Artificial Intelligence Review (2023) 56:14159–14191
https://doi.org/10.1007/s10462-023-10491-7

Non-player character decision-making in computer games

Muhtar Çağkan Uludağlı[1] · **Kaya Oğuz**[1]

Published online: 29 April 2023

Abstract
One of the most overlooked challenges in artificial intelligence (AI) for computer games is to create non-player game characters (NPCs) with human-like behavior. Modern NPCs determine their actions in different situations using certain decision-making methods, enabling them to change the current state of the game world. In this paper, we survey current decision-making methods used by NPCs in games, identifying five categories. We give detailed overview of these five categories and determine the previous studies that belong to each of these categories. We also discuss the hybrid methods which are the combinations of different decision-making methods and the frameworks that are created for NPC decision-making. As a result of this analysis, we create a taxonomy table based on these covered studies. Lastly, the challenges faced in our study and future possibilities for improvement are described.

Keywords Video game · Non-player character (NPC) · Artificial intelligence (AI) · Decision-making · Survey · Literature review

1 Introduction

Ever since the earliest titles, players have been interacting with autonomous agents in computer games, which are commonly referred to as non-player characters (NPC). As the name suggests, these are the characters that are not controlled by the players. The term NPC was first used for the characters in role-playing games (West 1996), but is now widely used for characters in almost all genres. NPCs populate the game world to provide an immersive game play experience (Ochs et al. 2009), and therefore, should be as realistic as possible in their looks, movements, dialogues, and decisions.

It should be emphasized that decision-making algorithms are used in many different contexts ranging from medical applications (Harper 2005) to autonomous cars (Receveur

Muhtar Çağkan Uludağlı and Kaya Oğuz have contributed equally to this work.

✉ Muhtar Çağkan Uludağlı
cagkan.uludagli@ieu.edu.tr

Kaya Oğuz
kaya.oguz@ieu.edu.tr

[1] Department of Computer Engineering, İzmir University of Economics, İzmir, Turkey

et al. 2020). Similarly, a computer game can employ decision-making on different levels depending on the game genre. Real-time strategy games, for instance, demand a higher level of intelligence to control the game's units, which must make decisions based on their current situations. Our survey focuses distinctly on the decision-making applications for NPCs, rather than decision systems that govern the game logic or a higher level of intelligence responsible for other entities in the game.

In computer games, decision-making enables NPCs to determine the actions taken with respect to both their internal states and their perception of the environment (Millington 2019). With the basic algorithms, such as finite state machines or decision trees are suitable for most games, it is possible to create more believable NPCs using algorithms specifically focused on NPC decision-making, such as behavior trees. Also, more sophisticated machine learning methods, such as deep learning, are now available. However, these methods require a lot of data to train and it is very challenging to test them against unforeseen situations (Justesen et al. 2020). These are excluded because there are no published studies that report their usage as a NPC decision-making method.

In games, most NPCs are neutral; they mind their own business in the background to give the illusion of a real town. Some NPCs are the enemy, and their task is to attack the players. Few NPCs are allies of the players, designed in such a way that the players bond with and care about them. For all these NPCs, decision-making algorithms are employed to ensure characters behave appropriately in both simple and complicated scenarios.

Decision-making in computer games must be in real-time, or in almost real-time, so that NPCs can react to the changes in the environment. For most algorithms, the process of taking a decision is almost instantaneous, but gathering and processing information for the decision-making system might take time in environments with many entities to take into consideration.

This study is an attempt to survey and categorize the approaches used for NPC decision-making in computer games. To the best of our knowledge, no existing survey focuses on the NPC decision-making. A comprehensive survey of existing studies, along with a taxonomy of NPC decision-making, has been provided for a broad view of the field.

The following section describes our literature review methodology for this study. Section 3 provides an overview of surveys related to decision-making algorithms in general. Section 4 gives definitions and survey the decision-making methods. The following section surveys the decision-making methods that consist of combinations of the methods given in the previous section. Section 6 presents the frameworks that are created for the demonstration of decision-making methods. Section 7 discusses our results and limitations, and the final section concludes the survey with our final thoughts and future directions.

2 Review methodology

This section discusses the methodology used to identify the studies published in academic databases and search engines. There are several methodologies, such as Kitchenham's Guidelines (Kitchenham 2004) and PRISMA (Moher et al. 2009) to conduct a systemic literature review. However, we have systematically followed the steps mentioned below to survey the field to the best of our ability. These steps are similar to the dynamic version of

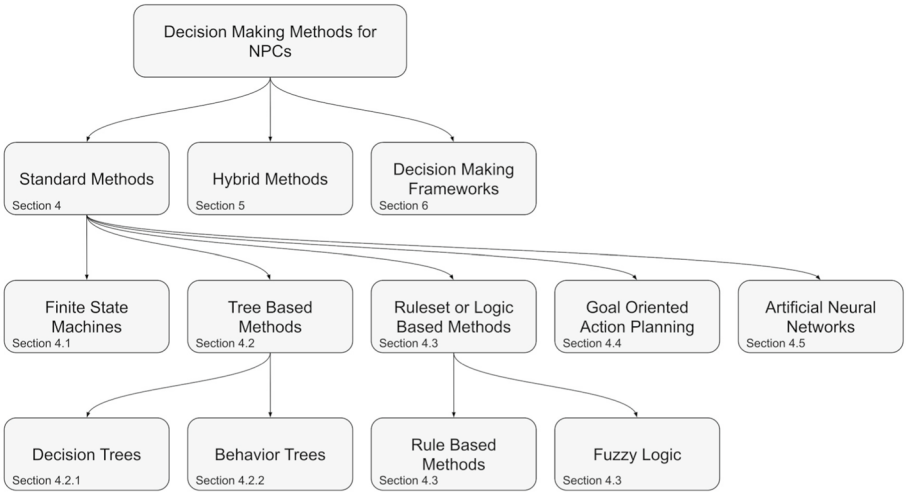

Fig. 1 The summarized layout of our study

PRISMA (Yanase and Triantaphyllou 2019), but differ in several key concepts so that we can get the best results to review this specific field.

Our inclusion criteria demand that the studies were published in peer-reviewed academic journals and indexed in academic databases. In this survey, to review the studies, we use established academic databases, *Scopus*, *IEEE Xplore* and *ScienceDirect*. Other than these databases, we also use the search engine *Google Scholar*, and academic network site *ResearchGate*.

