Adversarial Search

Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

- In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which often expressed in the form of a sequence of actions.
- But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.
- The environment with more than one agent is termed as **multi-agent environment**, in which each agent is an opponent of other agent and playing against each other. Each agent needs to consider the action of other agent and effect of that action on their performance.
- So, **Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games**.
- Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

Types of Games in AI:

	Deterministic	Chance Moves
Perfect information	Chess, Checkers, go, Othello	Backgammon, monopoly
Imperfect information	Battleships, blind, tic-tac-toe	Bridge, poker, scrabble, nuclear war

- **Perfect information:** A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other moves also. Examples are Chess, Checkers, Go, etc.
- **Imperfect information:** If in a game agents do not have all information about the game and not aware with what's going on, such type of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.

- **Deterministic games:** Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.
- **Non-deterministic games:** Non-deterministic are those games which have various unpredictable events and has a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called as stochastic games. Example: Backgammon, Monopoly, Poker, etc.

> Note: In this topic, we will discuss deterministic games, fully observable environment, zero-sum, and where each agent acts alternatively.

Zero-Sum Game

- Zero-sum games are adversarial search which involves pure competition.
- In Zero-sum game each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.
- One player of the game try to maximize one single value, while other player tries to minimize it.
- Each move by one player in the game is called as ply.
- Chess and tic-tac-toe are examples of a Zero-sum game.

Zero-sum game: Embedded thinking

The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out:

- What to do.
- How to decide the move
- Needs to think about his opponent as well
- The opponent also thinks what to do

Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

Formalization of the problem:

A game can be defined as a type of search in AI which can be formalized of the following elements:

- **Initial state:** It specifies how the game is set up at the start.
- **Player(s):** It specifies which player has moved in the state space.
- **Action(s):** It returns the set of legal moves in state space.
- **Result(s, a):** It is the transition model, which specifies the result of moves in the state space.
- **Terminal-Test(s):** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
- **Utility(s, p):** A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

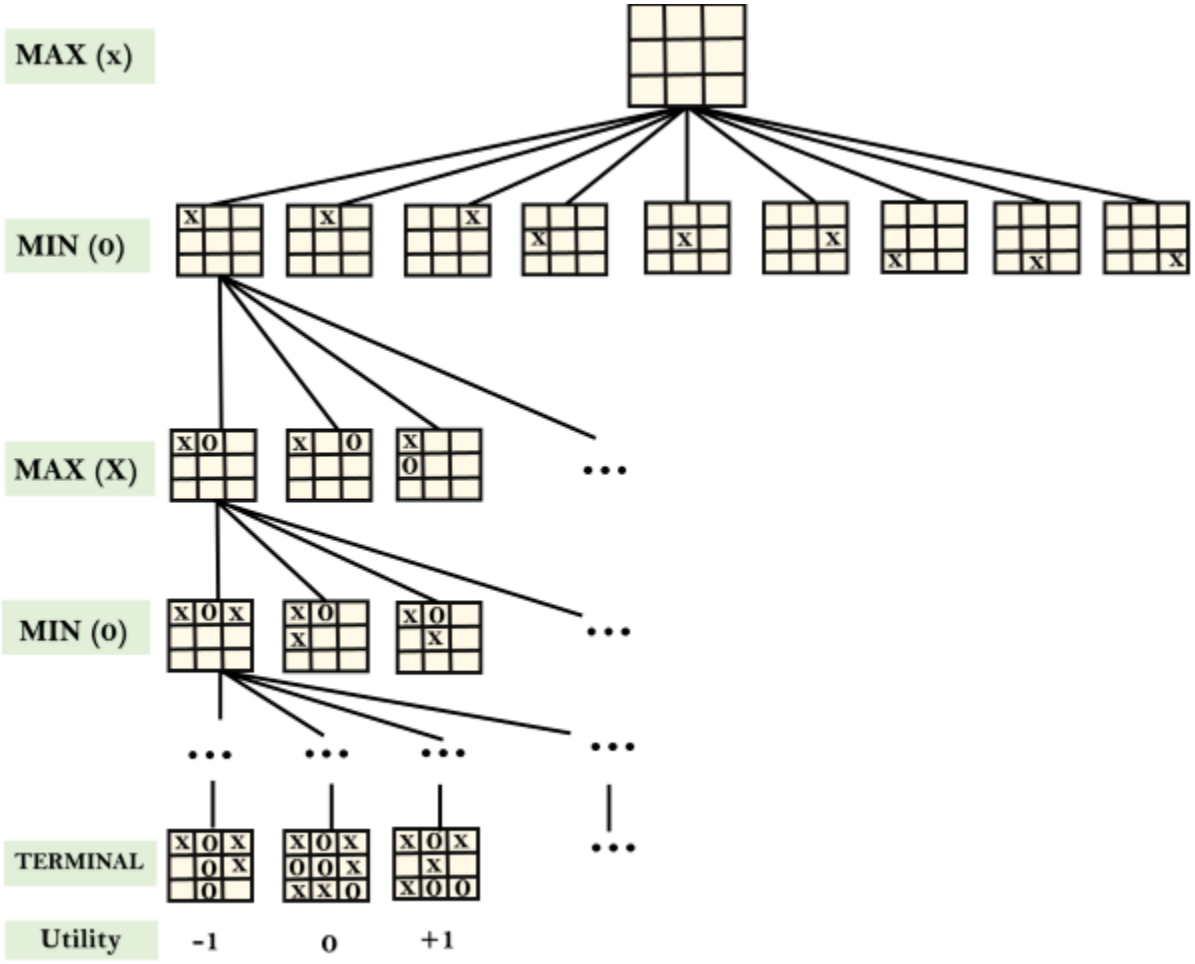
Game tree:

A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function.

Example: Tic-Tac-Toe game tree:

The following figure is showing part of the game-tree for tic-tac-toe game. Following are some key points of the game:

- There are two players MAX and MIN.
- Players have an alternate turn and start with MAX.
- MAX maximizes the result of the game tree
- MIN minimizes the result.



Example Explanation:

- From the initial state, MAX has 9 possible moves as he starts first. MAX place x and MIN place o, and both player plays alternatively until we reach a leaf node where one player has three in a row or all squares are filled.
- Both players will compute each node, minimax, the minimax value which is the best achievable utility against an optimal adversary.
- Suppose both the players are well aware of the tic-tac-toe and playing the best play. Each player is doing his best to prevent another one from winning. MIN is acting against Max in the game.
- So in the game tree, we have a layer of Max, a layer of MIN, and each layer is called as **Ply**. Max place x, then MIN puts o to prevent Max from winning, and this game continues until the terminal node.
- In this either MIN wins, MAX wins, or it's a draw. This game-tree is the whole search space of possibilities that MIN and MAX are playing tic-tac-toe and taking turns alternately.

Hence adversarial Search for the minimax procedure works as follows:

- It aims to find the optimal strategy for MAX to win the game.
- It follows the approach of Depth-first search.
- In the game tree, optimal leaf node could appear at any depth of the tree.
- Propagate the minimax values up to the tree until the terminal node discovered.

In a given game tree, the optimal strategy can be determined from the minimax value of each node, which can be written as MINIMAX(n). MAX prefer to move to a state of maximum value and MIN prefer to move to a state of minimum value then:

For a state S MINIMAX(s) =

$$\begin{cases} \text{UTILITY}(s) & \text{If TERMINAL-TEST}(s) \\ \max_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{If PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{Actions}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{If PLAYER}(s) = \text{MIN}. \end{cases}$$

Mini-Max Algorithm in Artificial Intelligence

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudo-code for MinMax Algorithm:

```
1. function minimax(node, depth, maximizingPlayer) is
2. if depth ==0 or node is a terminal node then
3. return static evaluation of node
4.
5. if MaximizingPlayer then      // for Maximizer Player
6. maxEva= -infinity
7.  for each child of node do
8.  eva= minimax(child, depth-1, false)
9. maxEva= max(maxEva,eva)       //gives Maximum of the values
10. return maxEva
11.
12. else                  // for Minimizer player
13.  minEva= +infinity
14.  for each child of node do
15.  eva= minimax(child, depth-1, true)
16.  minEva= min(minEva, eva)       //gives minimum of the values
17.  return minEva
```

Initial call:

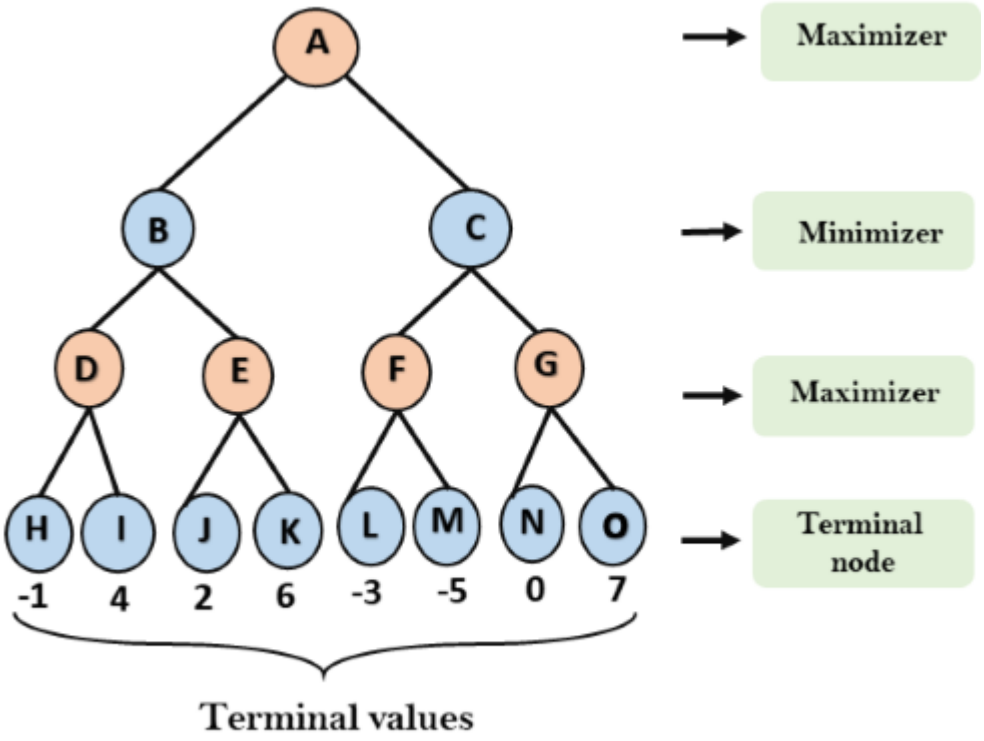
Minimax(node, 3, true)

Working of Min-Max Algorithm:

- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.

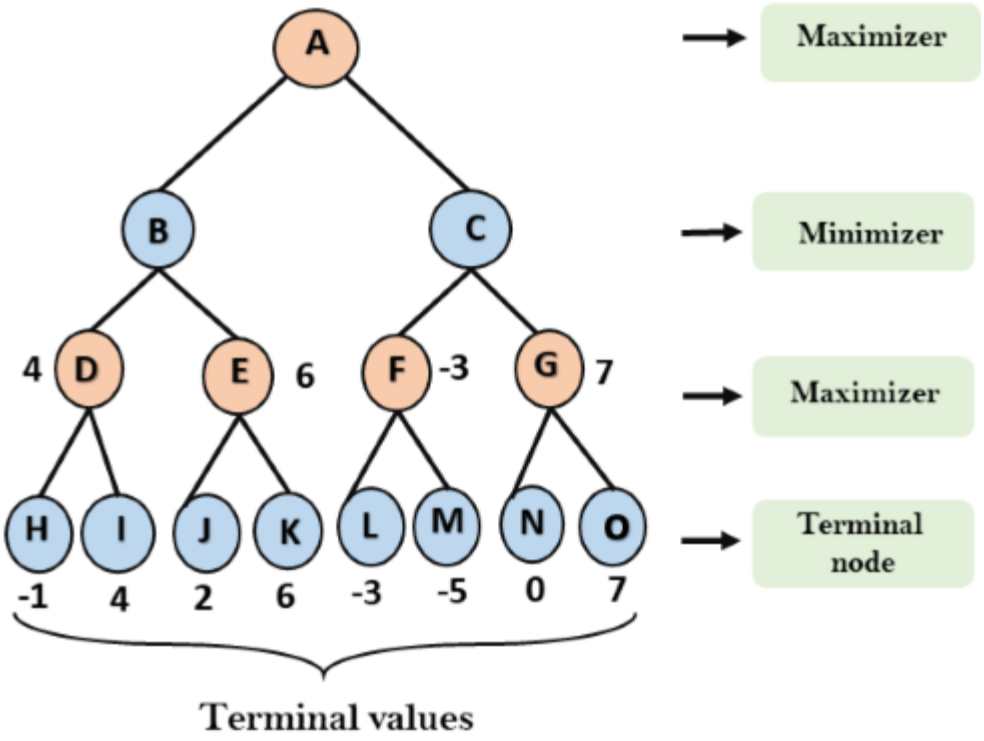
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.



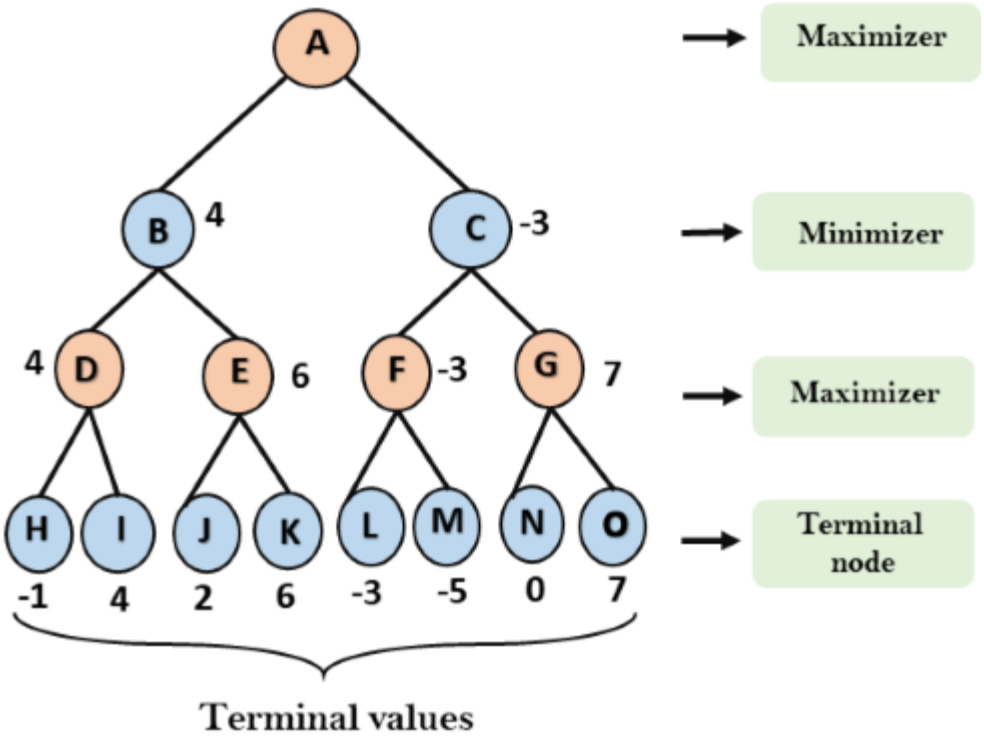
Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- For node D max(-1,- -∞) => max(-1,4)= 4
- For Node E max(2, -∞) => max(2, 6)= 6
- For Node F max(-3, -∞) => max(-3,-5) = -3
- For node G max(0, -∞) = max(0, 7) = 7



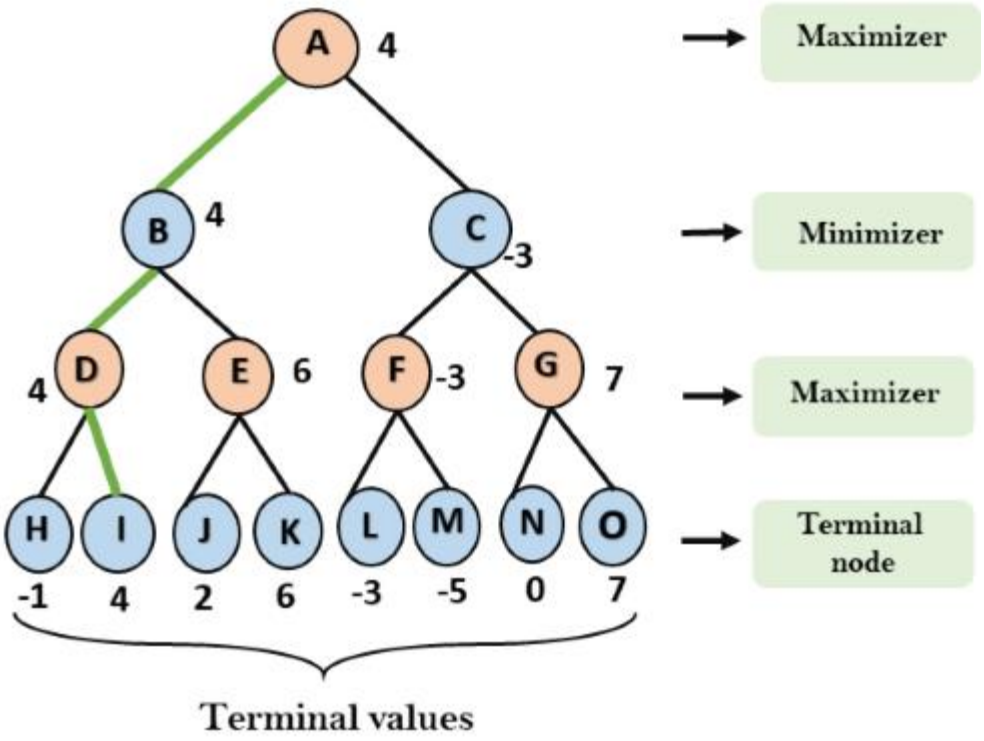
Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3



Step 4: Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- For node A max(4, -3)= 4



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the

player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning** which we have discussed in the next topic.

