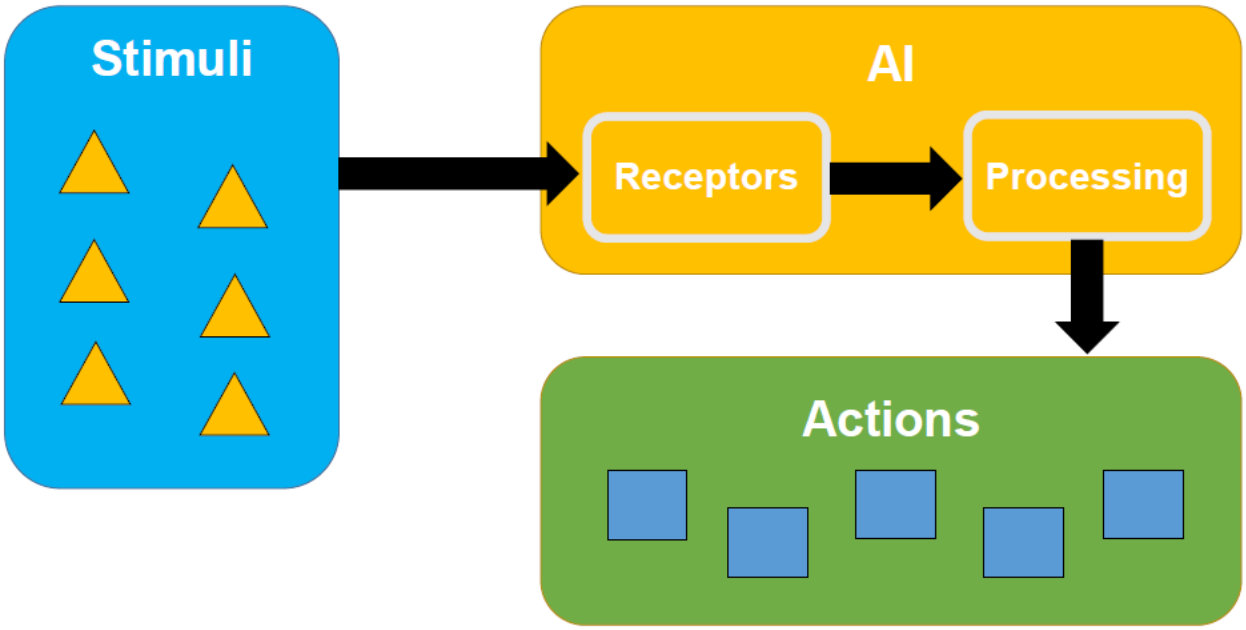

Figure 2. Visual representation of artificial intelligence.

In terms of video games, however, this definition can be more specific. Harbing Lou, a graduate student at Harvard University, states that "the most common role for AI in video games is controlling non-player characters (NPCs)" (Lou, 2017). She continues by discussing the Finite State Machine (FSM) algorithm, an algorithm used to generalize all possible scenarios that an AI could encounter and then specifying specific reactions for each of those scenarios. While predictable, the AI is guaranteed to work towards success in the game (Lou, 2017).

However, adding randomness to an FSM would make the AI player much less predictable. In terms of the AI for Grant Anon (which will be discussed later), this is the approach that was used.

Additionally, other approaches for AI in games, specifically for RTS games, may make a more realistic AI at the cost of extra implementation time. Glen Robertson and Ian Watson offer a multitude of possible AI implementations in their paper "A Review of Real-Time Strategy

Game AI". This list includes, but is not limited to, reinforced learning, game-tree search, behavior trees, and goal-driven autonomy (Roberts and Watson, 2014). While these methods weren't implemented in Grant Anon's AI, they may be worth noting for future improvements or iterations.

Introduction to Grant Anon

Grant Anon is both an RTS and local multiplayer game. Because of the steady decline of both RTS and local multiplayer games as stated previously, it is likely that very few titles fit within the crossover of both genres. This is part of the reason we created Grant Anon, as it allows us to explore and innovate in a relatively niche space. A link to the source code for Grant Anon can be found in the Appendix section of the paper.

In Grant Anon, each player controls a college competing to collect grants. The first player that collects three grants wins the game. Because it is a competition, a minimum of two players is required to play the game, with a maximum of four. Figure 3 gives an introductory look into Grant Anon itself.