A list of English words have been collected from high-level topics such as *Interactive Agents* or *Autonomous characters* to specifics such as *Behavior Tree* or *Goal oriented action planning NPC computer games*. A complete list of keywords has been made available as a supplementary material to this article.

While we aimed to cover as much ground as possible, some exclusion criteria were specified to limit the focus on decision-making methods for the NPCs in computer games. We excluded studies that did not conform to the following:

1. Studies that include a decision-making method used in a context other than computer games.
2. Studies that have a computer game context, but in which the decision-making method discussed is not applied to the NPCs or to the game environment.

A few exceptions had to be made to these criteria for studies with a historical significance, and when the study provided a clear explanation of the method in question.

Our search efforts resulted in 449 studies. We carefully investigated these and eliminated those failing to meet our criteria. After this analysis, we included 106 studies in total, of which 6 contained introductory explanations, 3 were in this section, 10 were related surveys and the rest were original studies covered in this survey.

The decision-making methods for NPCs can be grouped into the categories shown in Fig. 1. Naturally, it is possible to use these methods simultaneously, or with different

complementary methods to create a *hybrid* method. There are also *frameworks* that provide a complete solution to the decision-making problems in a computer game. Even though they use the decision-making methods presented earlier, we believe that their approaches merit a category of their own.

3 Related surveys

There are several surveys on decision-making algorithms that make incidental comments on their use in computer games. Some surveys focus on a specific decision-making algorithm, some cover general artificial intelligence algorithms for games, and others focus on general decision-making in computer games. Table 1 lists existing surveys with respect to the algorithms covered. None of these surveys focuses exclusively on NPC decision-making algorithms.

In their study, Kaelbling et al. (1996) surveyed some practical applications of reinforcement learning, including game play and robotics. This is one of the first articles that reviews reinforcement learning in the context of computer games by implementing an AI agent able to play backgammon. Therefore, the study has been included for its historical importance, but has been excluded from the table.

The survey by Cavazza (2000) is one of the first that reviewed rule-based systems by referring to finite state machines for NPC decisions, as well as path planning, search algorithms and storytelling.

Sweetser and Wiles (2002) surveyed the artificial intelligence methods used in games, such as finite state machines, scripting, fuzzy logic, decision trees, neural networks and genetic algorithms. The paper introduces the methods, describes how they are applied in games, and comments on possible future use.

In their paper, Charles and McGlinchey (2004) reviewed effective implementations of artificial neural networks in game AI. They accurately predicted the future directions for fields to which the neural networks could be applied, such as online learning, player modeling, intelligent character animation and behavior prediction.

Pirovano (2012) surveyed fuzzy logic usage in games, taking examples from industry and the academia. The paper highlights the benefits and the drawbacks of fuzzy logic and explores areas of application.

Sekhavat (2017) reviewed the behavior trees in computer games and presented a taxonomy of their properties. He describes strengths and weaknesses of different behavior tree creation methods and indicates how these trees are used in the game industry.

Neufeld et al. (2019) surveyed the literature for the usage of planner systems in computer games. They included articles on different methods that make use of goal-oriented action planning. To help developers through the process of implementing a planner for their NPC decision-making processes, they identify the key areas of plan creation and execution, and discuss the potential fixes for issues arising in different application areas. They also detail the various options for resolving these issues and discuss the particular situations for which each option is most appropriate.

Zijie et al. (2021) presented research on the fundamental ideas, current development patterns, and cutting-edge technologies on behavior trees. They also examined how behavior trees are used and developed in various industries, in addition to explaining their usage for NPC decision-making in games.

Table 1 An overview of the literature surveys

Title	Year	FSM	DT	BT	L/R	GOAP	ANN
Al in computer games: survey and perspectives (Cavazza 2000)	2000	✓			✓		
Current AI in games: s review (Sweetser and Wiles 2002)	2002	✓	✓		✓		✓
The past, present and future of artificial neural networks in digital games (Charles and McGlinchey 2004)	2004						✓
The use of fuzzy logic for artificial intelligence in games (Pirovano 2012)	2012				✓		
Behavior trees for computer games (Sekhavat 2017)	2017	✓		✓			
Building a planner: s survey of planning systems used in commercial video games (Neufeld et al. 2019)	2019					✓	
A survey: development and application of behavior trees (Zijie et al. 2021)	2021			✓			
A survey of behavior trees in robotics and ai (Iovino et al. 2022)	2022	✓		✓			
Our survey	2023	✓	✓	✓	✓	✓	✓

FSM finite state machines, *DT* decision trees, *BT* behavior trees, *L/R* logic-based or rule-based algorithms, *GOAP* stands for goal oriented action planning, *ANN* artificial neural networks

In a recent study, da Silva and de Souza Ribeiro (2021) surveyed the literature in a systematic method for the NPC AI development with believable behaviors. They addressed five research questions in their survey and they found that the game industry's use of some specific methods for the NPC decision-making unintentionally created a significant barrier to producing more engaged NPC behavior in virtual game environments. This study is also omitted from the summary table because it does not survey the decision-making methods, but rather the processes used in the game development.

Most recently, Iovino et al. (2022) comprehensively surveyed behavior tree usage in games and robotics and created a very detailed taxonomy. They categorized behavior trees from different points of view, such as the application type of the behavior tree algorithm, type of game in which the trees are used, and robotic systems for which behavior trees are used.

In contrast to existing surveys, this study focuses on all available decision-making algorithms with respect to their application for the NPCs in computer games. Some of these are created solely for NPCs and others are decision-making methods also used in other scenarios. The study provides a categorization of these algorithms along with their usage on different computer game genres. Finally, the current study not only gives fundamental summaries of these algorithms, but also discusses recent advances and methods in the field.

4 Decision-making methods

NPCs are part of the game and require a mechanism to sense their environment. In the methods discussed, we assume that the game provides NPCs with these mechanisms. For example, to enable the NPC *see* the player, there are two possibilities: either the code for the NPC may query an interface for the *game world* to get a list of entities in a certain distance, or the NPC could use techniques like ray casting to detect the objects around it. We assume that these mechanisms are provided in the game software, and will not discuss them unless they are integral to the decision-making method.

