Artificial Intelligence in Games

CSE 352 : Professor Wasilewska : Team 15

Aaron Lee, Daniel Sha, Colin Monteil-Anatra

Table of Contents

1. Gaming AI, What is it? Why is it important?
2. Strategies for Designing AI in Games
3. Different types of gaming AI
    a. Super Mario
    b. Quake
    c. Starcraft
4. Conclusion

What is Gaming AI and Why is it Important?

- Simulation of Human Intelligence in Non-Playing Characters (NPC)
  - CPU Bots in games such as: Chess, Go, Starcraft, Hearthstone, League of Legends, etc.
  - Interactive Characters in MMORPG: Runescape, Maplestory, World of Warcraft, Tera, etc.
- Machine Learning / Deep Learning in Gaming AI
  - AlphaGo considered a milestone
- Games provide a very good environment for testing and getting feedback
  - Many problems are complicated and can be translated to real word problems
  - In most cases simple to quantify

Creating A.I. for Games

1) Identifying the challenge / task you want your A.I. to embody
    a) Ex: An object to test mastery over skill
        i) Embodied by: goombas, chess bots, etc.
2) Identify how the A.I. should respond to expected input
    a) Ex: Told to win tic-tac-toe with present board
    b) Ex2: Told to move to input coordinate as fast as possible
3) Create A.I. algorithm that can correctly turn input into expected output
    a) Deep Learning* (term normally encompassing neural networks; weighted tree traversals, Markov Chains for scripting)
    b) Pathfinding (A* is the preferred method, to be modified to suite game's needs)

Deep Learning in Video Game A.I.

A) AlphaGo (most famous)

   a) “Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play” - snippet from article by creators of AlphaGo: “Mastering the game of Go with deep neural networks and tree search”

B) Creating a “driving” video games for deep learning A.I. to teach real world object recognition

   a) Adrien Gaidon(Research Scientist at Xerox): “You don’t just generate pixels, you also generate the supervision [AI] requires. … What I’m showing is that the technology is mature enough now to be able to use data from computers to train other computer programs”

C) MineCraft: Potential to be used as real-world training for a.i. (easier as it has a very finite & defined potential of inputs, opposed to real life’s near infinite potential of inputs.)

D) Super Mario (https://www.youtube.com/watch?v=qv6UVOQ0F44 Generation; Species; Genome: based off parent’s fitness score success. Random mutations affect movement through the level & Fitness score increases as mario progresses in level.)

Citation :

Evolutionary Algorithm with Example: Super Mario (https://www.youtube.com/watch?v=qv6UVOQ0F44&feature=youtu.be&t=1m4s)

Source Code: http://pastebin.com/ZZmSNaHX

CS.UTexas paper the code is based off: Evolving Neural Networks through Augmenting Topologies (http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf)

Population & Mutation chance: modifying how you want further generations

Generic Functions: New + Get : Node, Neuron, Genome, etc.

Interesting Functions: `function mutate(genome) , function removeWeakSpecies(), generateNetwork(genome)`

Gaming A.I. Case Study:

id Software

A.I. In Previous First Person Shooter Games

- Early FPS games had very basic A.I. mainly relying on the A* algorithm
- In the 1994 game DOOM, enemy A.I. worked as follows:
  - When an enemy sees or hears a player, it will move in a straight line towards them
  - If the straight path is obstructed, it will move in a random direction for a fixed time