Figure 3. Screenshot of Grant Anon. The main base can be seen in the screenshot (building with tentacles around it) as well as the map (grass, rocks, and lava on the screen).

Because Grant Anon allows up to four players to play on a single screen, it is necessary to divide the screen as more players join. Therefore, when two players join, the screen is split horizontally (where player one gets the top half of the screen and player two gets the bottom half of the screen) and when there are three or four players the screen is split both horizontally and vertically (where each player gets a separate quadrant of the screen).

The economy in Grant Anon uses crystals as currency. Each player starts with zero crystals and accrues five crystals every ten seconds. Players have the option to buy two items with their crystals, units (which cost five crystals each) or income buildings (which cost ten crystals each). A player can use their units to destroy the units and income buildings of other players as well as to collect grants. Income buildings generate an additional one crystal every

five seconds, meaning that players forfeit money in the short term for more money in the long term. Each player also starts with a hero unit, which is the same as other units except when it dies it respawns at the main base. When players purchase a unit, it spawns at their main base. When players purchase an income building, it spawns where their cursor is. Figure 4 shows what each of the elements looks like in game.

Figure 4. Overview of Grant Anon main elements.

Grants in Grant Anon spawn every fifteen seconds in the center of the map. To collect a grant, a player must have their units close to the grant for three seconds without any other player's units being in the vicinity of the grant. If another player enters the vicinity of the grant, then the three second timer pauses until the other player's units have left or been killed. Figure 5 shows how players can collect grants.

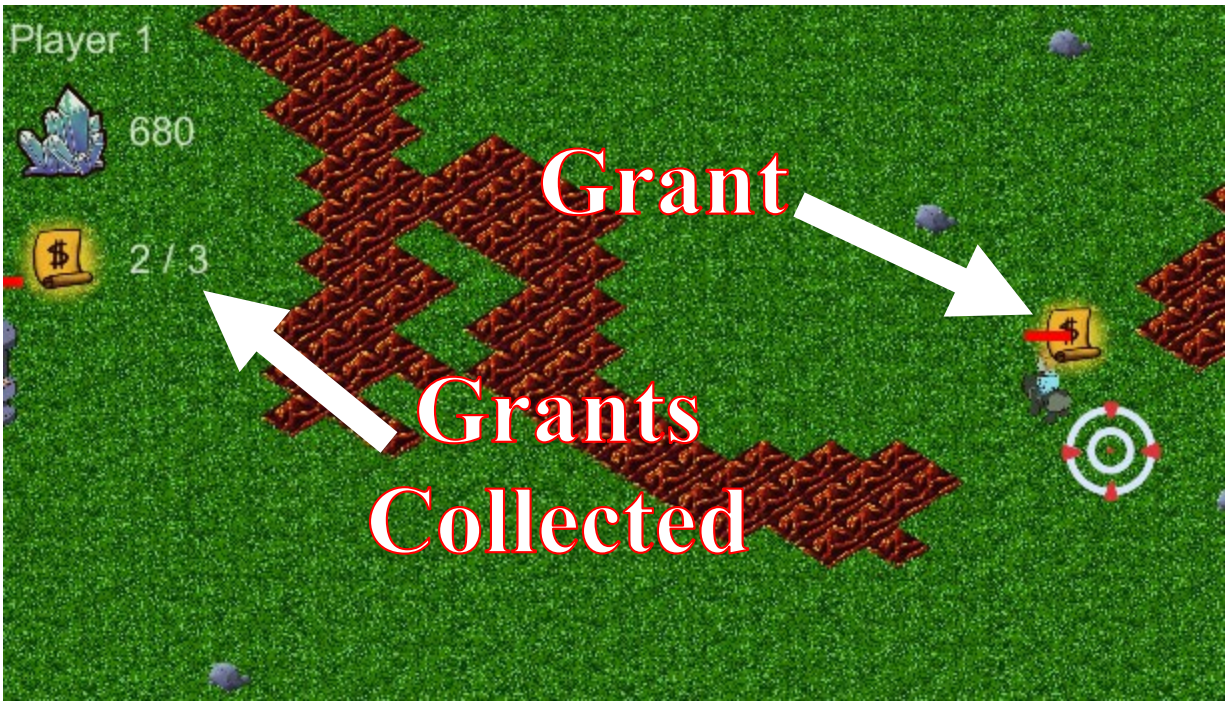

Figure 5. Grant collection in Grant Anon.

