



AI in Video Games

Victor Vargas, Ian King, Iris Wang, and Connor McElroy



Table of Contents

- An Introduction to AI – 3
 - What is AI? – 4
 - A Brief History – 5
 - Common Implementations – 6
- Ethics – 17
 - Why We Need a Code of Ethics – 18
 - IEEE Code of Ethics – 19
- AI Demos – 29
 - Basic Ideas (Pong) – 30
 - Pathfinding – 35
 - Difficulty Balancing – 36
 - Behavior Trees – 38
- References – 40

Introduction



What is AI?

- **Artificial intelligent** or **Agent of Intelligence**
- [Artificial Intelligence] is intelligence demonstrated by machines, as opposed to intelligence of humans and other animals¹; Used in numerous contexts, speech recognition, image recognition, the most popular one is **Chat GPT**.
- [Agent of Intelligence] agent is usually a character in the game – but could also be a vehicle, a robot, or occasionally something more abstract such as a whole group of entities, or even a country or civilization. In each case it is a thing that needs to observe its surroundings, make decisions based on that, and act upon them; Enemies, NPCs, procedural generation.
- From [Agent of Intelligence] to [Artificial Intelligence]



Artificial Intelligence – A Brief History

- 1950-1980: from non-AI to AI
 - Video game was born in 1951. Video game from 1951 to 1970 featured discrete logic and two players, which means no AI involved in that period.
 - The first single player game appeared in 1970s. Famous ones in this period are Speed Race, Qwark, Hunt the Wumpus and Star Trek. In these games, enemies can move based on the **stored pattern**.
 - Space Invader in 1978 has distinctive movement pattern, which is based on **Hash Function**.
- 1980-2000: Booming period
 - AI pattern was introduced into maze games, fighting games and sport games. “**Tactics**” system was introduced into game, which allowed characters to be controlled by computer AI in following the leader.
 - In 1990s, a new game genre prompted the use of AI tools such as **finite state machine**. But it seems to be a not successful practice.
- 2000-Present:
 - In 2000 years, **bottom-up method** was introduced into video game. Some famous games are Creatures or Black and White. **AI are the main aspect of the game.**



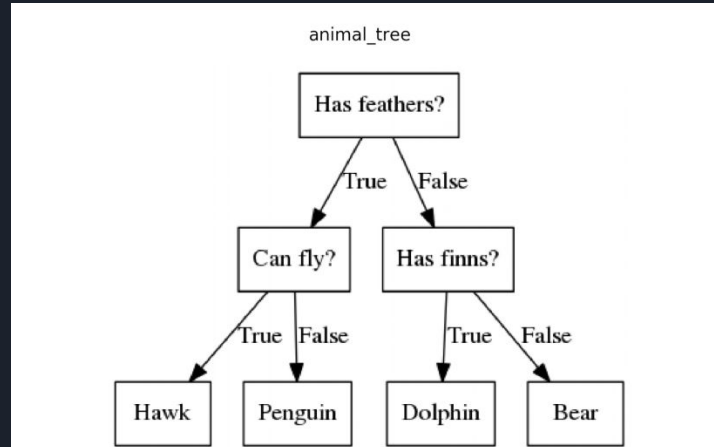
Common AI Implementations

- **NPC Decision-Making**
 - Decision Trees, Finite State Machines, and Behavior Trees can be used to make NPC decision-making more dynamic and believable.
- **Dynamic Difficulty Balancing**
 - dynamic difficulty adjustment (DDA) or dynamic game balancing (DGB), is the process of automatically changing parameters, scenarios, and behaviors in a video game in real-time, based on the player's ability, in order to avoid making the player bored.
- **Pathfinding**
 - The A* algorithm and Dijkstra's Algorithm can be used to plot the shortest route between two points.
- **Procedural Generation**
 - AI can be used to create new levels on each load; the practice of Deep Learning makes the possibilities of realistic generation endless.

NPC Decision-Making – Decision Trees

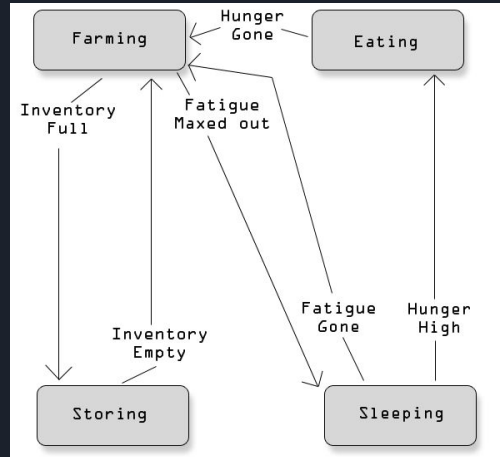
- The most basic version of AI decision-making.
- Can be implemented in a number of ways, but the easiest is in a layered if-else statement.
- Generally, the tree can be expressed as a graph-based data structure:
 - Internal nodes hold pointers to predicate function
 - Leaf nodes hold pointers to action function

```
public string AnimalTree(Animal anim)
{
    if(anim.Has("Feathers")){
        if(anim.Has("Flight")){
            return "Hawk";
        } else {
            return "Penguin";
        }
    } else {
        if(anim.Has("Fins")){
            return "Dolphin";
        } else {
            return "Bear";
        }
    }
}
```



NPC Decision-Making – Finite State Machines

- A step up in complexity from Decision Trees, adding the concept of *state*.
- Object transitions between states, behaving differently based on this state.
- Consider the example of an NPC bear:
 - The bear starts out wandering the woods.
 - If hungry, the bear attacks other entities.
 - If full, the bear does not approach other entities.
 - If fighting another entity and losing, the bear goes back to wandering.