Source: http://mehm.net/blog/wp-content/uploads/2015/02/Doom_Sprites.gif

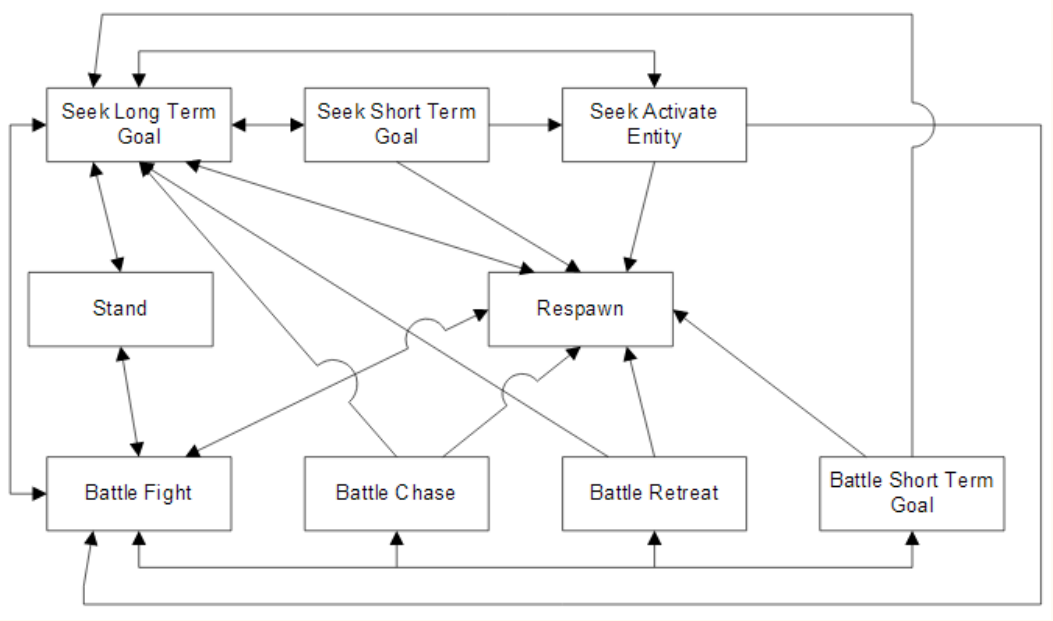
Goals of the A.I. In Quake III (1999)

- Seeing the shortcomings of the A.I. in previous FPS games, id Software set out to create an extremely realistic form of A.I for Quake III
- Their major goals with this project were to make a bot that could:
    - Act like a human player and be hard to distinguish from one
    - Navigate its environment easily and pick up items and use weapons
    - Chats with other players like a human

Source: http://www.ericspitler.com/images/3d/visor_final01.jpg

Quake’s A.I. Network

- This network shows the various states a bot can be in and how it can move between them



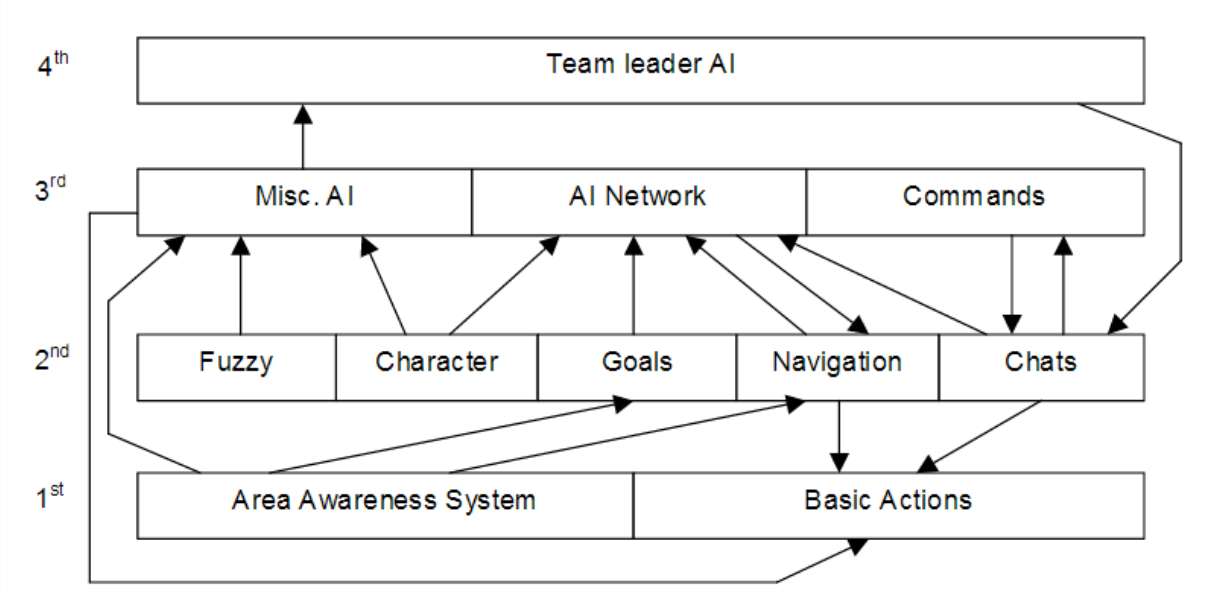
Source http://fd.fabiensanglard.net/quake3/The-Quake-III-Arena-Bot.pdf