4.1 Finite state machines

A finite state machine (FSM) is an abstract machine that can exist in one of several different and predefined states (Bourg and Seemann 2004). In early days of game development, they were sufficient to provide a decision-making mechanism for the NPCs. Titles such as Pac-Man and Sonic implemented FSMs to guide their characters depending on their current states; either to chase or flee from the player. FSMs were most commonly used because of the elegant characteristics of the method and the computational limitations of the early computing devices. However, FSMs are still used in more recent games, either on their own or as a supplementary method.

FSMs are a common sight in automata theory. Formally, an FSM can be defined as a 5-tuple $(\Sigma, S, s_0, \delta, F)$ where Σ is the input alphabet, S is a finite set of states, s_0 is the starting state ($s_0 \in S$), δ is the state-transition function as $\delta : S \times \Sigma \rightarrow S$ and F is the finite set of final states ($F \subseteq S$). Their implementation in games follow the same principles, such as being in one state at a time. An FSM contains a set of states and conditions that allow the machine to transition from one state to another. Unlike their automaton counterparts, they do not produce a reject or an accept response. The FSMs in games are assigned to

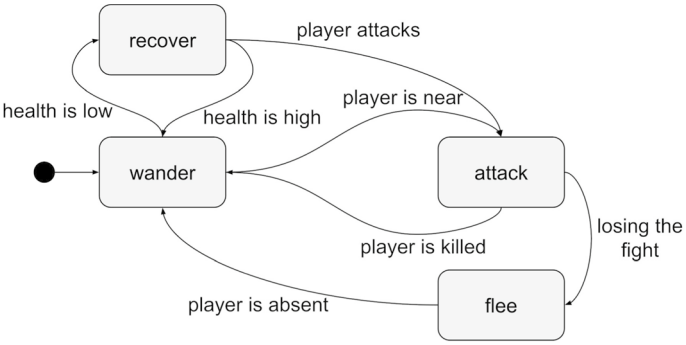

Fig. 2 A sample finite state machine for an NPC with four states and seven transitions. In this scenario, when an NPC comes across the player while wandering, it will attack the player. If the player is winning the fight, the NPC runs for its life

game entities, which includes NPCs, and allow the entity to behave accordingly depending on their current state. When a condition is met, the machine can transition to another state. Once the entity is removed, as in the death of an NPC, the state machine is also destroyed.

Figure 2 contains a sample FSM for an NPC that contains four states: $S = \{wander, attack, flee, recover\}$; and seven conditions: $\Sigma = \{$*health is low, health is high, player is near, player is killed, player attacks, player is absent, losing the fight*$\}$; for when the NPC should transition to another state. The smaller black circle indicates the initial state: s_0. Once the state machine begins working, that is, when the NPC is created, it starts in the *wander* state, and the NPC behaves accordingly. When the NPC senses the player, which is indicated by the condition *player is near*, it transitions to the *attack* state. If the fight is won, and the player is killed, the NPC returns to the previous *wander* state. If, on the other hand, the NPC is *losing the fight*, it might choose to *flee*. The remaining states and their transitions can be followed from the figure. In each of these states, the NPC has the opportunity to include actions related to the state; for example it seeks health potions when in the *recover* state.

FSMs have been in use in several games. The conceptual FSM for the Ghosts in Pac-Man had two states, *Chase* and *Evade*, with two transitions between them. If the player eats the power pill, the ghosts will escape from the player, hence the FSM goes to the *Evade* state; and if the power pill duration expires, then the ghosts will continue to chase the player, hence FSM goes to the *Chase* state.

Half-Life had many NPC types, such as Scientist, Headcrab and Soldier all of which have been extended from the Monster types. The game used FSMs for their decision-making processes with states like Idle, Dead, and Moving. The transitions were handled by 32 different conditions, as in *see enemy*, and *hear sound*. The NPCs behaved according to the predefined tasks in game to achieve their goals by using these state transitions with respect to their conditions in the game. It is one of the games that fully used FSM capabilities without needing support from more modern techniques. The NPCs in more recent games, such as Wolfenstein and Doom, also use FSMs.

The literature on FSMs for NPC decision-making is limited (Cavazza 2000; Nareyek 2000; Fu and Houlette 2002). One of the games that implemented FSMs is in the study by Laird (2001), in which an extension of the FSM concept was used to implement more human-like NPCs. More recent studies used FSMs for NPCs in various games, including

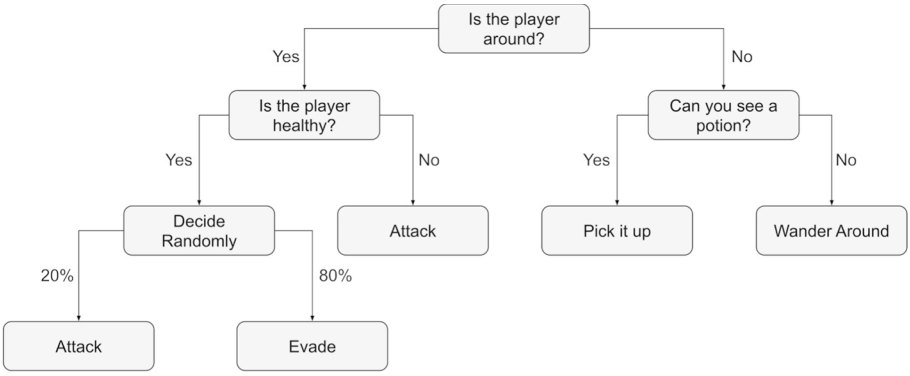

Fig. 3 A sample decision tree. If the player is around and healthy, then with a chance of 20 percent, NPC will attack to the player; and with a chance of 80 percent, will evade the player. If the player is not around and the potion is around, then NPC will pick it up

those with a historical theme (Syahputra et al. 2019), in character recognition games (Fathoni et al. 2020), in strategy games that have defensive tactics (Fauzi et al. 2019) or 2D character mimicking games (Sindhu et al. 2022).