Alpha-Beta Pruning

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called **pruning**. This involves two threshold parameter Alpha and beta for future expansion, so it is called **alpha-beta pruning**. It is also called as **Alpha-Beta Algorithm**.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.
- The two-parameter can be defined as:
    a. **Alpha:** The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is **-∞**.
    b. **Beta:** The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is **+∞**.
- The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

> Note: To better understand this topic, kindly study the minimax algorithm.

Condition for Alpha-beta pruning:

The main condition which required for alpha-beta pruning is:

1. $\alpha >= \beta$

Key points about alpha-beta pruning:

- The Max player will only update the value of alpha.
- The Min player will only update the value of beta.

- While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
- We will only pass the alpha, beta values to the child nodes.

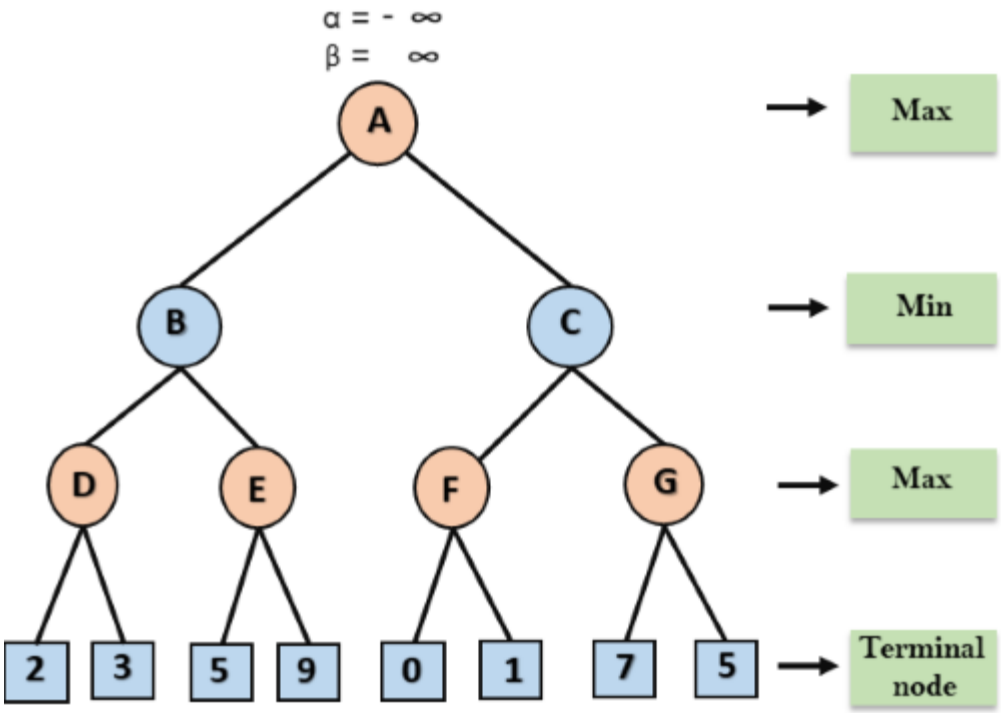
Pseudo-code for Alpha-beta Pruning:

```
1. function minimax(node, depth, alpha, beta, maximizingPlayer) is
2. if depth ==0 or node is a terminal node then
3. return static evaluation of node
4.
5. if MaximizingPlayer then      // for Maximizer Player
6.    maxEva= -infinity
7.    for each child of node do
8.    eva= minimax(child, depth-1, alpha, beta, False)
9.   maxEva= max(maxEva, eva)
10.  alpha= max(alpha, maxEva)
11.   if beta<=alpha
12. break
13. return maxEva
14.
15. else                    // for Minimizer player
16.   minEva= +infinity
17.   for each child of node do
18.   eva= minimax(child, depth-1, alpha, beta, true)
19.   minEva= min(minEva, eva)
20.   beta= min(beta, eva)
21.    if beta<=alpha
22.  break
23. return minEva
```

Working of Alpha-Beta Pruning:

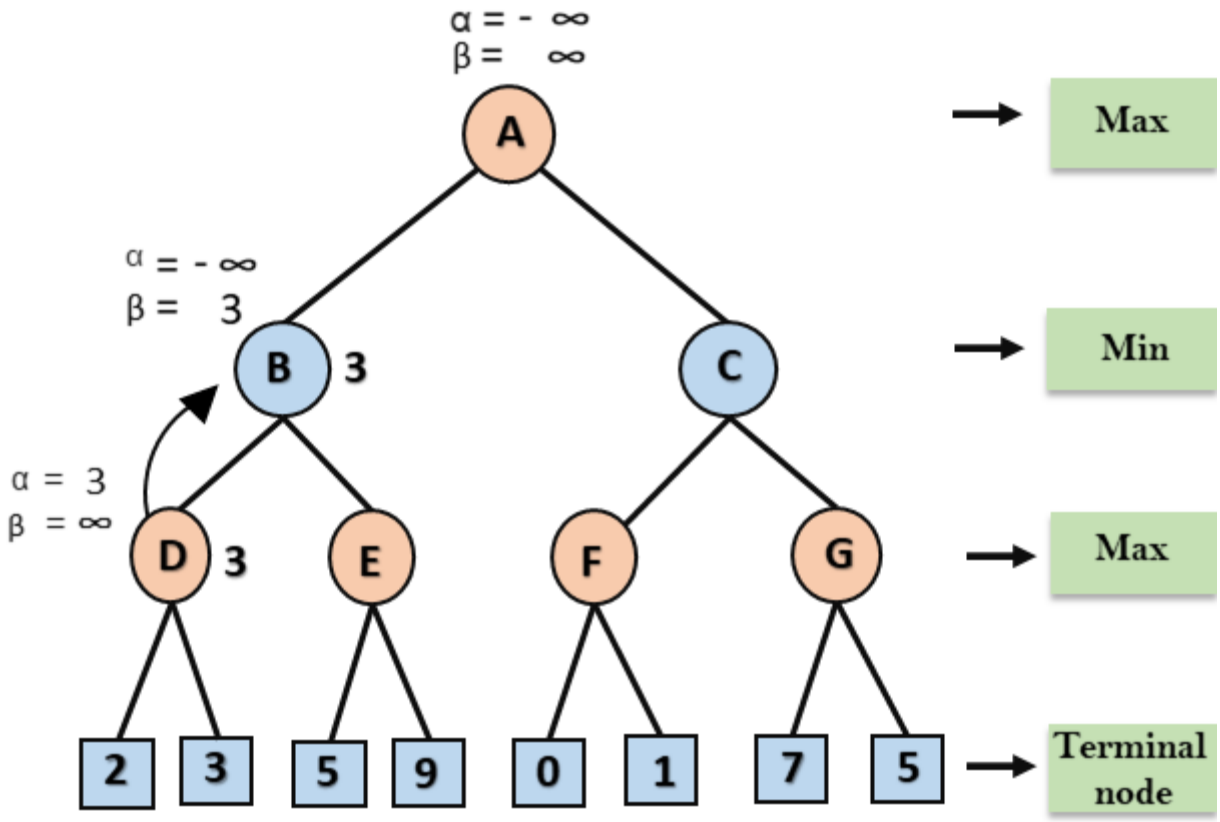
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

Step 1: At the first step the, Max player will start first move from node A where α= -∞ and β= +∞, these value of alpha and beta passed down to node B where again α= -∞ and β= +∞, and Node B passes the same value to its child D.