Long Term Goals: Capture the flag, kill an enemy, get a specific weapon

Short Term Goals: Pick up ammo on the way to a long term goal. Does not divert bot very much.

Design of A.I. In Quake III (1999)

- The high-level design of Quake's A.I. is made up of 4 layers that interact with each other.



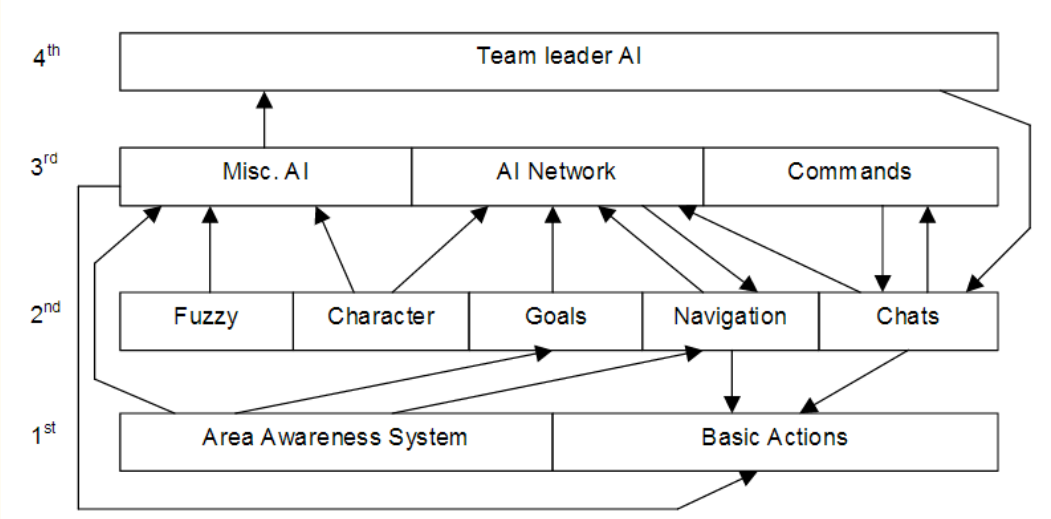
Source http://fd.fabiensanglard.net/quake3/The-Quake-III-Arena-Bot.pdf

Design of A.I. In Quake III (1999)

- **LEVEL 1**

This is the input/output level

The bot sends information about the environment to higher layers and receives movement information from higher layers