One disadvantage of using FSMs is that the number of transitions can grow exponentially as the number of states increases. This can be avoided in some capacity by using Hierarchical FSMs (HFSMs) as discussed in Harel (1987), however, this is not a complete solution to this problem. There is another study which evaluates FSMs and HFSMs for comparison (Fauzi et al. 2019). Despite the efforts to solve the problems they cause, FSMs also tend to behave inaccurately in certain conditions, such as having incomplete states or transitions, and also may have performance issues due to excessive state usage. Such limitations in FSMs led the industry to explore alternative methods for implementing the decision-making of NPCs.

4.2 Tree-based decision-making

4.2.1 Decision trees

Decision trees (DTs) are algorithmic trees that can be characterized as a formalized set of nested if-then-else rules. According to Millington (2019), they are the simplest decision-making technique, being modular and easy to implement.

Similar to the regular tree abstract data type, a DT is made up of nodes, each of which may contain zero or more child nodes. Nodes with no children are called leaf nodes. In a DR, the root node is called the *start node*; nodes with decisions, including the start node, are called *decision nodes*; and the leaf nodes are called the *end nodes* which include actions executed once they are reached (Millington 2019).

The basic DT algorithm starts evaluating decisions from the root. Each decision directs the algorithm to the next node until an action in a leaf node is executed. Figure 3 shows a sample DT with a total of nine nodes. At the root node, depending on the answer to the decision in that node, the algorithm moves on to the *Yes* or *No* nodes. If *the player is around* and if *the player is healthy*, the DT reaches the node where a random action has

to be decided. The tree selects the *Attack* action with 20%, and the *Evade* action with 80% probability, in their respective end nodes. This probabilistic approach introduces *chance nodes*, discussed in Danielson and Ekenberg (2011).

DTs are commonly used for decision-making processes of NPCs in both industry and academia. For one particular example, Mas'udi et al. (2021) employed the NPC decision-making using DTs for a 3D kart-racing game developed by them. Quadir and Khder (2022) also used DTs in their study, in which they discuss the progress gained in creating a responsive foe based on DTs, along with some prominent past instances and potential applications for such AI agents in the future.

Lie and Istiono (2022) employed DTs for their study that includes a fighting game. Implementing an adaptive AI is a method to solve the repetitive action problem in fighting games. Because of this assumption, researchers used an adaptive AI system to create a fighting game, and they assessed the player satisfaction rate with the resulting game. Results reveal that, the satisfaction rates of the users are optimal, and thus using their approach can be beneficial in this game genre.

Another game that used the advantages of DTs was Black and White (Yannakakis and Togelius 2018). Wexler (2002) who analyzed this game in his study states that the DTs in the game represented the beliefs of NPCs about general object types. The main NPC in the game is a very large beast who learns to distinguish between good and bad by receiving from the player either encouragement, which is mapped to the game as gently petting the beast, or punishment, mapped as a harsh slapping. This creates the problem of a dynamic decision tree that must be updated depending on the player's input. To generate these DTs, the game uses the Iterative Dichotomiser 3 (ID3) algorithm by Quinlan (1986) which employs a top-down greedy approach.

These predefined DTs, however, can result in known problems according to Lara-Cabrera et al. (2015) such as the feeling of unreality, the occurrence of abnormal behaviors in unexpected situations, or the existence of predictable behaviors that were unintended by the designers of the tree.

4.2.2 Behavior trees

A behavior tree (BT) (Loyall and Bates 1991; Mateas and Stern 2002) is a tree of hierarchical nodes that control the flow of decision-making of an AI agent. The leaves of the tree are the actual commands that control the AI entity. Other branches in the tree includes different types of utility nodes that control the AI's tree traversal to reach the sequences of commands according to the situation. A BT consists of several types of nodes, but the core functionality is common to any type of node in a tree. They can return one of three statuses: *success*, *failure* or *running*. A simple categorization of BT nodes and tasks is in Table 2.

All BTs start their execution from the root node. The nodes in the tree create signals that are called *ticks* with regular intervals. They send these ticks to their children to process the nodes. The child node returns *running* as the result of the current tick to its parent node if its execution continues. It returns *success* if the node achieved its purpose, otherwise returns *failure*. Sequence nodes execute its child nodes from left to right with a preorder traversal. If all the child nodes return *success* to a parent sequence node, then the sequence node can continue its execution, such as passing to the next node in the same level of the tree or returning a result to its own parent node. If the tree has selector (fallback) nodes,

Table 2 Behavior tree: node, task and result types

Types	Names	Behavior	Symbol
Composite nodes	*Sequence*	Behaves like an AND gate	→
	Selector (Fallback)	Behaves like an OR gate	?
Decorator nodes	*Inverter*	behaves like a NOT gate.	
	Succeeder	always return success result while processing this branch.	
	Repeater	reprocess its children each time they return a result.	
	Repeat until fail	reprocess its children until they return failure result	
Leaf nodes	nodes that implemented the actions of AI.		
Result types	*Success*	*Failure*	*Running*

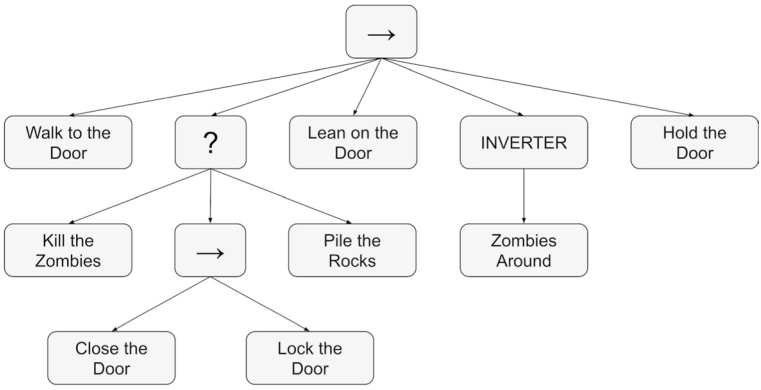

Fig. 4 An example behavior tree. If NPC can walk to door, then it must carry out at least one of the actions in the selector branch. If it kills the zombie, then it returns success to the parent selector node. Or if it closed and locks the door, then it returns success to the parent selector node also. If it wants to carry out all of the sequence, then it must lean on the door, and when zombies are not around, it must hold the door

then they tend to behave differently than sequence nodes. If any of the children of a selector node returns success, then that selector node also returns *success* to its own parent node. It should be remembered that if a child of a sequence node yields a *running* or *failure* result; or a child of a selector node yields a *running* or *success* result, then these nodes do not send any ticks to the next child. Their execution flow finishes at that step.