Each unit and income building have a limited amount of health in Grant Anon (indicated by the red health bar above them). When two units are within one tile of each other they begin attacking each other until one (or both) dies. When a unit is within one tile of an income building it will begin attacking the income building, and income buildings cannot defend themselves. When a unit is over lava (the red tiles on the map), its health will drain over time until it leaves the lava tile or is killed.

Game Mechanics Necessary for AI

Before moving on to how the AI was developed, it is important to first understand the game mechanics that were necessary for the AI to work. Pathfinding, grant collection, and grid translation are all fundamental parts of how the AI function (although not necessarily part of the AI development itself).

Pathfinding

Pathfinding in a game is determining a path between two points without any obstacles. However, it is often desirable to determine the *best* path between two points, which is a much more complex problem to solve. Fortunately, there are some well known algorithms out there that are optimally designed to calculate the best path. Due to the way Grant Anon was developed, we chose to use the A* pathfinding algorithm, which determines the best path between two points using a heuristic of the more general Dijkstra's pathfinding algorithm (Hart, Nilsson and Raphael, 1968). This paper will not explain how these algorithms are implemented, but it is important to know that a unit in Grant Anon can ask for a path between its current location and its desired destination and the pathfinding algorithm will determine the best path for the unit to follow.

Grant Collection

Grant collection is another fundamental part of Grant Anon. To determine if a unit is within the collection area of a grant, a distance formula is used. The distance between two points, where point one is represented by (x_1, y_1) and the other by (x_2, y_2), is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. If the distance is less than the capture distance, then the unit is in the collection area. Figure 6 is a visual representation of what this looks like.

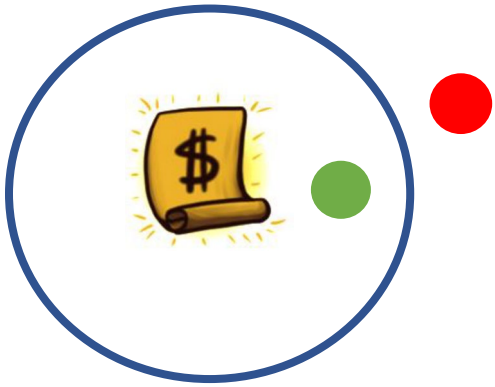

Figure 6. The blue circle represents collection area around the grant. The red dot represents a unit outside of the collection area. The green dot represents a unit within the collection area.

Grid Translation

Figure 3 shows that the map is rotated 45°, which is called an isometric perspective. The benefit of the isometric perspective is that it simulates a 3d environment despite all assets being in 2d. However, particularly in terms of pathfinding, some additional math needs to be done in order to translate an isometric grid coordinate to a regular grid coordinate. This is necessary because the pathfinding algorithm requires that all coordinates be regular (and not isometric). The following formula was used to convert an isometric 2d grid coordinate to a regular 2d grid coordinate: (regular_x, regular_y) = (isometric_x / tile_width + isometric_y / tile_height, isometric_y / tile_height – isometric_x / tile_width).

Developing the AI

With the basic understanding of the necessary game mechanics for AI, we can begin to explore how the AI was developed. The AI for Grant Anon was broken down into four parts, each building upon the previous to reach the end goal of a semi-intelligent opponent that is capable of winning the game. The four iterations of the AI were setup, moving units towards the grant on spawn, randomly spawning a unit or building based on the AI player's resources, and intelligently moving units towards the grant based on other players' units and locations.

Task 0: Setup

The setup of the AI involved programming a way to enable or disable AI players. As this project is separate from the core game, an entirely different copy of the game with AI players

enabled was created. In the copy of the game with AI enabled, any slot (of the four) that is not occupied by a real player is taken by an AI player. Because of this, it is possible to only play the game with one player in this variant. Likewise, if all four real players are present, the game will simply not spawn any AI players.