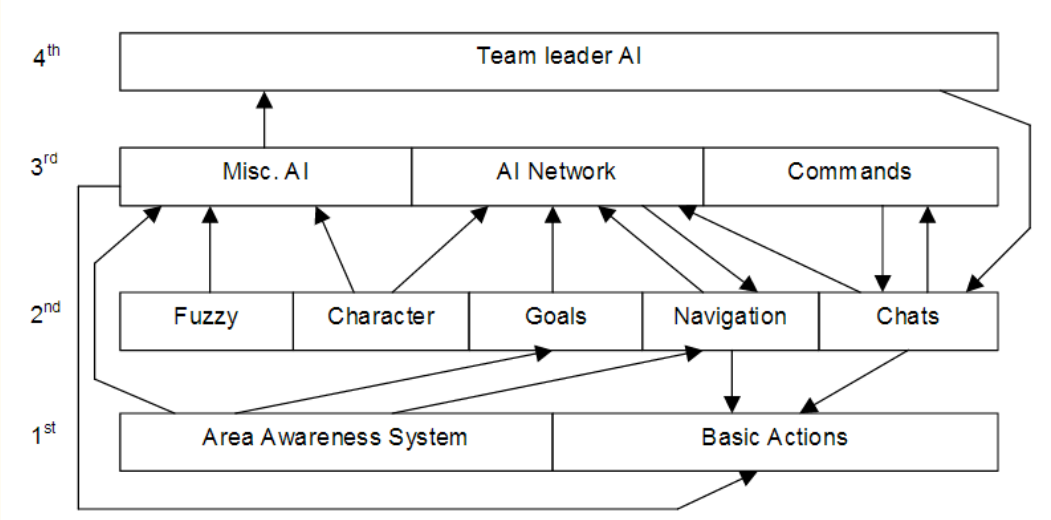


Design of A.I. In Quake III (1999)

- **LEVEL 2**

This is the intelligence layer

Uses **fuzzy logic** to select goals, navigates its environment, and chats with other players



Fuzzy Logic In Quake III

- To act human, there needs to be a way for the bot to set goals and decide how to act on them
- For this, id used **fuzzy logic** as a way for the bot to express how much it wants certain weapons or how important certain actions were to it.
    - The bot can assign a high fuzzy value to retrieving more ammo for a powerful weapon it already has which is low on ammo
- A **genetic algorithm** is used to make sure the fuzzy relations for item preferences are in

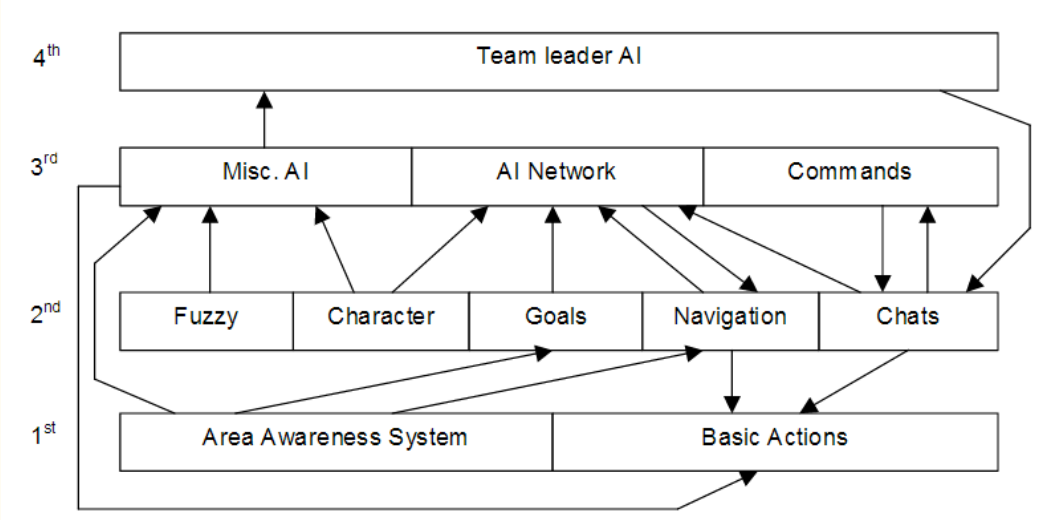
Source: http://i301.photobucket.com/albums/nn51/rager825/QuakeArmory.png

Design of A.I. In Quake III (1999)

- **LEVEL 3**

This is the production rules layer

Has various states describing the situation the bot is in and is able to reason about what actions the bot should take.