A simple example of a BT usage scenario is shown in Fig. 4. The root node is a sequence, denoted by →, so all its children must return true. If the NPC can walk to door, then that action returns *success* to the root node. Then, the execution of other children of the root node can continue with the next node. The next node is a selector node, denoted by a question mark. If any of the actions in its children can be completed successfully in that branch, such that NPC can kill the zombies, or close and lock the door, or pile the rocks, then one of those children will return *success* to the selector node. After that, if the NPC can lean on the door and hold the door while zombies are not around, then these three nodes also return *success* to the root. When the *success* results come from every child of the root node, the execution of this BT is successfully completed. If any child of the root

node returns *failure* (e.g., suppose there is a huge rock on the walk path and the NPC cannot walk to the door because of that), then it returns *failure* to the parent node. Hence, the execution of this BT halts.

According to Millington (2019), BTs have become a popular tool for developing the decisions of NPC characters. Halo 2 (Yannakakis and Togelius 2018) was one of the first modern games that used BTs for the decision-making process of NPCs and since then, many more games have used it, too. Isla (2005) describes the methods employed to model in-game NPC behavior using BTs. He et al. in his company implemented a hierarchical BT by employing a directed acyclic graph (DAG) as the underlying data structure. The actual core behavior DAG of Halo 2 contains about 50 different behaviors. While Johansson and Dell'Acqua (2012) compared BTs with emotional behavior networks in their study, Simonov et al. (2019) embedded personality characteristics of NPCs to decision-making process to create a more believable game AI. These last two examples clearly show that using BTs in a combination with NPC character properties or conceptual emotions is a common scenario for decision-making in games.

Holba and Huber (2021) explained the NPC decision-making system used in Mafia III game. Some important aspects of this system are perception, position selection and the cover system. They mainly used BTs for NPCs, but with some supporting systems for the previously mentioned core aspects. Panwar (2022) discussed the difficulties the developers faced when developing the AI for the game The Last of Us. The NPCs, as well as the decision-making strategies they employed to overcome the difficulties they faced are thoroughly investigated. Additionally, the study analyzed the strategies used by other games from the leading companies in the industry.

In their book, Colledanchise and Ögren (2018) described how they created a Pac-Man playing agent from scratch using BTs. They report that the details of the AI lies in considerations of the maze geometry, choosing correct paths, and possible capture by the ghosts. They concluded that selecting the correct functionalities to address in the BTs needs further investigation, and should be decided by using various cases.

Lin et al. (2019) discussed the possibility of BT usage in the AI design of multiplayer online battle arena (MOBA) game genre. In order to assist game designers in creating games that satisfy the needs of players, they examine the function of BT in the decision-making mechanisms of MOBA game AI. They accomplished this by investigating the features of MOBA games and the advantages of incorporating hierarchical logic through BTs. According to them, the most sophisticated, reliable, and simple to update decision-making method used in MOBA game AI design is the BT.

BTs are also used in different game genres, such as role-playing games (RPGs). One such example is implemented in the research from Rodrigues et al. (2021). They created a choice-based story game with using Unreal Engine 4. By utilizing BTs, the game also provides a dynamic difficulty setting and an adaptive AI. Their system evaluates player performance and adapts the AI of the game accordingly.

BTs can be used in first-person shooter (FPS) games and virtual reality (VR) games. The study from Pyjas et al. (2022) used BTs in an FPS VR game set in the World War II era. They employed BTs to enhance the user experience in their VR environment.

In Miyake et al. (2019), the authors explained that they used BTs and FSMs together to create a decision-making tool for NPCs in the game Final Fantasy XV which they call *AI Graph Editor*. They consider that this tool allows game developers to take advantage of both methods in a nested hierarchical node architecture.

Different from these examples, BTs can also be used with a stochasticity concept. The term stochastic BTs (SBTs) (Colledanchise and Ögren 2018; Colledanchise et al.

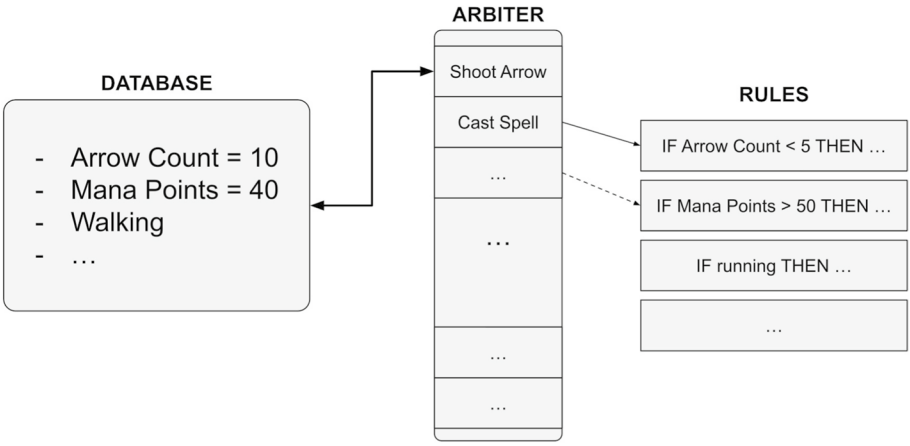

Fig. 5 A sample schema for rule-based decision-making. The arbiter is an intermediate process that checks the rules according to the information in the database to complete certain actions. In this example, the first rule will not be fired at this moment to complete *Cast Spell* action, since *Arrow Count* is 10 at the database and the first rule states that it must be less than 5

2014) refers to the BTs that add a probability for the outcomes of actions in the tree. The probability of the outcomes are known in advance for the actions, and with a certain probability, as the result of that action, one or the other outcome returns to the parent action node in the tree.

With recent technological developments, FSMs, DTs and BTs for managing decisions of NPCs are deficient in some situations, such as bad performance on overgrown tree structures, repeating mistakes without learning, and selecting the same decisions without being adaptive to different conditions, creating the need for more believable character management methods. Researchers use complementary techniques such as employing reinforcement learning or genetic algorithms to overcome these shortcomings in earlier methods. We present such example usages in Sect. 5.