NPC Decision-Making – Behavior Trees

- Easier to design and more lightweight than Finite State Machines.
- Internal nodes represent behaviors.
- Leaf nodes represent one of two world interactions:
 - Conditions – Read-only data that allow sharing of information with other objects.
 - Act as filters to indicate which actions are performed.
 - E.g. is there another object nearby?
 - Actions – World-affecting behaviors.
 - E.g. playing a sound or picking up an object.
- Composite tasks may be written in one of two ways:
 - Sequence – Executes tasks in order until one fails.
 - Selector – Executes tasks in order until one succeeds.
- Example later...



Dynamic Difficulty Balancing

- AI can responsively adjust the difficulty based on user performance.
- Useful for creating an enjoyable and replayable experience for the user.
- Useful for fixing balancing issues in unbalanced games.
- The AI will run a neural network to find the optimal difficulty for the game.
 - This neural network will test many variables and find one where the game is balanced with the right amount of difficulty.




Pathfinding – Basics

- The shortest path between 2 points on a map.
- Used to move NPCs around an area on the map.
- Very important in many modern video games.
- Example of how Pathfinding works in video games:
 - Nodes around the map signifying walkable areas.
 - Algorithm finds the shortest path from one node to another.
 - Algorithm accounts for any walls or obstacles around the map.
- Has many uses outside of video games:
 - Google Maps directions
 - Logistics



Pathfinding – Graph-Based

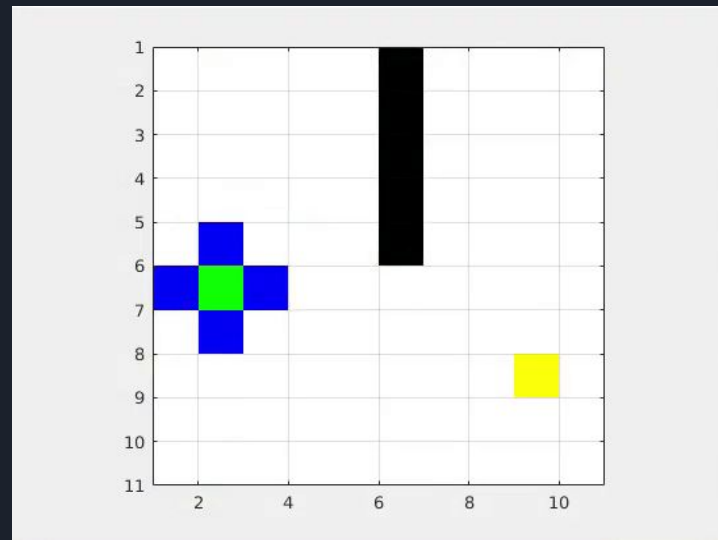
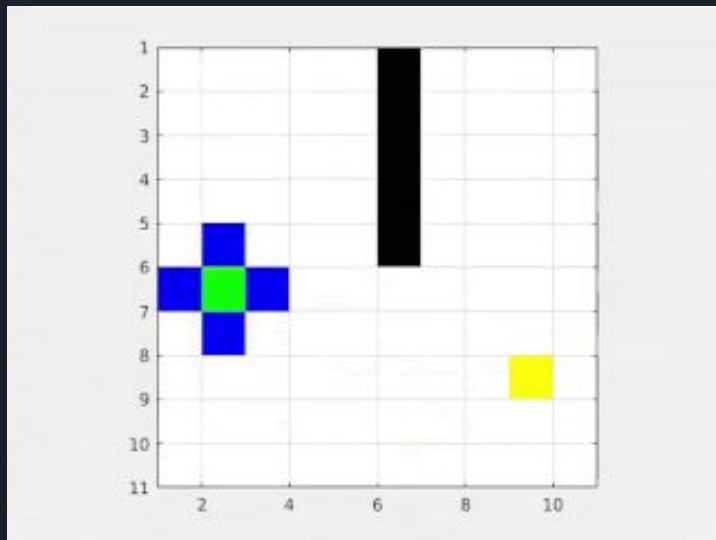
- Most simple type of Pathfinding.
- Uses Graph Theory to find the optimal path between 2 nodes.
- Graph will usually be a weighted graph
- Traveling Salesman Problem
 - **“Find the shortest possible route that visits each node exactly once and returns to the original node”**
 - Uses a more complex and restricted version of Graph-Based Pathfinding.