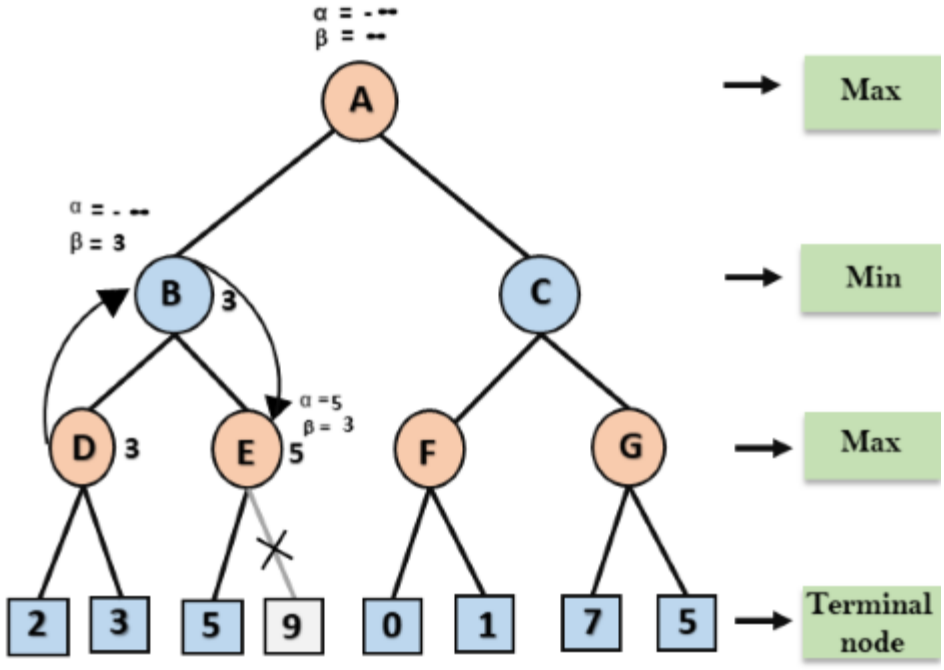
Step 2: At Node D, the value of α will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.

Step 3: Now algorithm backtrack to node B, where the value of β will change as this is a turn of Min, Now β= +∞, will compare with the available subsequent nodes value, i.e. min (∞, 3) = 3, hence at node B now α= -∞, and β= 3.



In the next step, algorithm traverse the next successor of Node B which is node E, and the values of α= -∞, and β= 3 will also be passed.

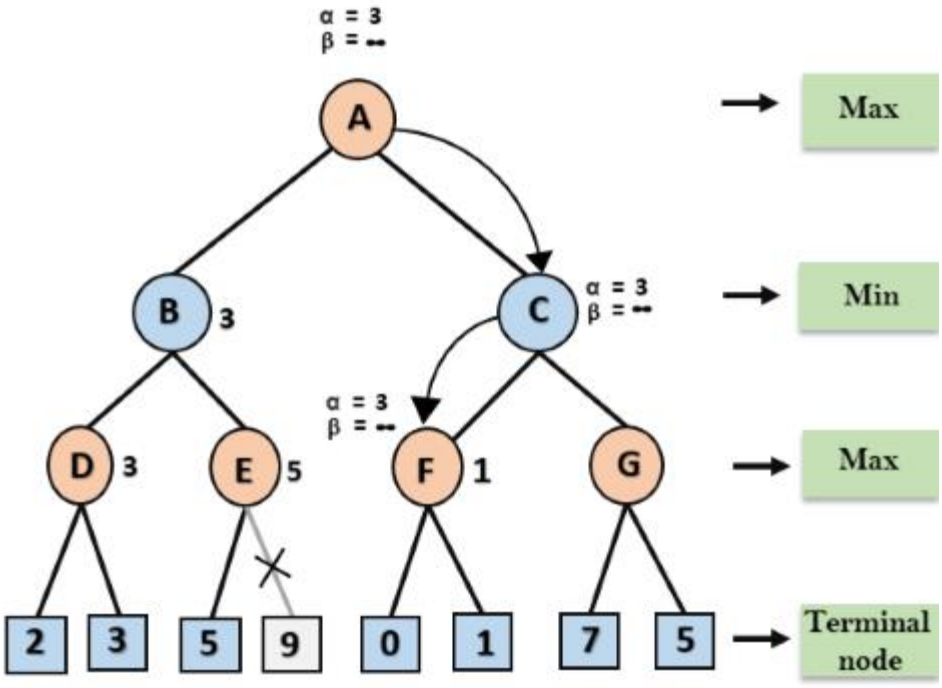
Step 4: At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞, 5) = 5, hence at node E α= 5 and β= 3, where α>=β, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.



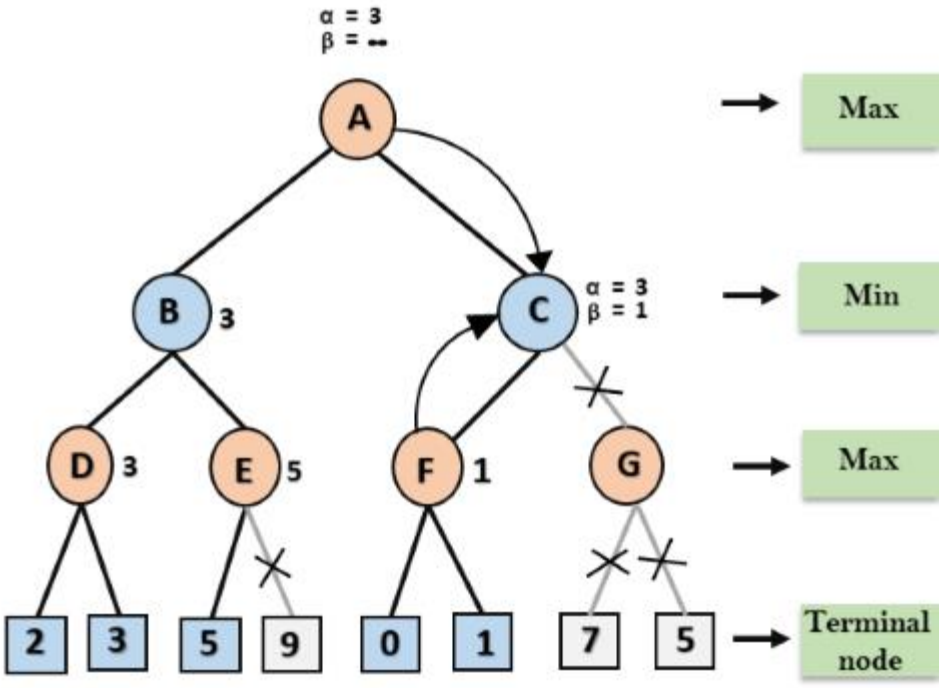
Step 5: At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-∞, 3)= 3, and β= +∞, these two values now passes to right successor of A which is Node C.

At node C, α=3 and β= +∞, and the same values will be passed on to node F.

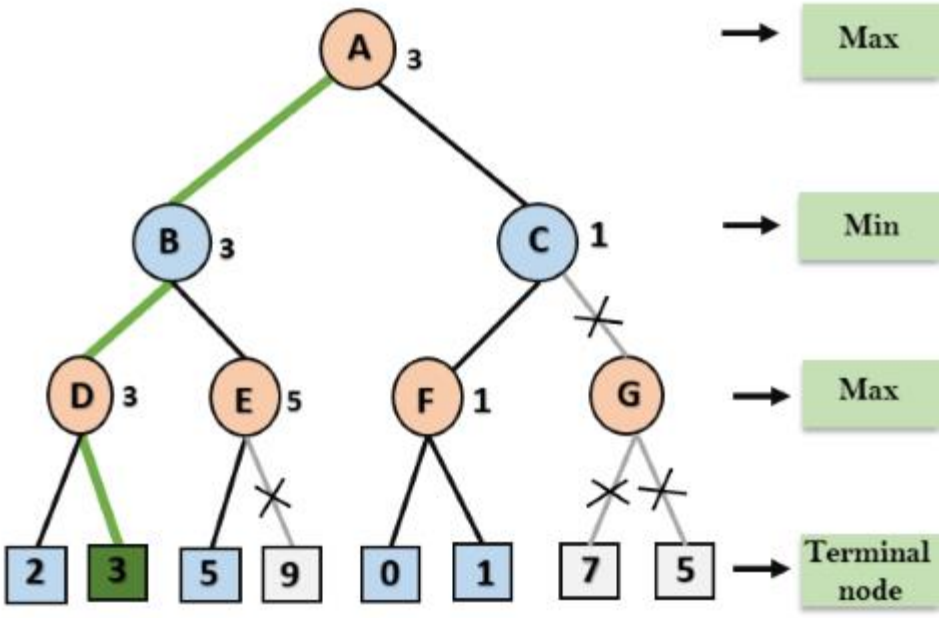
Step 6: At node F, again the value of α will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



Step 7: Node F returns the node value 1 to node C, at C α= 3 and β= +∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, α=3 and β= 1, and again it satisfies the condition α>=β, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8: C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



Move Ordering in Alpha-Beta pruning:

The effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning.