4.3 Ruleset- or logic-based decision-making

Rule-based systems (RBSs) were used on AI research in early years of the industry. According to Millington (2019), usage of RBSs is not a common approach, since implementing such a complex rule-based system is less efficient and harder to implement than FSMs or DTs.

The organization structure of RBSs includes three elements: *an information database*, *a rule set*, and between these, *an arbiter* as shown in Fig. 5. The database contains the current information known by the AI agent about the environment and other characters. The rule set contains a set of *if-then-else* rules for the execution of the actions. According to the decision made by the arbiter, one of the *if* rules in the rule set is triggered to be fired and the action at the *then* component is performed at that moment.

Most of the AI systems use the Rete matching algorithm to match the correct rules according to the correct information on the database for RBSs, as detailed in Forgy (1979, 1989). Two games known to use the rule-based decision-making algorithms, such as Rete

for controlling the behaviors of the NPCs, are Baldur's Gate (Yue and de Byl 2006) and Virtua Fighter (Yue and de Byl 2006).

Fujii et al. (2008) examine RBSs in a car racing game. They experimented on two different sensory input methods that generate a set of patterns for constructing these systems, namely, egocentric and third-person sensory information. They compared the performance of four types of RBSs in this game and completed an analysis on different sensory input from these virtual cars. They found that, third-person sensory information of the cars may be used for decision-making in these systems. They also employed FL in these RBSs, a topic further explained in the next part.

As previously mentioned, the other method for decision-making by using rules is fuzzy logic (FL). FL is a form of logic in which the truth-values of the given variables can be real numbers between 0 (false) and 1 (true). These truth values are called the *partial truth values* and their values, such as *half true* or *nearly false*, can be defined with this usage (Novák et al. 1999). FL usage in game AI was first described in O'Brien (1996) and it is widely used in both the academic and industrial game community.

FL is applied to games by changing the values of different variables to fuzzy values, at first, with a procedure called *fuzzification*. A set of membership values is set for every variable in the decision-making for their possible ranges, which give a degree of truth for these variables. As an example, consider a scenario in which a character has health points (HP) which range from 0 to 100 for its likelihood of survival. The fuzzification can include three conditions; *near death*, *half-healthy*, and *healthy*, and the values between 0 and 100 should be matched to these conditions with varying degrees. For an HP value of 40, the membership to *near death* could be 0.1, to *half-health* could be 0.8, and to *healthy* could be 0.1. According to these values, it is seen that the character is closer to the *half-healthy* than the *near death* condition. Remember that the sum of the degrees of truth for the intermediate points such as this example must always be equal to 1.0.

The fuzzy rules are usually similar to if-then-else rules in Boolean logic for respective values. Fuzzy variables and their sets must be defined to create these fuzzy rules. After the rules have been defined, a *fuzzy inference engine* deduces the fuzzy outcomes with given rules and the values of variables with respect to these rules. In the final phase, with a *defuzzification* procedure, which is the reverse of fuzzification, all the fuzzy output values are converted to actual results.

The FL can also be combined with FSMs to create Fuzzy State Machines (FuSM). Existing FSMs can easily be transformed into FuSMs if needed, bringing the benefits of both approaches. FSMs can be converted to FuSMs in three different ways (Millington 2019): changing transitions to fuzzy transitions, changing states to fuzzy states, or adding FL to particular states without changing other states in FSM.

Many games used FL as one of their decision-making methods. Unreal (Johnson and Wiles 2001), Civilization: Call to Power (Johnson and Wiles 2001) and Close Combat (Sweetser and Wiles 2002) series all employed FuSMs as their decision-making methods for NPCs, while the game S.W.A.T. 2 used FL to enable their characters to behave according to their given personalities (Sweetser and Wiles 2002; Johnson and Wiles 2001).

There are numerous articles covering FL usage in game AI. Shaout et al. (2006) created a Pac-Man clone game in which all the decisions of the enemies are made by FL. They have acquired the ratings of players that play their version, and they compared their game with non-FL decision-making methods. They reported that overall performance of NPC agents produce satisfying results. Van Waveren (2001) also used FL in his thesis for developing the decision-making of NPCs in the game Quake III Arena clone. NPC bots, developed specially for his game clone, use FL to control item and weapon selection.

Li et al. (2004) used belief-desire-intention agent theory for an earlier arcade game called BattleCity, and utilized an FL-based framework for enemy NPCs to increase the details of their decision-making process. As inputs from the NPC tanks, their framework takes the last attack time, shield level and visible enemies' strength values, and gives fuzzy outcome outputs as information to be used by the player's tank.

Soylucicek et al. (2017) also used FL for the decisions of drones in a space-themed 2D arcade game called Meteor Escape. They used a radar-like structure that calculates the distance between the drone and the player character. If the spaceship of the player is in a certain range, the drone can decide to shoot itself to explode on player, according to the fuzzy rules. Researchers reported that an FL based NPC AI can make the games more challenging.

There are more complex usage examples of FL in decision-making for NPCs. Niewiadomski and Renkas (2014) created a new fuzzy controller type as hierarchical fuzzy controller for the purpose of controlling vehicles in computer games. In their method, fuzzy outputs on a higher level at hierarchy are used as inputs for another fuzzy controller in a lower level at hierarchy. They applied their method to control virtual vehicles in a 2D game and comparison with a classical fuzzy controller shows better results for their method.

Ohsone and Onisawa (2008) created a DT with FL to determine if a partner NPC in a poker game can use facial expressions to give advice to the player character about the hand of the opponent NPC. Fuzzy decision tree part is used for the decisions, and neural networks, for the facial expressions of this partner NPC. In the experiments with human subjects, the usefulness and the friendliness of the partner NPC was evaluated and the proposed system was found successful in both of these aspects. In Abiyev et al. (2016), researchers combined the BTs with FL to present a decision-making method for the robot agents that play a football game simulation. They used behavior trees with fuzzy inputs for general decisions such as moving, passing or shooting the ball, and the FL for obstacle avoidance part of the agents. They tested their algorithm on simulations, finding satisfying results in both the decision-making and obstacle avoidance aspects.

The use of FL in decision-making for NPCs has some drawbacks, such as the need of correct definition of input and output variables, high computational cost for complex systems and due to this reason, a combinatorial explosion of complexity (Zarozinski and Than 2001). According to Millington (2019), FL usage in academic community is mainly discarded because the probability is better at modeling and representing all uncertainty problems than FL.