Pathfinding – Grid-Based

- Essentially the same as Graph-Based Pathfinding but in a grid.
- Video games will often use Grid-Based Pathfinding.
- 2 popular algorithms for Grid-Based Pathfinding.
- Dijkstra's algorithm
 - An undirected algorithm that tests every direction to the destination node.
 - Returns the first path to get to the destination node.
 - A brute-force version of Pathfinding
 - Much more thorough than A* algorithm
- A* algorithm
 - A directed algorithm that first tests in the direction of the destination node.
 - Path starts at the starting node and then continues in the direction of the destination node.
 - Deviates from path once it hits an obstacle.
 - Much faster on average than Dijkstra's algorithm

Dijkstra's and A* algorithm Comparison





Procedural Generation

- Procedural Generation is using computer-generated data, usually pseudorandom, to create a level or world without predefinition by the game developers.
- A basic example of this practice is *Enter the Gungeon*, a roguelike game in which the player fights through newly generated floors in each new game.
- This can be performed with a simple random function:
 - Create a list of basic room layouts.
 - Sort these rooms based on entrances/exits.
 - Randomize the room led to by a doorway based on whether the doorways match up.



The Future of Procedural Generation

- In 2016, the video game *No Man's Sky* implemented procedural generation on a radical level: over 18 *quintillion* Earth-sized planets were generated, each with semi-unique flora and fauna.
- Not a single one of these planets is hand-designed.
- More recently, AI chatbots have shown the extent to which Deep Learning algorithms can replicate human speech, stock trading, and sales principles.
- What else could Deep Learning algorithms generate?
 - NPCs
 - Dialogue
 - Weapons/tools
 - Etc.

Ethics

Former Cisco engineer sentenced to prison for deleting 16k Webex accounts

Former Cisco engineer accessed Cisco's AWS accounts, and deleted 456 virtual machines, which resulted in the loss of 16k Webex accounts



Written by **Catalin Cimpanu**, Contributor on Dec. 11, 2020



IEEE Code
of Ethics

<https://www.zdnet.com/article/former-cisco-engineer-sentenced-to-prison-for-deleting-16k-webex-accounts/>



IEEE Code of Ethics

- Set of standards created by a joint team from IEEE-CS and ACM.
- Intended to set forth standards for ethical behavior among software engineers.
- 8 Principles:
 1. Public
 2. Client and Employer
 3. Product
 4. Judgment
 5. Management
 6. Profession
 7. Colleagues
 8. Self



1. Public

- “Software engineers shall act consistently with the public interest.”
- What does this mean?
 - Accepting responsibility for one’s own work.
 - Approve only software that is reasonably expected to be safe.
 - Disclose and handle any potential danger to users.
 - Be fair and avoid deception in all public statements.
 - Consider and aid in reduction of disadvantage from user to user.
 - Contribute to public causes and education wherever possible.
- In summary, behave in an edifying and constructive way in regards to how the public interacts with your software.



2. Client and Employer

- “Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.”
- What does this mean?
 - Be honest about abilities and limitations.
 - Only use legal, approved software and documentation.
 - Maintain confidentiality.
 - Identify, document, and report problematic projects.
 - Accept no work that conflicts with primary employment.
 - Promote no interest adverse to employer unless ethically compelled.
- In summary, be straightforward and loyal to your employer wherever it does not conflict with public interest.



3. Product

- “Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.”
- What does this mean?
 - Strive for high project quality, clearing trade-offs with employer and client.
 - Ensure proper and achievable goals, qualifications, and appropriate methods.
 - Work to follow ethical, economic, cultural, legal, and professional standards.
 - Understand and document specifications of software.
 - Ensure realistic estimates and adequate testing.
 - Create software that respects user privacy.
 - Use only accurate, ethical, lawful information.
- In summary, create high-quality, legal, well-documented software.



4. Judgment

- “Software engineers shall maintain integrity and independence in their professional judgment.”
- What does this mean?
 - Support and maintain human values in all technical judgments.
 - Only endorse documents witnessed firsthand and with which you agree.
 - Maintain professional objectivity in software evaluation.
 - Do not engage in deceptive financial practices.
 - Disclose conflicts of interest.
 - Do not participate in organizations which may lead to conflicts of interest.
- In summary, behave professionally and ethically in all judgments and disclose any situation which may conflict with this behavior.



5. Management

- “Software engineering managers and leads shall subscribe to and promote an ethical approach to the management of software development and maintenance.”
- What does this mean?
 - Ensure effective, informative management.
 - Assign work based on employee contributions and ambition.
 - Ensure realistic quantitative estimates.
 - Attract employees by accurate description, hiring and paying fairly.
 - Ensure fair ownership agreements concerning intellectual property.
 - Provide for due process for Code violations.
 - Do not make employees violate this Code or punish them for voicing concern.
- In summary, effectively lay out expectations and treat employees justly.