It can be of two types:

- **Worst ordering:** In some cases, alpha-beta pruning algorithm does not prune any of the leaves of the tree, and works exactly as minimax algorithm. In this case, it also consumes more time because of alpha-beta factors, such a move of pruning is called worst ordering. In this case, the best move occurs on the right side of the tree. The time complexity for such an order is $O(b^m)$.
- **Ideal ordering:** The ideal ordering for alpha-beta pruning occurs when lots of pruning happens in the tree, and best moves occur at the left side of the tree. We apply DFS hence it first search left of the tree and go deep twice as minimax algorithm in the same amount of time. Complexity in ideal ordering is $O(b^{m/2})$.

Rules to find good ordering:

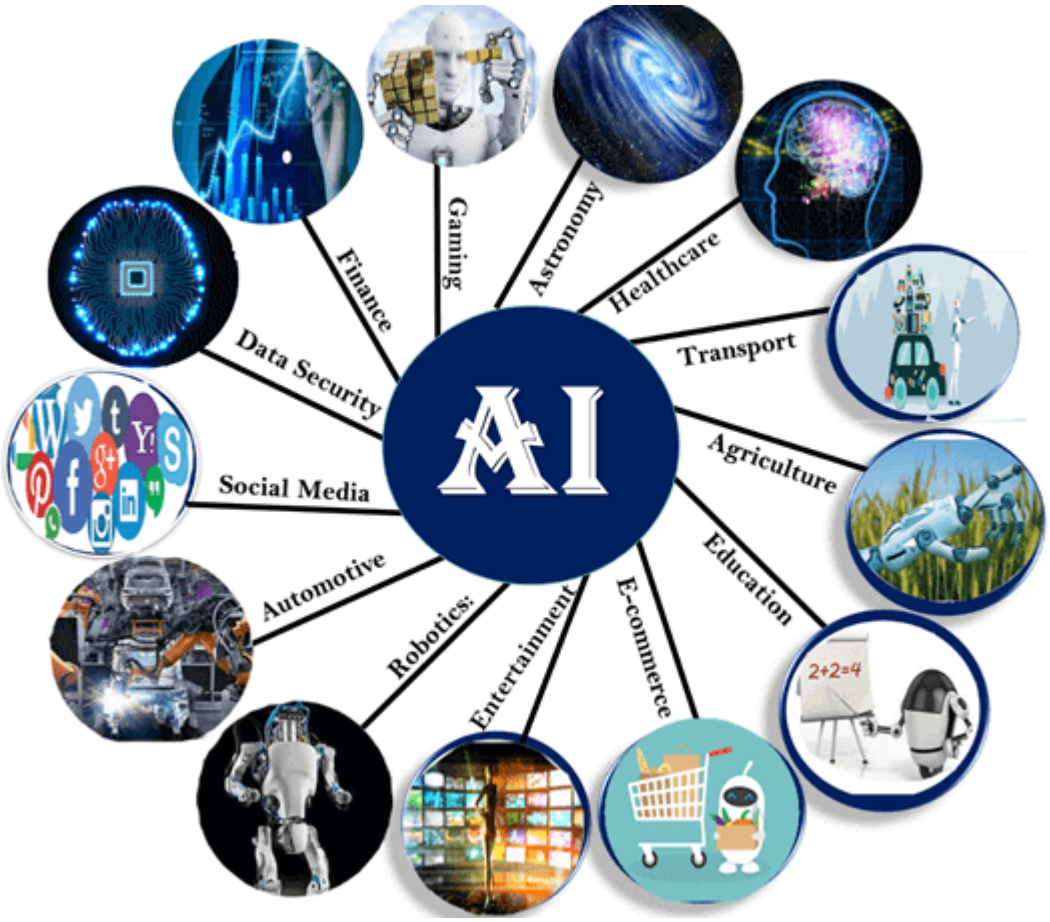
Following are some rules to find good ordering in alpha-beta pruning:

- Occur the best move from the shallowest node.
- Order the nodes in the tree such that the best nodes are checked first.
- Use domain knowledge while finding the best move. Ex: for Chess, try order: captures first, then threats, then forward moves, backward moves.
- We can bookkeep the states, as there is a possibility that states may repeat.

Application of AI

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:



1. AI in Astronomy

- Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare

- In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
- Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

3. AI in Gaming

- AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

4. AI in Finance

- AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

5. AI in Data Security

- The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform,are used to determine software bug and cyber-attacks in a better way.

6. AI in Social Media

- Social Media sites such as Facebook, Twitter, and Snapchat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hashtag, and requirement of different users.

7. AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

- Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
- Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

9. AI in Robotics:

- Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.
- Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

10. AI in Entertainment

- We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

11. AI in Agriculture

- Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

12. AI in E-commerce

- AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

13. AI in education:

- AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
- AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

GAME DEVELOPMENT IN AI

- AI is becoming a universal tool for game design. Despite video game artificial intelligence still being in its infancy, game companies have already started to recognize attractive benefits such as enhanced player experience and cost reduction.
- Ever since the first program that played chess in the '50s, video games have been associated with artificial intelligence. The efforts to solve the challenge of computers defeating human experts in strategy games such as chess, poker, and Go have significantly advanced AI research. In turn, this has led to enhancements in the design of new games.
- In a broad sense, most video games incorporate some form of AI.

  Nevertheless, modern AI methods can be applied in distinct areas that help game companies realize business benefits as well. For example, through enhancing photorealistic effects, generating content, balancing in-game complexities, and providing intelligence to non-playing characters (NPCs), AI improves the overall user experience while saving game companies budget and time.

- Read on to find out more about the importance and impact of AI in games, typical applications, commonly employed AI methods, the most popular games that utilize AI, the business benefits, current limitations in how AI can be applied, and the future of AI in games.

The importance and impact of AI in games

- Game developers strive to deliver valuable interactive experiences to players. These experiences derive from the cumulative effects of a number of orchestrated game elements, including graphics, sounds, gameplay timing, narratives, challenges, and content that directly interacts with the players (allies, opponents, or other objects). Because of this, AI is becoming an unmatched tool that can help designers coordinate the ever-growing complexity of game dynamics.
- The growing popularity of AI in games also has significant business benefits for enterprises. The gaming industry is becoming one of the most profitable sectors, with a market value predicted to reach around 314 billion USD by 2026. As a result, funding of the development of AI-based games worldwide has been steadily rising.

  Today, many startups are emerging in this area. For example, latitude, a startup that develops games using AI-generated infinity storylines, raised 3.3 million USD in seed funding

  in January 2021. Osmo, an interactive play company, has raised 32.5 million USD in funding so far. Gosu Data Lab, another AI gaming startup based in Lithuania, has raised 5.1 million USD in funding. Gosu mainly focuses on exploring gaming data for AI purposes, helping gamers get better at playing.

Our AI development services can improve your

The application of AI in games is diverse; it can be used for image enhancement, automated level generation, scenarios, and stories, balancing

in-game complexity, and adding intelligence to non-playing characters (NPCs).

Image enhancement

At the forefront of image enhancement are the AI engineers who are making efforts to use a deep learning system that turns 3D rendered graphics into photorealistic images. Such a system has been tested on Grand Theft Auto 5. The developed neural network is capable of recreating LA and southern Californian landscapes in great detail. The most advanced image enhancement AI algorithms can convert high-quality synthetic 3D images into real-life-like depictions.
One application of image enhancement in video games is to improve the graphics of classic games. The main idea behind the algorithms proposed for this job is to take a low-resolution image and turn it into a version that looks the same but has many more pixels. This process is called “AI upscaling”.

Game level generation

Game level generation is also known as Procedural Content Generation (PCG). These are the names for a set of methods that use advanced AI algorithms to generate large open-world environments, new game levels, and many other game assets. This is one of the most promising applications of artificial intelligence in game design. Open world or open map games include some of the most popular games to date. These games allow players to explore vast landscapes. Creating such games is very time-consuming from both the design and development perspectives. But AI algorithms can build and optimize new scenery in relation to the game’s status. For example, No Man’s Sky is an AI-based game with an infinite number of new levels generated on the fly while you play.