Production Rules in Quake III (1999)

- Uses a rule-based system with rules in the form **IF (condition) THEN (action)** just like in procedural programming
    - The condition is a logical expression of facts from the knowledge base
    - The action operates on these facts
- Quake extracted and stored the expertise from human players so that the actions could accurately portray what a person would do in a situation

<u>An example production rule:</u>

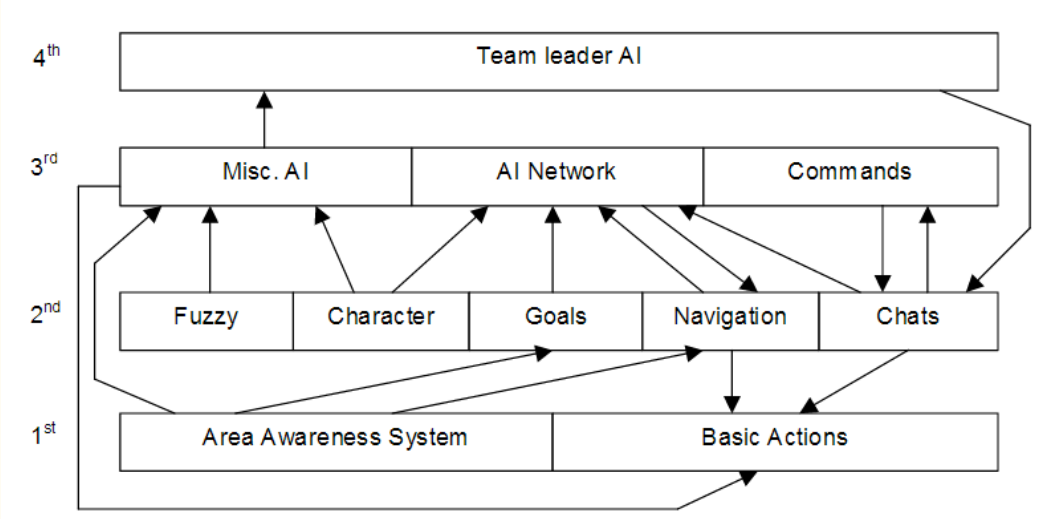
- **IF** bot is fighting **AND** low on health **AND** does not have good weapon **THEN** retreat from the fight

Design of A.I. In Quake III (1999)

- **LEVEL 4**

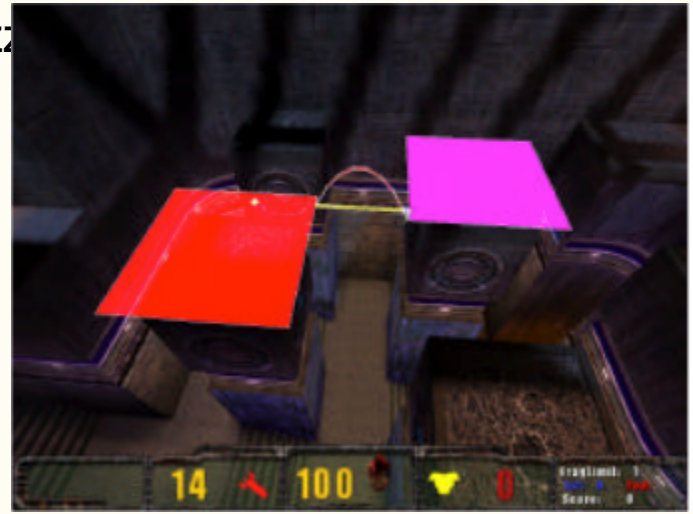
This is the team leader layer

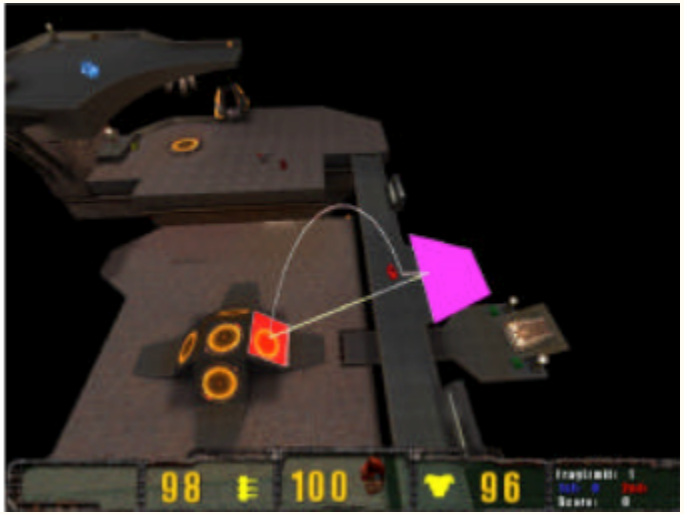
Used for game types where the bot will be a team leader and need to coordinate a strategy to accomplish tasks