6. Profession

- “Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.”
- What does this mean?
 - Help develop an environment favorable to ethical behavior.
 - Aid public knowledge through organizations, meetings, and publications.
 - Support other engineers striving to follow this Code.
 - Do not promote your own interest over profession, client, or employer.
 - Obey all laws governing work except in extraordinary ethical circumstances.
 - Be accurate in software descriptions, and correct errors in software.
 - Make known your commitment to - and report violations of - this Code.
- In summary, uplift your profession and encourage commitment to this Code.



7. Colleagues

- “Software engineers shall be fair to and supportive of their colleagues.”
- What does this mean?
 - Assist colleagues in Code adherence and professional development.
 - Fully credit others’ work.
 - Review others’ work objectively and give fair hearing to others’ opinions.
 - Assist colleagues in being aware of standard work practices.
 - Do not unfairly intervene in the career of any colleague.
 - Call upon others’ opinions to supplement your areas of competence.
- In summary, assist colleagues whenever possible and accept assistance in return.



8. Self

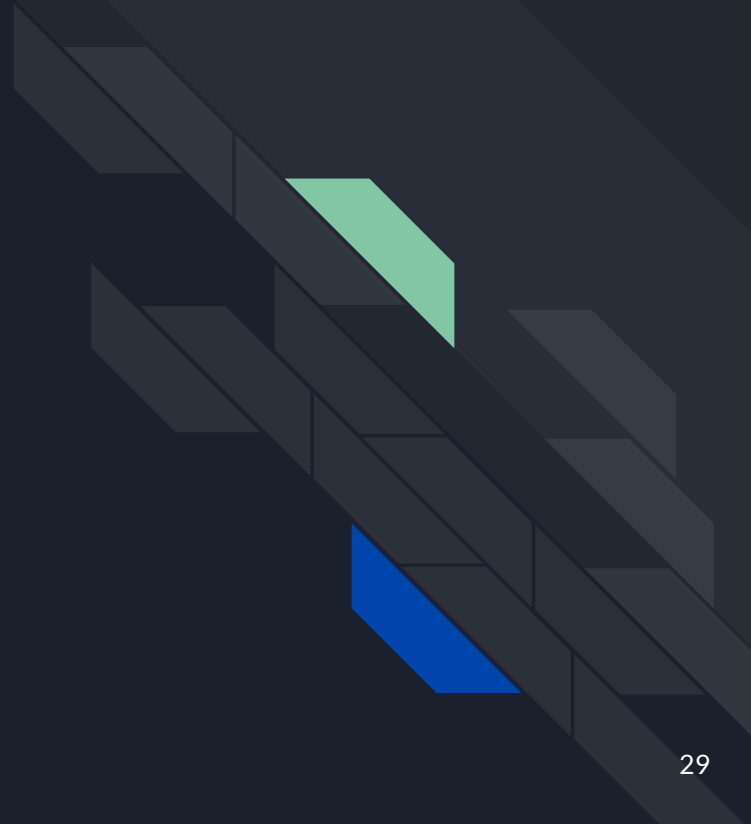
- “Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.”
- What does this mean?
 - Pursue awareness of developments in software engineering.
 - Improve your ability to create quality software and accurate documentation.
 - Understand software you use, relevant standards, and this Code.
 - Do not give unfair treatment to others based on irrelevant prejudice.
 - Do not influence others to violate this Code.
- In summary, continue to pursue professional knowledge and self-improvement.