Scenarios and stories

AI is used to generate stories and scenarios. Most often, AI is used to create an interactive

narrative. In this kind of game, users create or influence a dramatic storyline through actions or what they say. The AI programs use text analysis and generate scenarios based on previously learned storylines. AI Dungeon 2 is one of the most famous examples of this application. The game utilizes a state-of-the-art open-source text generation system built by OpenAI and trained on the Choose Your Own Adventure books.

Balancing in-game complexity

The main advantage of AI algorithms is their ability to model complex systems. Game developers are continuously trying to create more immersive and realistic games. However, modeling the real world is complex. AI algorithms can predict the future effects of gamer actions, and even model things like weather and emotions to balance in-game complexity. A real example of this application is FIFA's ultimate team mode. FIFA automatically calculates a team chemistry score based on the personality traits of the players in a football team. Team
morale oscillates from low to high based on the in-game events (losing the ball, making a well-timed pass, etc.) In this way, teams with better players can lose games against weaker teams because of their morale. In this way, AI can be used to add a layer of complexity.

Adding intelligence to non-playing characters (NPCs)

In most current games, the opponents are pre-programmed NPCs; however, AI is on the path to adding intelligence to these characters. This will make them less predictable and more enjoyable to play against. In

addition, AI allows NPCs to get smarter and respond to the game conditions in novel and unique ways as the game progresses. Many gaming companies have already started working on AI-based NPCs. For example, SEED (EA) trains
NPC characters by imitating the top players in games. This approach will profoundly reduce the development time of NPCs, as hard coding of their behavior is a tedious and lengthy process.

Get more insights into this blockchain battleship game

AI methods used in games

Traditionally the behavior of NPCs was programmed using rule based and finite state machines. The development using these techniques involved programming many conditionals that gave NPCs deterministic behaviors. To reduce the development effort while introducing a degree of unpredictability in games, developers used fuzzy logic. One of the first uses of AI in games programming was through so-called A* pathfinding algorithms that define the behavior of NPCs and their exploration of open worlds. Other techniques include scripting, expert systems, and artificial life (A-life) approaches.

Many popular games such as Black & White, Battlecruiser 3000AD, Creatures, Dirt Track Racing, Fields of Battle, and Heavy Gear employed non deterministic methods such as decision trees, (deep) neural networks, genetic algorithms, and reinforcement learning methods. Let’s explore these techniques in detail.

Decision trees

Decision trees (DTs) are supervised learning models that can be trained to perform classification and regression. They are one of the most basic machine learning methods for game design, and can enable the value of a variable of interest to be predicted through learning simple decision rules inferred from the data features.

Decision trees are pretty simple to understand, and the results can be easily interpreted. Tree visualization techniques are also very advanced. The developed models are known as white box models and can be validated using various statistical tests.

In artificial intelligence game design, DTs are used to describe choices and consequences (predictions of actions). Most modern games use DTs, primarily narrative-based games. In one such use, decision trees can provide players with insights into how the future will lookdepending on their choices. For example, in Star Wars Jedi: Fallen Order, decision trees provide hints regarding the past and future of the main character if certain circumstances were to occur.

(Deep) neural networks

Artificial neural networks (NNs) are structures akin to human brains that can learn various features from training data. Given a large set of data, NNs are capable of modeling very complex real-world and game scenarios. NNs overcome some of the shortcomings of classic AI techniques in game agent design. Furthermore, NNs are self-adaptive and adapt well to game environments that change in real-time.

NN-based game agents can learn in two ways. Either they are trained before being deployed in a game (offline), or the learning process can be applied in real time during the gameplay (online). Online training allows for

the creation of game agents that continuously improve while the game is being played.

NN-based agents can quickly adapt to the changing tactics of human players or other NPCs, and can make sure the game remains challenging even during extended gameplay.

Lately, Deep NN (deep learning) has become a more popular choice for game agent design.

Deep learning in games utilizes multiple layers of neural networks to “progressively” extract features from the input data. Due to its layered approach and increased architectural complexity, deep NN can achieve better results when controlling one or several game agents. These agents can either be NPCs or the game environment itself.

Genetic algorithms

In the most basic terms, a genetic algorithm (GA) is a higher-level procedure, a heuristic, inspired by the theory of natural evolution. The genetic algorithm mimics the process of natural selection, where the fittest candidates are chosen to produce offspring of the next

generation.

GAs are used for various optimization tasks. When compared to different optimization techniques, GAs are capable of delivering excellent results for multicriteria optimizations. In the past, GAs found their place in board games that employ various search techniques when seeking the next best moves. The most recent applications of GAs to NPCs allow adaptation of

these agents to defend against effective but repetitive tactics that human players may employ. The application of GAs leads to a more realistic game experience, where human players or other AI agents cannot find loopholes and dominate the game with repeated steps that always lead to success. The end benefit of GAs is extended playability.

Reinforcement learning

Reinforcement learning (RL) is a machine learning method that is based on learning from trial and error. During training, the model is allowed to play out scenarios and learn from whether things ended well or not so well.

Reinforcement learning is effective when designing NPCs to make decisions in dynamic and unknown environments. Reinforcement learning has been used in games for a long time. Therefore, games are rich domains for testing reinforcement learning algorithms. At the same time, some of the best computer players use reinforcement learning (AlphaGo). However, the primary reinforcement learning algorithms are not sufficient for high-level game playing, so these methods are often used with other AI methods such as deep learning.

Overview of popular AI-based games

There are many examples of AI applications in game design. Each of these AIs has a different level of sophistication. Here are some examples of the most highly regarded AI in the gaming industry.

F.E.A.R.

F.E.A.R. is a first-person shooter and psychological horror game in which the main player engages with robots, various creatures, and cloned

supersoldiers. The game creators have developed AI that generates context-sensitive behaviors. For example, so-called Replicas can utilize the game environment to their advantage. Replicas can overturn tables to provide cover, open doors, crash through windows, or even alert the rest of their peers to the players' actions. Additionally, game AI can perform a flank attack, put out a fire, and throw grenades to force a player out of cover.

StarCraft II

Starcraft II is a real-time strategy game where players take a seat in a 1 vs. 1, 2 vs. 2, or 3 vs. 3 battle arena. The main aim of the players is to destroy their opponents' bases. This is done by creating units that are effective at defeating the opponents' units. Players can choose to play against various levels of AI from easy to Cheater 3. Starcraft's AI is capable of cheating to defeat human players by processing information about human player bases. Starcraft II as a game has also become a popular environment for AI research. In a joint push, Blizzard and DeepMind have released a public Starcraft II environment where scientists and enthusiasts can test various AI algorithms.

Alien: Isolation

Alien: Isolation is based on the Alien sci-fi horror movie series. The setting of this survival game is 15 years after the events of the movie, when Amanda Ripley, the daughter of Ellen Ripley (main character of the movie), investigates her mother's disappearance. Game developers utilize AI to measure the amount of stress the player is experiencing. AI measures three critical elements at all times: if the Xenomorph (Alien) can be seen by the player, distance between the Xenomorph and the player, and the

proximity of the Xenomorph to the motion tracker and how fast it can reach the player.

AI uses these three factors to determine the stress level that the player experiences. If the stress level is too low, it instructs the Xenomorph to move to a specific location closer to the player. If the level is too high, it moves the Xenomorph away from the player. In this way, AI creates ups and downs, a true characteristic of good horror.