Putting it All Together

- All of this allows the bots in Quake to be very lifelike
    - All 3 of the original design goals were met
- Bots can interact with all parts of the environment and even solve simple puzz





Source: http://fd.fabiensanglard.net/quake3/The-Quake-III-Arena-Bot.pdf

What is Starcraft?

- Real Time Strategy (RTS), similar to Warcraft III, developed by Blizzard
- Was the most popular game in E-Sports before League of Legends
- Win Condition: Destroy all Enemy Buildings

https://upload.wikimedia.org/wikipedia/en/2/20/StarCraft_II_-_Box_Art.jpg

Three Races

http://i.stack.imgur.com/LA6AZ.gif

Three Races

http://i.stack.imgur.com/66FDZ.jpg

Larger Army / Unit Composition

http://www.sc2blog.com/wp-content/uploads/2008/06/terran-versus-zerg.jpg

Unit Counters

https://i.ytimg.com/vi/OdTE2QRi8TM/maxresdefault.jpg

Simplistic Starcraft Models

Army control and production

- Bigger / Stronger army wins
- Unit Counters (Air vs Ground)

Economy Maintenance

- Resources require to produce units
- More resources faster unit production, which means larger army

Advantages

Computer Advantages:

- Actions Per Minute (APM)
    - 300-500 vs 1000s
- Perfect Execution
    - Multi-tasking
    - Micro and Macro