IEEE Code of Ethics

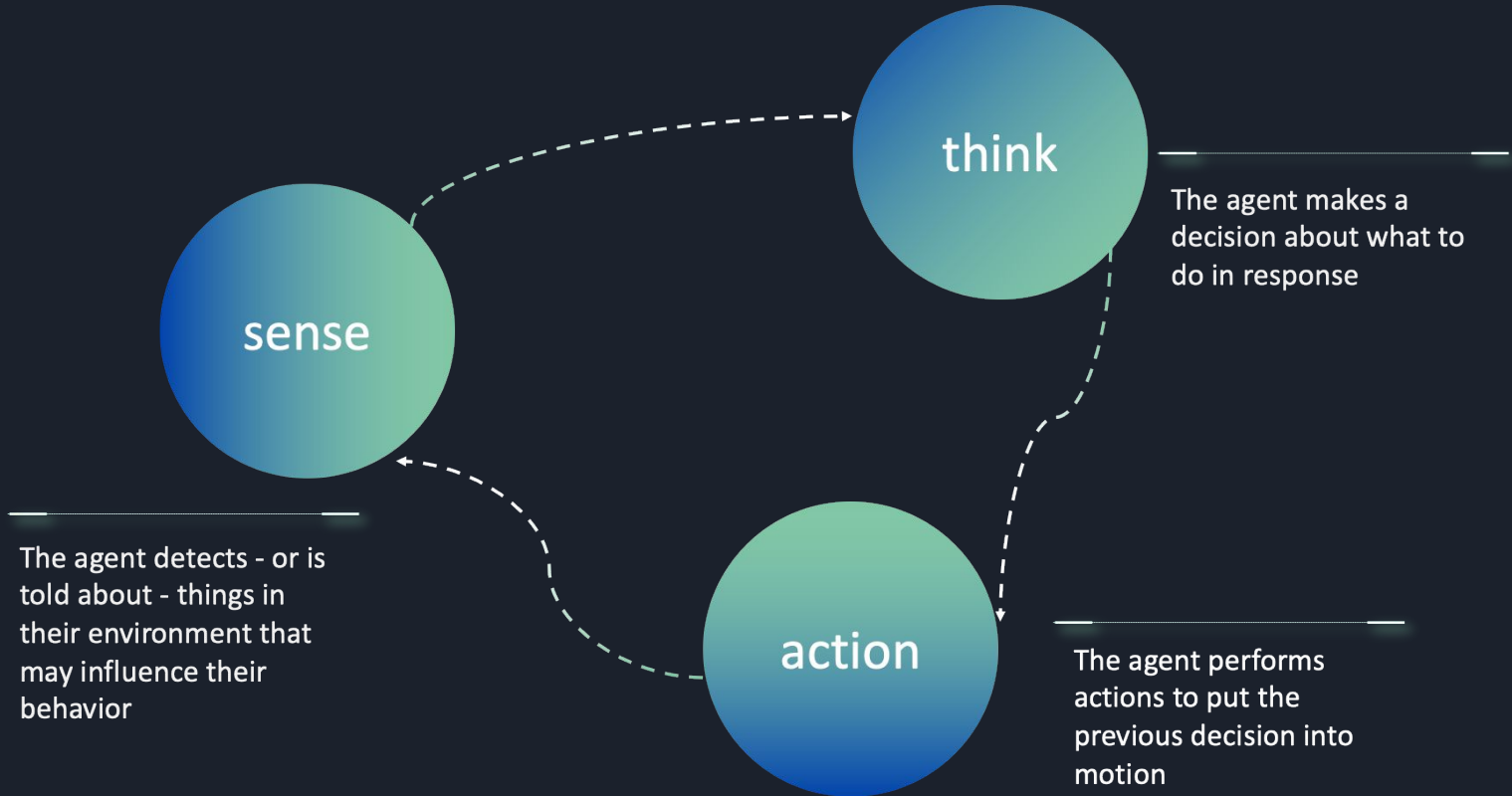
- Ethics issues are not common in the industry.
- Ethics issues are not **uncommon** in the industry.
- Every programmer has the impulse to try “rm -rf /*”
- Be responsible; be ethical.

AI Demos



Basic Ideas in Basic Game AI

Game AI is mostly focused on which actions an entity should take, **based on the current conditions**.





Basic Ideas in **Basic** Game AI (cont.)

Restrictions

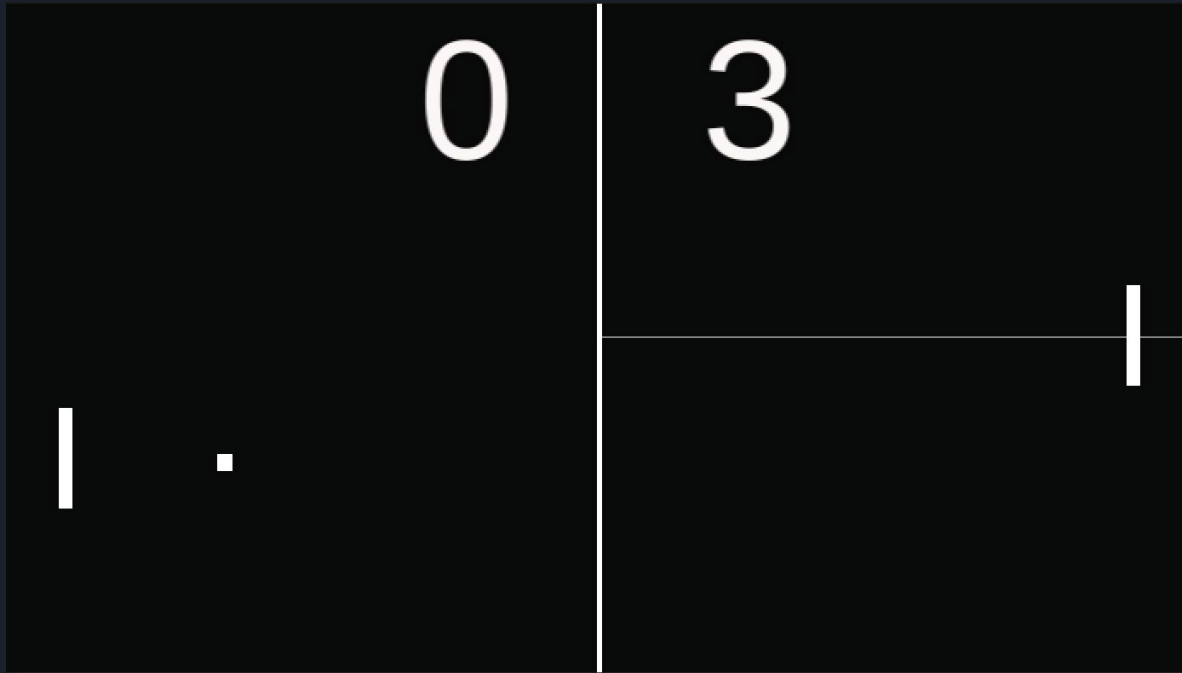
1. It **isn't** usually 'pre-trained' like a **machine learning** algorithm would be.
2. The game is usually supposed to provide **entertainment** and challenge **rather than** be 'optimal'.
3. There is often a requirement for agents to appear '**realistic**', so that players can feel that they're competing against human-like opponents. Alpha GO is not needed in video game.
4. It needs to run in '**real-time**'.
5. It's ideal if at least some of the system is **data-driven rather than** hard-coded,