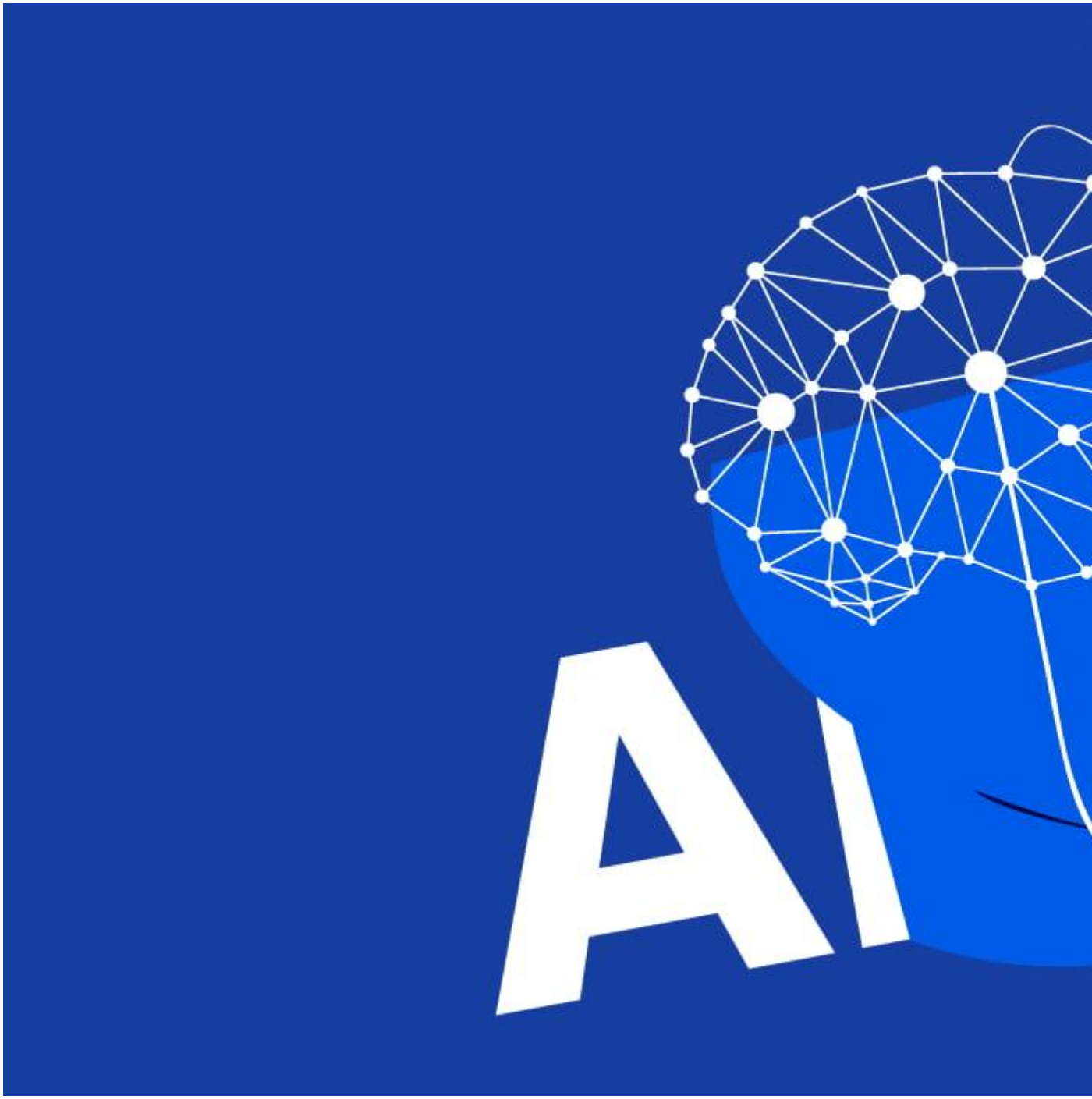
Forza Horizon Series

Forza Horizon is a simulation racing game that emulates real-world racing car performance

How Artificial Intelligence (AI) Upends Game Development

Published: 2 November, 2021

-
-
-



AI is becoming a universal tool for game design. Despite video game artificial intelligence still being in its infancy, game companies have already started to recognize attractive benefits such as enhanced player experience and cost reduction.

Ever since the first program that played chess in the ’50s, video games have been associated with artificial intelligence. The efforts to solve the challenge of computers defeating human experts in strategy games such

as chess, poker, and Go have significantly advanced AI research. In turn, this has led to enhancements in the design of new games.

In a broad sense, most video games incorporate some form of AI. Nevertheless, modern AI methods can be applied in distinct areas that help game companies realize business benefits as well. For example, through enhancing photorealistic effects, generating content, balancing in-game complexities, and providing intelligence to non-playing characters (NPCs), AI improves the overall user experience while saving game companies budget and time.

Read on to find out more about the importance and impact of AI in games, typical applications, commonly employed AI methods, the most popular games that utilize AI, the business benefits, current limitations in how AI can be applied, and the future of AI in games.

The importance and impact of AI in games

Game developers strive to deliver valuable interactive experiences to players. These experiences derive from the cumulative effects of a number of orchestrated game elements, including graphics, sounds, gameplay timing, narratives, challenges, and content that directly interacts with the players (allies, opponents, or other objects). Because of this, AI is becoming an unmatched tool that can help designers coordinate the ever-growing complexity of game dynamics.

The growing popularity of AI in games also has significant business benefits for enterprises. The gaming industry is becoming one of the most profitable sectors, with a market value predicted to reach around 314 billion

USD by 2026. As a result, funding of the development of AI-based games worldwide has been steadily rising.

Today, many startups are emerging in this area. For example, latitude, a startup that develops games using AI-generated infinity storylines, raised 3.3 million USD in seed funding in January 2021. Osmo, an interactive play company, has raised 32.5 million USD in funding so far. Gosu Data Lab, another AI gaming startup based in Lithuania, has raised 5.1 million USD in funding. Gosu mainly focuses on exploring gaming data for AI purposes, helping gamers get better at playing.

Typical applications of AI in games

The application of AI in games is diverse; it can be used for image enhancement, automated level generation, scenarios, and stories, balancing in-game complexity, and adding intelligence to non-playing characters (NPCs).

Image enhancement

At the forefront of image enhancement are the AI engineers who are making efforts to use a deep learning system that turns 3D rendered graphics into photorealistic images. Such a system has been tested on Grand Theft Auto 5. The developed neural network is capable of recreating LA and southern Californian landscapes in great detail. The most advanced image enhancement AI algorithms can convert high-quality synthetic 3D images into real-life-like depictions.

One application of image enhancement in video games is to improve the graphics of classic games. The main idea behind the algorithms proposed for this job is to take a low-resolution image and turn it into a version that looks the same but has many more pixels. This process is called “AI upscaling”.

Game level generation

Game level generation is also known as Procedural Content Generation (PCG). These are the names for a set of methods that use advanced AI algorithms to generate large open-world environments, new game levels, and many other game assets. This is one of the most promising applications of artificial intelligence in game design. Open world or open map games include some of the most popular games to date. These games allow players to explore vast landscapes. Creating such games is very time-consuming from both the design and development perspectives. But AI algorithms can build and optimize new scenery in relation to the game’s status. For example, No Man’s Sky is an AI-based game with an infinite number of new levels generated on the fly while you play.

Scenarios and stories

AI is used to generate stories and scenarios. Most often, AI is used to create an interactive narrative. In this kind of game, users create or influence a dramatic storyline through actions or what they say. The AI programs use text analysis and generate scenarios based on previously learned storylines. AI Dungeon 2 is one of the most famous examples of this application. The game utilizes a state-of-the-art open-source text generation system built by OpenAI and trained on the Choose Your Own Adventure books.

Balancing in-game complexity

The main advantage of AI algorithms is their ability to model complex systems. Game developers are continuously trying to create more immersive and realistic games. However, modeling the real world is complex. AI algorithms can predict the future effects of gamer actions, and even model things like weather and emotions to balance in-game complexity. A real example of this application is FIFA's ultimate team mode. FIFA automatically calculates a team chemistry score based on the personality traits of the players in a football team. Team morale oscillates from low to high based on the in-game events (losing the ball, making a well-timed pass, etc.) In this way, teams with better players can lose games against weaker teams because of their morale. In this way, AI can be used to add a layer of complexity.

Adding intelligence to non-playing characters (NPCs)

In most current games, the opponents are pre-programmed NPCs; however, AI is on the path to adding intelligence to these characters. This will make them less predictable and more enjoyable to play against. In addition, AI allows NPCs to get smarter and respond to the game conditions in novel and unique ways as the game progresses. Many gaming companies have already started working on AI-based NPCs. For example, SEED (EA) trains NPC characters by imitating the top players in games. This approach will profoundly reduce the development time of NPCs, as hard coding of their behavior is a tedious and lengthy process.

Get more insights into this blockchain battleship game powered by Echo smart contracts

AI methods used in games

Traditionally the behavior of NPCs was programmed using rule based and finite state machines. The development using these techniques involved programming many conditionals that gave NPCs deterministic behaviors. To reduce the development effort while introducing a degree of unpredictability in games, developers used fuzzy logic. One of the first uses of AI in games programming was through so-called A* pathfinding algorithms that define the behavior of NPCs and their exploration of open worlds. Other techniques include scripting, expert systems, and artificial life (A-life) approaches.