Task 1: Move Units to the Grant on Spawn

This was the first iteration of the AI itself. The goal of this was extremely simple; if a grant spawns the AI player should move all its units towards the grant. In this case, the stimulus is the grant itself, the receptor is a piece of code that checks if the grant has spawned, and the processor is a piece of code that tells all the AI units to move toward the grant if it exists. Figure 7 is a visual representation of this implementation.

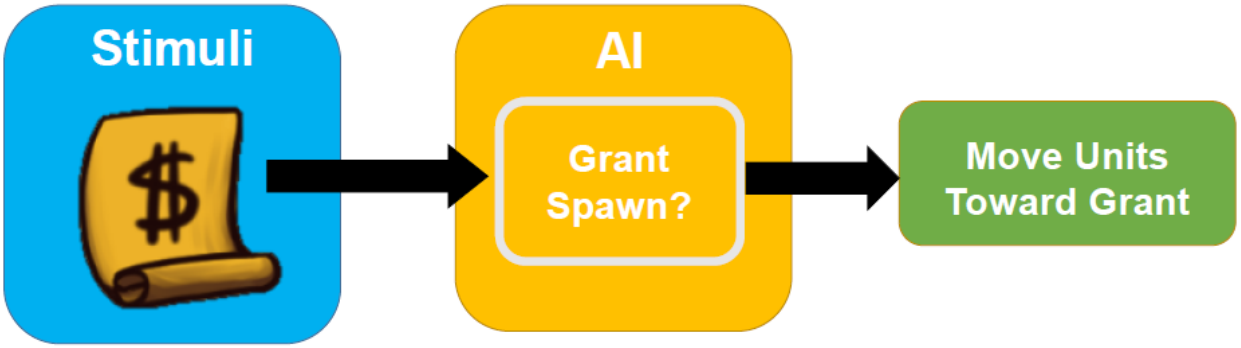

Figure 7. Visual representation of Task 1 of the AI.

In this scenario, it is technically possible for the AI player to win the game, although the AI is extremely disadvantaged because it does not have the ability to create additional units (for more fighting power) or create buildings (for more income). This led to the second task: choosing to spawn a unit or building randomly and waiting for the right amount of resources to accumulate before spawning it.

Task 2: Randomly Spawn Either Unit or Building

This iteration required two different pieces of stimuli: the player's money and a random element that determines whether the player should spawn a unit or building. The first stimuli that the processor considered was the random variable determining whether to spawn a unit or building. After this was determined, the processor then waited for the AI to have the correct amount of resources before spawning in the unit or building. Figure 8 is a visual representation of this implementation. From Figure 8 it is evident that as the number of stimuli increases, the number of possible paths to an output increases exponentially.

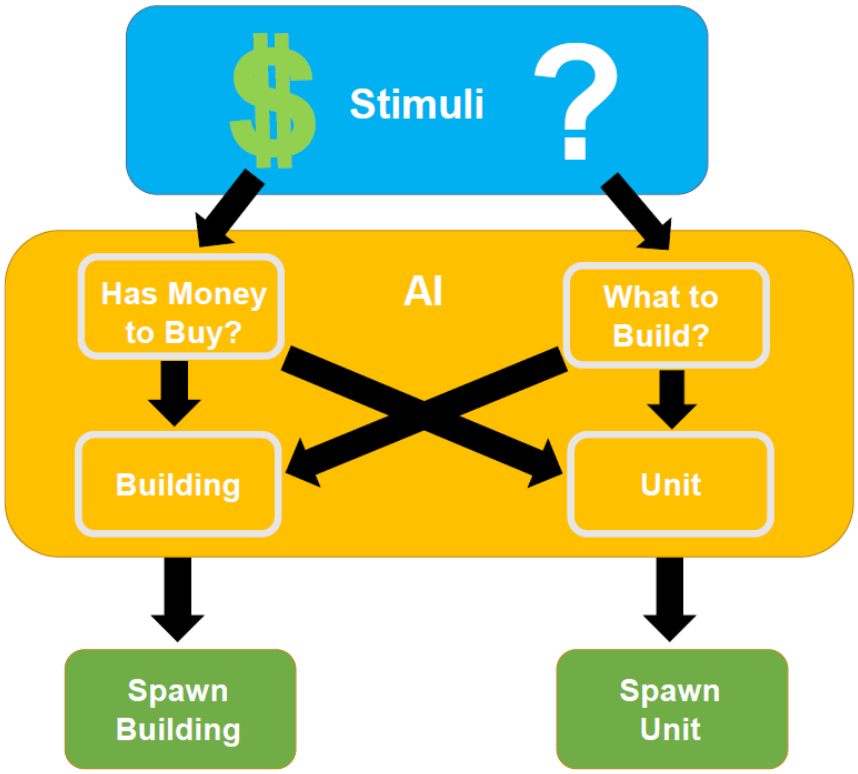

Figure 8. Visual representation of Task 2 of the AI.