Pong

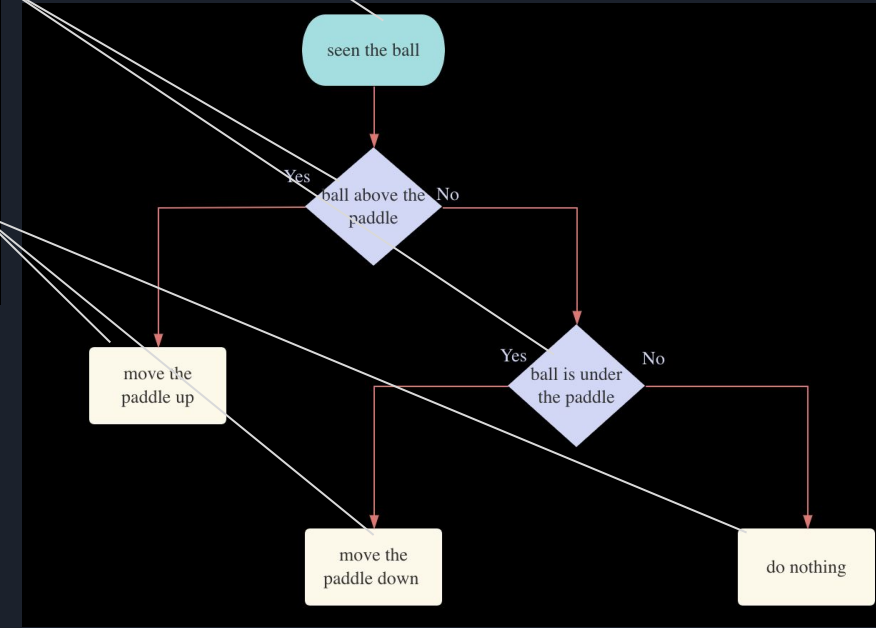
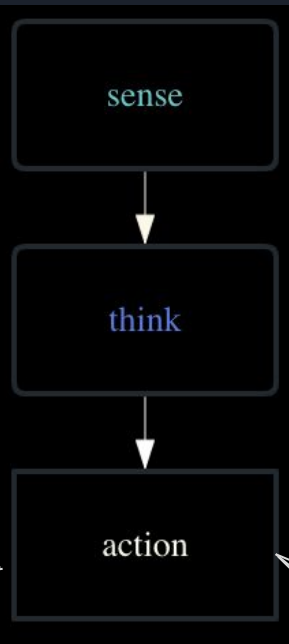
Basic Idea in PONG

- **Goal:** Move the paddle and make the ball bounce back.
- **Solution:** try and position the paddle below the ball at all times. By the time the ball reaches the paddle, the paddle is ideally already in place and can return the ball.



Pong

Basic Idea in PONG



every frame/update while the game is running:

if the ball is above the paddle:

move the paddle up

else if the ball is to the bottom of the paddle:

move the paddle down

else

do nothing

Code in Pong

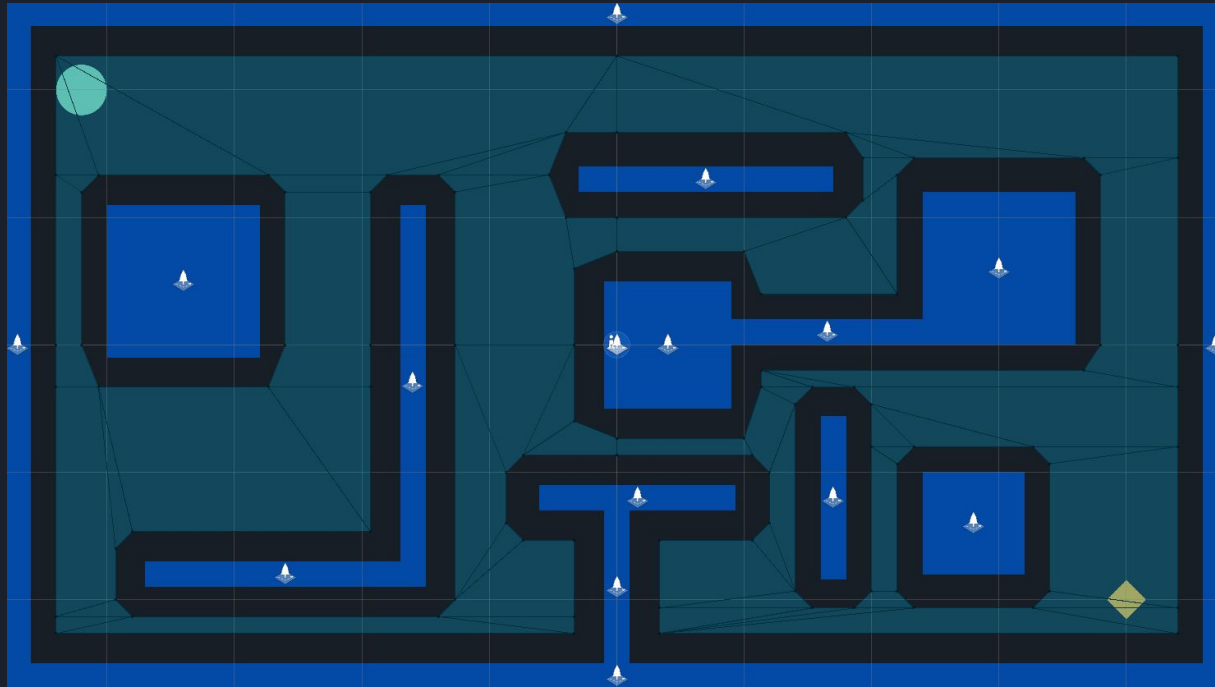
```
public class ComputerPaddle : Paddle
{
    public float speed = 10.0f;
    public Rigidbody2D ball;
    private void FixedUpdate()
    {
        if (this.ball.velocity.x > 0.0f)
        {
            if(this.ball.position.y > this.transform.position.y)
            {
                _rigidbody.AddForce(Vector2.up * this.speed);
            }else if(this.ball.position.y < this.transform.position.y)
            {
                _rigidbody.AddForce(Vector2.down * this.speed);
            }
        }
        else
        {
            if(this.transform.position.y > 0.0f)
            {
                _rigidbody.AddForce(Vector2.down * this.speed);
            }else if(this.transform.position.y < 0.0f)
            {
                _rigidbody.AddForce(Vector2.up * this.speed);
            }
        }
    }
}
```



Pathfinding in Unity

- Unity has a Pathfinding tool called Navmesh.
- Navmesh simplifies the work needed to create an efficient Pathfinding model.
- Navmesh uses the A* algorithm for Pathfinding
- 3 Components for Navmesh Pathfinding:
 - Baked Navmesh map – A map of all the walkable and non-walkable areas on the map.
 - Navmesh Agent – The NPC that is being moved by Navmesh Pathfinding.
 - Navmesh Target – The destination point for the Navmesh Agent

Navmesh Example





Difficulty Balancing

- AI can be used to create realistic and challenging enemy encounters.
- The video game *Alien: Isolation* used the popular *Alien* film universe to create a terrifying experience of powerlessness and solitude as the lone human on a broken-down space station.
- The player is faced with a nearly invulnerable alien that hunts them through the station.
- While many developers would implement a simple prescribed route or pathfinding system, the creators of *Alien: Isolation* wanted the player to feel that they were being stalked by a legitimate intelligent being.
- To do this, two AI systems were implemented:
 - The director, which always knows the player's location.
 - The enemy, which is occasionally lured toward the player by the director.