Many popular games such as Black & White, Battlecruiser 3000AD, Creatures, Dirt Track Racing, Fields of Battle, and Heavy Gear employed non deterministic methods such as decision trees, (deep) neural networks, genetic algorithms, and reinforcement learning methods. Let’s explore these techniques in detail.

Decision trees

Decision trees (DTs) are supervised learning models that can be trained to perform classification and regression. They are one of the most basic machine learning methods for game design, and can enable the value of a variable of interest to be predicted through learning simple decision rules inferred from the data features.

Decision trees are pretty simple to understand, and the results can be easily interpreted. Tree visualization techniques are also very advanced. The developed models are known as white box models and can be validated using various statistical tests.

In artificial intelligence game design, DTs are used to describe choices and consequences (predictions of actions). Most modern games use DTs, primarily narrative-based games. In one such use, decision trees can provide players with insights into how the future will look depending on their choices. For example, in Star Wars Jedi: Fallen Order, decision trees provide hints regarding the past and future of the main character if certain circumstances were to occur.

(Deep) neural networks

Artificial neural networks (NNs) are structures akin to human brains that can learn various features from training data. Given a large set of data, NNs are capable of modeling very complex real-world and game scenarios. NNs overcome some of the shortcomings of classic AI techniques in game agent design. Furthermore, NNs are self-adaptive and adapt well to game environments that change in real-time.

NN-based game agents can learn in two ways. Either they are trained before being deployed in a game (offline), or the learning process can be applied in real time during the gameplay (online). Online training allows for the creation of game agents that continuously improve while the game is being played.

NN-based agents can quickly adapt to the changing tactics of human players or other NPCs, and can make sure the game remains challenging even during extended gameplay.

Lately, Deep NN (deep learning) has become a more popular choice for game agent design. Deep learning in games utilizes multiple layers of neural networks to “progressively” extract features from the input data. Due to its layered approach and increased architectural complexity, deep NN

can achieve better results when controlling one or several game agents. These agents can either be NPCs or the game environment itself.

Genetic algorithms

In the most basic terms, a genetic algorithm (GA) is a higher-level procedure, a heuristic, inspired by the theory of natural evolution. The genetic algorithm mimics the process of natural selection, where the fittest candidates are chosen to produce offspring of the next generation.

GAs are used for various optimization tasks. When compared to different optimization techniques, GAs are capable of delivering excellent results for multicriteria optimizations. In the past, GAs found their place in board games that employ various search techniques when seeking the next best moves. The most recent applications of GAs to NPCs allow adaptation of these agents to defend against effective but repetitive tactics that human players may employ. The application of GAs leads to a more realistic game experience, where human players or other AI agents cannot find loopholes and dominate the game with repeated steps that always lead to success. The end benefit of GAs is extended playability.

Reinforcement learning

Reinforcement learning (RL) is a machine learning method that is based on learning from trial and error. During training, the model is allowed to play out scenarios and learn from whether things ended well or not so well.

Reinforcement learning is effective when designing NPCs to make decisions in dynamic and unknown environments. Reinforcement learning has been used in games for a long time. Therefore, games are rich domains for testing reinforcement learning algorithms. At the same time, some of the best computer players use reinforcement learning (AlphaGo).

However, the primary reinforcement learning algorithms are not sufficient for high-level game playing, so these methods are often used with other AI methods such as deep learning.

Take a look at this AI-powered retina disease diagnosis tool

Overview of popular AI-based games

There are many examples of AI applications in game design. Each of these AIs has a different level of sophistication. Here are some examples of the most highly regarded AI in the gaming industry.

F.E.A.R.

F.E.A.R. is a first-person shooter and psychological horror game in which the main player engages with robots, various creatures, and cloned supersoldiers. The game creators have developed AI that generates context-sensitive behaviors. For example, so-called Replicas can utilize the game environment to their advantage. Replicas can overturn tables to provide cover, open doors, crash through windows, or even alert the rest of their peers to the players' actions. Additionally, game AI can perform a flank attack, put out a fire, and throw grenades to force a player out of cover.

StarCraft II

Starcraft II is a real-time strategy game where players take a seat in a 1 vs. 1, 2 vs. 2, or 3 vs. 3 battle arena. The main aim of the players is to destroy their opponents' bases. This is done by creating units that are effective at defeating the opponents' units. Players can choose to play against various levels of AI from easy to Cheater 3. Starcraft's AI is capable of cheating to defeat human players by processing information about human player bases. Starcraft II as a game has also become a popular environment for AI research. In a joint push, Blizzard and DeepMind have released a public Starcraft II environment where scientists and enthusiasts can test various AI algorithms.

Alien: Isolation

Alien: Isolation is based on the Alien sci-fi horror movie series. The setting of this survival game is 15 years after the events of the movie, when Amanda Ripley, the daughter of Ellen Ripley (main character of the movie), investigates her mother's disappearance. Game developers utilize AI to measure the amount of stress the player is experiencing. AI measures three critical elements at all times: if the Xenomorph (Alien) can be seen by the player, distance between the Xenomorph and the player, and the proximity of the Xenomorph to the motion tracker and how fast it can reach the player.

AI uses these three factors to determine the stress level that the player experiences. If the stress level is too low, it instructs the Xenomorph to move to a specific location closer to the player. If the level is too high, it moves the Xenomorph away from the player. In this way, AI creates ups and downs, a true characteristic of good horror.

Forza Horizon Series

Forza Horizon is a simulation racing game that emulates real-world racing car performance

and handling characteristics. Forza employs a learning neural network in its design to control non-human drivers. The developed AI system can observe human drivers and imitate their style of driving. Under the name Drivatar, this AI system has recently been connected to Microsoft's cloud services, from which it gets driving data from a vast number of human racers. This data is used to create AI systems that mimic other players from around the world, not just their strengths but also their weaknesses, to provide unpredictable experiences for the competing human drivers.

The business benefits of using AI in games

AI brings a number of business benefits for game development companies. Companies that use AI in their games can save budget and time, provide a better user experience, and streamline their development processes.

Save budget and time

The gaming industry is one of those industries where a lot of budget and time are invested in development, i.e. while developing a game. In addition, there is always a risk that the audience may not accept the game. To avoid this, before a game is released to the market, it undergoes stringent quality assurance procedures and focus-group testing. As a result, a single game development process for a sophisticated game can sometimes take years.

AI is an indispensable tool allowing game companies to drastically reduce time and development budgets. When used for automated level generation, AI can save thousands of hours of development work. Furthermore, by employing data-driven techniques instead of hard-coded rules, AI eliminates the manual labor that would need to be invested otherwise. As a result, delivery costs can be reduced dramatically, meaning that game companies can hire better game developers to finish the job. This advantage is becoming increasingly important in a highly competitive job market for developers.

Better user experience

In the gaming business, the end-user experience is a critical success metric. User experience is an integrative component of the gaming business that determines sales volume, loyalty levels, marketing success, and many other business factors.

AI can make a game appear more sophisticated and realistic, sparking gamers' interest in playing and likelihood to recommend the game to others. For example, AI voice intelligence helps players understand their in-game actions better. It adds another level to the overall user experience by smartly engaging players' senses.

Another aspect of user experience that AI is enhancing is adaptation. For example, AI can make use of large amounts of personalized and privacy-protected data to create scenarios that certain types of gamers will enjoy the most.

Streamlined processes

> The streamlined process has fewer errors and delays. Game AI assists developers by automatically generating content such as

landscapes, levels, items, quests, and music. Once the AI development process is set in stone, human errors are removed from the picture. Delays in development can also be eliminated due to AI being very efficient at dedicated tasks.

Limitations in the use of AI in games

The applications of artificial intelligence in games have certain limitations. It is, for example, difficult to design realistic NPC enemies that can automatically produce an engaging level for each individual. AI-based NPC enemies are usually intended to respond in the best way to a player's moves. Such components are unbeatable but also predictable and quickly cease being fun.

Due to the ability of AI to predict possible future outcomes, AI can quickly become unbeatable. A famous example of such a scenario is a Tic-Tac-Toe game where AI implements a minimax algorithm that can lead to drawn games regardless of which move the human makes.
AI is often allowed to cheat when trying to defeat humans. The need to cheat, however, reveals the limits to achievable artificial intelligence. In games that require strategy and creativity, humans are generally able to beat AI. Since game artificial intelligence can still not learn from its own mistakes, the use of AI in such games is minimal.

Another limitation of artificial intelligence that does not only apply to gaming is the lack of context outside the training data. Again, this leads to ethical considerations and biases.

These limitations affect only a small set of game AI applications. The AI field is undergoing continuous improvement, and it’s likely that very soon these challenges will be successfully tackled.