4.4 Goal-based decision-making

NPCs in computer games may have clear objectives and goals to complete. *Goal oriented behavior* is when the NPC chooses one of possible actions from a set that drives the NPC closer to its goal. The base of goal oriented action planning (GOAP) is when multiple goal oriented actions are taken in a sequence to reach an overall goal in the long run (Millington 2019; Orkin 2003). GOAP is based on Stanford Research Institute problem solver (STRIPS), which has the same underlying elements to solve goal-based problems (Russell and Norvig 2002).

For a GOAP system, there are two essential elements: *the goals* and *the actions*. There can be multiple goals of the NPCs in the game, and every goal has a level of priority based on importance. The possible actions that a character can make is mostly predefined and finite. The actions can be related to the objects in the game such as shooting *a weapon*,

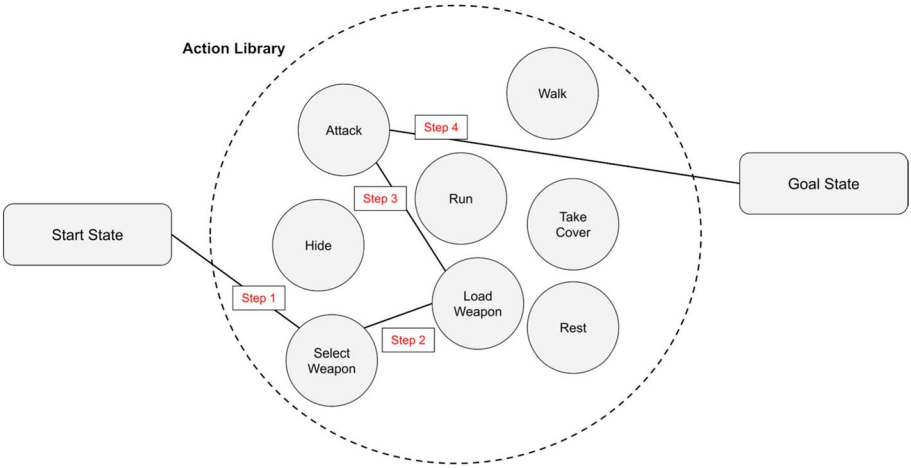

Fig. 6 A sample action library and action steps for GOAP. According to this example, NPC selects certain actions with orderly fashion from the library to fulfill its goals. This NPC selects *Select Weapon*, *Load Weapon* and *Attack* action states in this order and reaches its goal state, which we can guess that it may be *Defeat Enemy* state

sitting on *a chair*, lifting *a book* and so on. The NPC aims to achieve its current goal by using the possible actions at hand. Its ultimate goal can be reached by carrying out the actions most beneficial for its intermediate goals.

Generally, a process is responsible for the planning of the NPC actions in a GOAP system. This planner looks for the applicable actions and uses these actions to create an ordered series. If this series of actions yields to a goal state, then the NPC can choose to act on these actions to reach its goal. However, when a more urgent goal emerges for the NPC, the planner can formulate a new series of actions to reach this (See Fig. 6).

The games such as SIMS (Millington 2019), F.E.A.R (Orkin 2006), Condemned: Criminal Origins, S.T.A.L.K.E.R.: Shadow of Chernobyl (Long 2007), S.W.A.T. 4 (Pittman 2007), Deus Ex: Human Revolution (Sloan 2015) all use goal/action paradigm in some capacity for decision-making; and some titles actually used GOAP for decision-making of NPCs. Orkin (2006) describes how they used GOAP for the game F.E.A.R in his work, and dedicated an entire website [1] to his and others' GOAP research in the area.

Orkin (2006) explains that in the game F.E.A.R., they used an FSM with only three states for the movement of the NPCs. However, for the interactions and planning of the NPCs, they created a goal-based system that transitions into the correct state using the knowledge at hand. He used this system as a complementary method to the FSM, and states that different action sets were used for the same goal for different NPCs. This method, he argues, has some advantages, such as separating goals and actions from each other, managing complex behaviors by using simple behavior sub-layers and giving dynamic problem solving capabilities to NPCs. He refers their newly founded goal-based system as GOAP, and lists its three main differences from the STRIPS as follows: adding costs to actions, using fixed-size arrays to represent preconditions and effects, and adding

[1] http://alumni.media.mit.edu/~jorkin/goap.html.

procedural preconditions and effects. He asserts that using GOAP in games is an effective solution for creating cooperative NPCs and social behaviors between NPCs.

Long (2007) analyzed GOAP and compared it with FSMs, which is one of the traditional decision-making methods at the time of his dissertation. He compared two methods according to different aspects. For both methods, he created a game environment to test different game modes, such as domination, capture the flag, deathmatch and last man standing. The other two aspects, which he covered for comparison, are the technical efficiency and usability of both methods. His experimental and performance results expressed that GOAP is a better decision-making method than using FSMs. However, he also points out some of the key factors for using GOAP safely in computer games, such as managing the intelligence of the NPCs according to the player, and presenting the real significance of the method from the standpoint of the player rather than the developer.

According to Sloan et al. (2011), GOAP has some weaknesses, such as incapability of evaluating the full outcome of an action plan and incapability of defining more than one goal. They created a new approach called Utility-Directed Goal-Oriented Action Planning to overcome these weaknesses, and compared it with an environment similar to the game Sims. Their evaluation criteria include the time that an NPC can reach an ideal state, the total memory needed for their method, and the complexity of the NPC behaviors achieved by the system. They considered that their new method could be a promising alternative to GOAP. Sloan (2015) added another concept, called *smart ambiance*, to his earlier research. Smart ambiance allows allows the use of the context information on the environment to the NPCs' benefit as they make their decisions according to this known context. In his dissertation, he demonstrated different perspectives of this concept. One of the advantages of the smart ambiance is the selection of more suitable behaviors, compared to its predecessors.