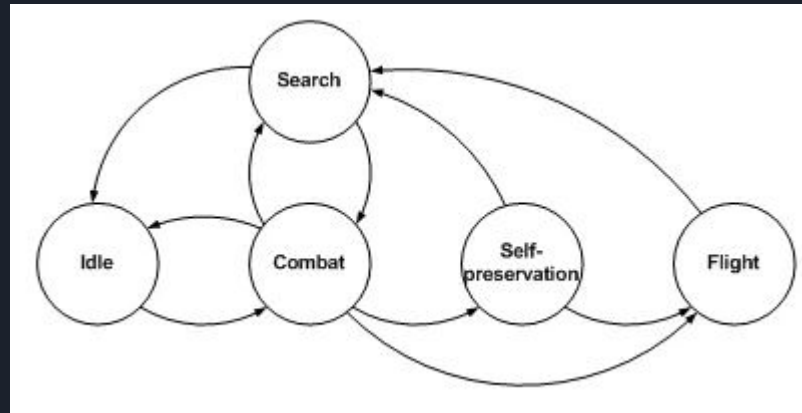


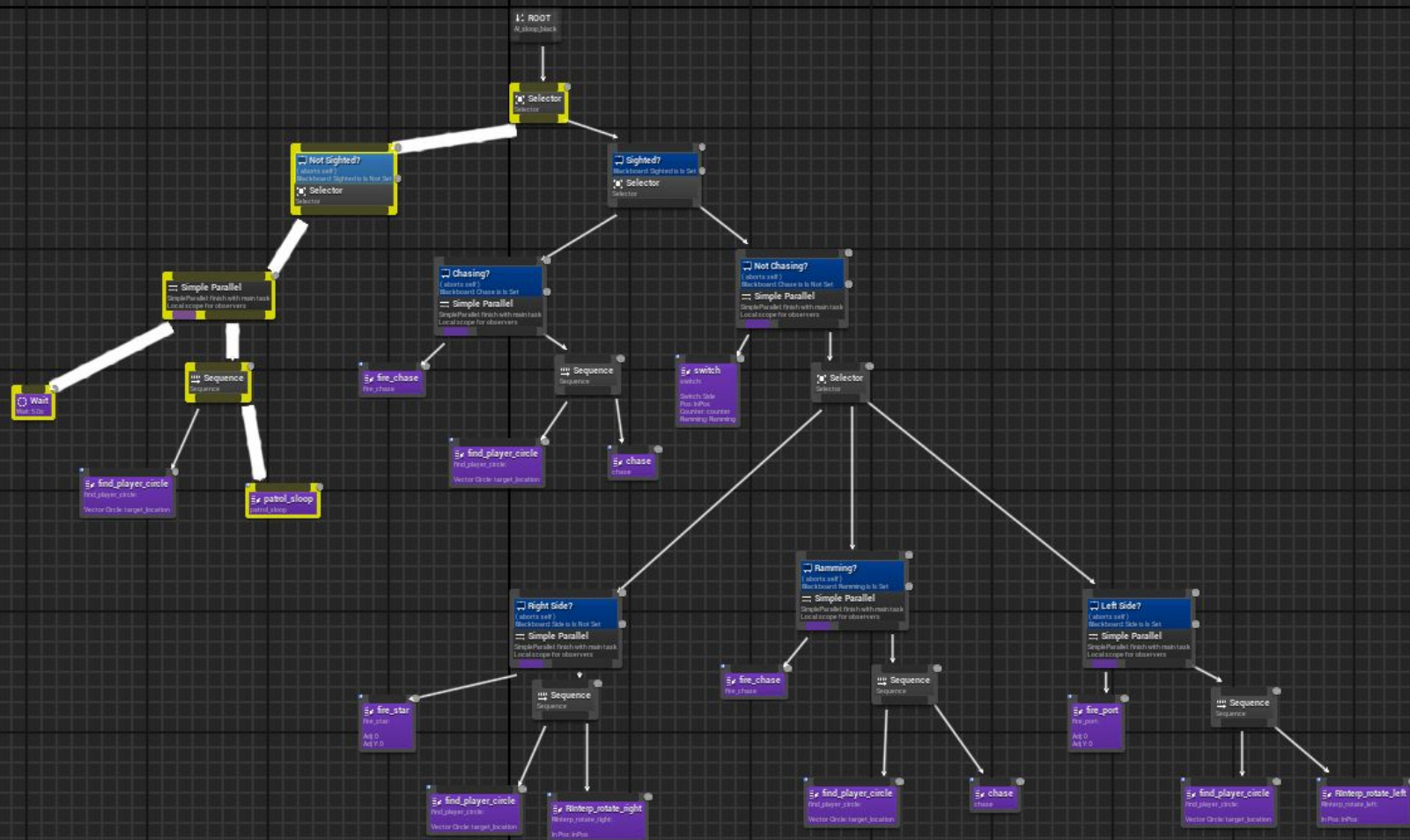
The Menace System

- Player can move around freely, but is being stalked by a deadly enemy.
- Director AI:
 - Monitors player location at all times but is not a direct guide to the enemy.
 - Maintains a “menace gauge” to keep track of player tension, taking the following into account.
 - Enemy distance from player
 - Whether the player can see the enemy
 - When menace gauge is low, the enemy is pinged more often. When the gauge is high, the enemy is pinged less often.
- Enemy AI:
 - Moves toward occasional pings from the director AI, and toward the actual player when within a certain distance.
 - Ends the game upon collision with the player.

The History of Behavior Trees

- Pioneered by R. Geoff Dromey in 2001
- Originally called "genetic software engineering"
- Expanded upon by the Dependable Complex Computer-based Systems Group (DCCS)
- Famous example - *Halo 2*





Questions?



References

1. https://en.wikipedia.org/wiki/Artificial_intelligence
2. Dromey, R. G. (2006, June 19). FORMALIZING THE TRANSITION FROM REQUIREMENTS TO DESIGN. Internet archive: Wayback Machine. Retrieved March 24, 2023, from <https://archive.org/web/>
3. Eliaçık, Eray. "Ai in Gaming: A Complete Guide." *Dataconomy*, 28 Aug. 2022, https://dataconomy.com/2022/04/artificial-intelligence-games/#Typical_applications_of_AI_in_games.
4. Gotterbarn, Donald. "Code of Ethics: IEEE Computer Society." *Code of Ethics | IEEE Computer Society*, 1999, <https://www.computer.org/education/code-of-ethics>.
5. Isla, Damian. "GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI" 11 Mar. 2005, www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai. Accessed 26 Mar. 2023.
6. McCarthy, John. "What Is Artificial Intelligence?" *Stanford University*, 12 Nov. 2007, <https://www-formal.stanford.edu/jmc/whatisai.pdf>.
7. Mount, Dave, and Roger Eastman. "Artificial Intelligence for Games: Decision Making." CMSC 425. 2018.