gifs.com/gif/korean-gamers-apm-demonstration-8WObO9

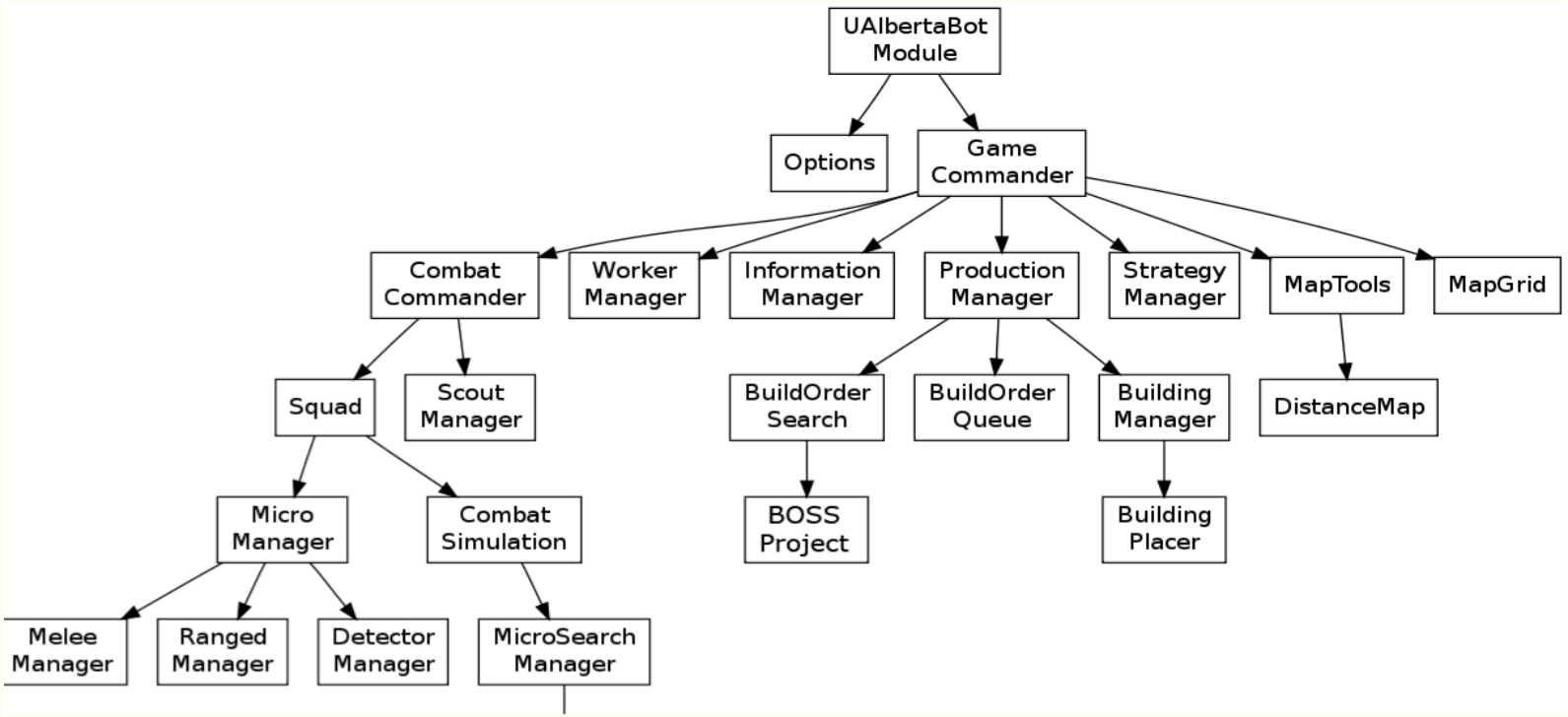
Designing a Solution

1. Immense amount of states:
    a. Chess: 10^50 states
    b. Go: 10^170 states
2. Partially Observable
    a. Fog of War
3. Real Time
    a. Constant changes in states

UAlbertaBot - Open Source

- AI / Bot, written in C++ to play Starcraft Brood War
- Uses the BWAPI: The BroodWar API
- Written by David Churchill, Assistant Professor of Computer Science, at Memorial University of Newfoundland

Design and Architecture



https://github.com/davechurchill/ualbertabot/wiki/images/ualbertabot.png

Logic Flow

CombatCommander.update()

- Scout to find enemy unit, if ready
- If enemy in our region send defence squad, produce units if needed
- If offensive units are available and sufficient in size
  - Attack known enemy base
  - Attack visible units
  - Attack closest enemy buildings
  - Explore until new target is found

Logic Flow

Squads.update()

- Perform CombatSimulation
    - If simulation returns victory then engage
    - Call MicroManager specific to each type of unit
- Else
    - Returns loss then retreat to base

The Reality

```
While (win != true) {
        Action = ask(“How can I create win condition?”);
        execute(action);
}
/* Multiple Win Conditions
  - Economic Advantage
  - Have Stronger Army
  - Outplay / Outmaneuver */
```

Why not Deep Learning?

A general model to win exists for games such as Go, Chess, etc.

- Make moves to maximize the probability of winning

A general model does not exists for RTS

- Multiple models can work
- Hidden Information, fake Information (Deceptions), processing new information
- Humans are able to adapt much quicker and exploit weaknesses

DeepMind and Starcraft II

Blizzard opens Starcraft II to AI and Machine Learning Researchers

- Many subproblems are related to real life problems
- Management of resources for production is very similar to industrial scale production
- Decision making and adaptation under constant state changes with uncertainty closely resembles real world scenarios

https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/

Demo of API Developed With Blizzard

https://www.youtube.com/watch?v=5iZlrBqDYPM

Conclusion

The possibility of AI taking over is still years away

- Humans are capable of understanding a system and exploiting its weakness way quicker than AI
- Human intelligence is not completely reproducible
- AI is extremely power at performing certain tasks
- AI is evolving rapidly

Citations

https://www.wired.com/2016/04/videogames-ai-learning/

http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf

http://doom.wikia.com/wiki/Monster_behavior

http://fd.fabiensanglard.net/quake3/The-Quake-III-Arena-Bot.pdf

http://mehm.net/blog/?p=1424

http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/custom-ai-programs-take-on-top-ranked-humans-in-starcraft

https://github.com/davechurchill/ualbertabot/wiki

https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/