This iteration of the AI player made for a much more competitive enemy. However, the AI player would only send one troop at a time to attack other players, still giving it a competitive disadvantage.

Task 3: Intelligently Move Units Towards Grant

The final iteration of the AI player was to have it intelligently move towards the grant. This iteration built on the Task 1 in that it determined when to move units towards the grant. This was accomplished by only having the AI player send troops towards the grant when it had at least one fourth of the total units on the map. This made it possible for the AI player to wait and get more players before attacking if it was weak (or investing heavily in additional resources), making for a far more difficult AI player overall.

Conclusion

Coming into this project, I assumed that the core game mechanics would be easy to implement, and that the AI implementation would be the difficult portion of the project. However, the opposite of both assumptions ended up being true. The core game mechanics, particularly the pathfinding and grid translation, proved to be much more difficult that I initially anticipated. This is due partly due to a misunderstanding of what the implementation of each of these portions would look like and partly due to some of the complexities of the system I was using, which meant that both pathfinding and grid translation had to be made entirely from scratch. On the other hand, the most difficult part of the AI implementation ended up being the core gameplay mechanics and setup, while the actual AI development was much easier than anticipated. However, I attribute a good portion of the ease of development due to prior planning

and setup. When developing the core gameplay mechanics, I ensured that they worked for both real players and AI players which made the development of the AI much easier.

The final version of the AI (after completing the last task) was almost impossible to beat. Because the AI wasn't bound by a controller, it could move units from any point on the map to any other point much faster than a real person could. To add on to this, the AI could use its resources in an optimal way (create buildings and units right after they had enough resources), which would be hard to do for a real player. I think adding a timer onto the AI (in which it couldn't move any units or buy anything) would rectify this issue a bit as it would act more like a real player would.

References

Artificial intelligence. (n.d.). Retrieved from https://www.sciencedaily.com/terms/artificial_intelligence.htm

Evans-Thirlwell, E. (2016, March 11). The Decline, Evolution and Future of the RTS. Retrieved from https://www.pcgamer.com/the-decline-evolution-and-future-of-the-rts/

Forehand, R., Bricker, B., Niederkorn, N., Hagans, J., Freshwater, I., & Grealy, J. (n.d.). Evolution of Multiplayer [PDF]. Ohio State University.

Fuller, S. (n.d.). Topic: Online Gaming Industry. Retrieved from https://www.statista.com/topics/1551/online-gaming/

Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." *IEEE Transactions on Systems Science and Cybernetics* 4, no. 2 (1968): 100-107.

Lou, H. (2017, August 28). AI in Video Games: Toward a More Intelligent Game. Retrieved from http://sitn.hms.harvard.edu/flash/2017/ai-video-games-toward-intelligent-game/

Moss, R. (2017, September 15). Build, Gather, Brawl, Repeat: The History of Real-Time Strategy Games. Retrieved from https://arstechnica.com/gaming/2017/09/build-gather-brawl-repeat-the-history-of-real-time-strategy-games/

Robertson, Glen, and Ian Watson. "A review of real-time strategy game AI." *AI Magazine* 35, no. 4 (2014): 75-104.

Rosenberg, A. (2017, May 01). Nintendo sold 1.2 million copies of 'Mario Kart Deluxe' in three days. Retrieved from https://mashable.com/2017/05/01/mario-kart-8-deluxe-sales-record-breaking/

Appendix

The source code for the AI branch of Grant Anon can be found at https://github.com/tmart2322/GrantAnon/tree/feature/AI. To download the project, go to https://github.com/tmart2322/GrantAnon/releases/tag/2.0 and download the project from there. Those who are familiar with git can also clone the repository.