Studiawan et al. (2018) developed a real-time tactical space game to find if GOAP will improve performance when used in this game type. They evaluated the runtime complexity and performance of NPC behaviors for their proposed approach, which resulted in a reduction in complexity, but they advised the AI designers to carefully adjust the connection between parameters and the actions for changing conditions. Suyikno and Setiawan (2019) evaluated GOAP usage in particularly the hide-and-seek behavior of the NPCs for the stealth-oriented game genre. Describing the development of the set of tactics for hiding task as difficult, they emphasized that the NPCs needed to exhibit different complex behaviors. They used some playtesting scenarios (e.g., using two NPCs at max. in one hiding place or limitless placement) with hiding behavior of NPCs to assess their approach. NPCs using GOAP in limited placement scenario was more successful than their alternative scenario.

The study from Johansen et al. (2022) investigated the control of NPCs in an interactive storytelling game engine using domain-independent classical planners. Their GOAP model considers the characters' emotional states while determining the best course of action, producing realistic agents who operate in a goal-driven way, and have their own objectives. They demonstrated that for situations with fewer characters, their GOAP model could discover optimal or almost ideal solutions in the order of milliseconds.

GOAP can also be developed to use it in different game engines or development environments. One example is Sielicki et al. (2018), who created GOAP modules in Unreal Engine, and reported a real shortage on effective GOAP applications and frameworks which can be used by regular developers on the most known game engines. Therefore, they

decided to develop one, implementing the essential elements of GOAP, using the engine's features, and evaluated the proposed framework's performance.

GOAP can be used with the Iterative Deepening A* (IDA*) algorithm to overcome time complexity problems if the system has only one definite goal (Millington 2019). IDA* is used for the planning phase of GOAP to find the most suitable action set to reach to the overall goal as efficiently as possible.

Determining the goals and the actions properly at first in GOAP is one of the most important aspects to consider. In this respect, the modularity aspect of GOAP has several shortcomings. Looking at earlier examples, another problem faced by the researchers and AI developers is the system's performance or time complexity.

4.5 Artificial neural networks

Artificial neural networks (ANNs) are computational systems that contain many processors with multiple connections between them. They are created with the influence of the neural system of humans, with which they have certain vague similarities (Gupta 2013). Because ANNs can process much information, they are mainly used for the data operations in engineering, such as pattern recognition, image processing and alike.

In an ANN, the *artificial neurons* (or simply *nodes*) contain information and *edges* connect these nodes by sending data signals. All the input nodes have weights that are calculated with a mathematical function to decide which neuron is *fired*. The output of a neuron is calculated using its inputs. There can be different nodes that contain information at different layers. These different layers can apply different modifications to their input signals. The signals start from the input layer and end at the output layer, while also crossing paths with themselves and hidden layers in between them multiple times.

ANNs are mostly used as a learning component in machine learning methods. However, in some cases, they can also be used as the main decision-making method of NPCs in a computer game. Therefore, in this section, we will only cover such instances.

Madsen and Adamatti (2013) used ANNs for the decision-making process of the NPCs in a role-playing game. They used ANNs and FSMs to deciding for an attacking behavior of the NPC. With using training of the ANN, the guard NPC can decide to attack or escape the player character. Even though they did not use a computer game in their research, Jolly et al. (2007) proposed a novel decision-making method with ANNs for the NPCs in a robot football league. This method has two stages; weight matrices of the ANNs are calculated in the first stage and in the second, decisions are made according to these estimates. Robot NPCs decide to take an action with respect to the zone in which they are located in the field (i.e. defense, pass and attack zones). Their approach is effective since their simulations give superior results when it is used on 3-on-3 robot football matches.

5 Hybrid methods for decision-making

Some decision-making techniques can be combined to provide more comprehensive methods. We list these studies to give an overview of the meaningful combinations of techniques in this section.

For instance, FL and ANNs can be combined as a new decision-making method. Based on a combination of these two concepts, Shu and Chaudhari (2008) proposed a behavioral decision-making method. Their algorithm has two parts: offline behavior rule extraction

and online behavior decision. Using this technique, they aim to avoid sequenced or ordered actions of NPCs.

El-Nasr et al. (2000) created a new emotion model called FLAME, which uses FL to map emotions to events. They also combined FL with machine learning (ML) techniques concerning aspects such as event expectations, rewards, user action patterns, and object-emotion relations. They state that their proposed model improves the believability of the AI agent's interactions. According to them, FLAME can be extended with different concepts such as personality, social behavior and so on.

BTs are one of the essential decision-making methods for the NPCs as we have covered in Sect. 4.2.2. They are also combined with other methods, such as FL or ML. With respect to that, researchers combined BTs with FL to present a decision-making method for the robot agents that play a football game simulation in Abiyev et al. (2016). They used BTs with fuzzy inputs for general decisions such as moving, passing or shooting the ball, and the FL for obstacle avoidance part of the agents. Testing their algorithm on simulations, they found that it gives satisfying results in both the decision-making and obstacle avoidance aspects.

BTs can also be combined with genetic algorithms (GAs) or ML techniques, such as reinforcement learning (RL). In the particular example from Schwab and Hlavacs (2015), they are combined with many methods, including GOAP and GAs. The researchers propose a new method that uses the given problem domain as a source for their BTs, extracted from GOAP model. Their procedure first analyzes GOAP's behavior in a Monte–Carlo simulation, and then employs a GA to create a suitable BT. Their evaluation showed that the method may be used to infer BTs of action sequences that are reasonably similar to those seen in GOAP.

Kopel and Hajas (2018) employed DTs combined with more than one method; the NPCs in their environment uses ANNs and GAs to find optimized outcomes for their actions generated by the traditional DTs. They also used RL (i.e. specifically, the Q-learning approach) to improve the decision-making of their NPC AI. They concluded that the best results could be produced by a hybrid approach combining multilayer perceptron neural network with Q-learning rule and GA.

Dey and Child (2013) used BTs with RL. The RL technique used in their study was Q-learning. The method they created, called QL-BT, facilitated AI practitioners' use of BTs by allowing them decide when each branch of the logic should be executed. They reported that it performs better than the original BT structure in many areas.

Fu et al. (2016) also used RL with BTs. They compared their proposed model, QBot, in an AI test-bed platform called *Raven* and found that their method outperformed three standard AI agents on that platform. Sagredo-Olivenza et al. (2017) created a new BT model called Trained BT (TBT). In the model, TBTs are generated from the information obtained in a gameplay session through using the programming by demonstration (PbD) approach. PbD, as a branch of ML, investigates how to carry out a task by observing how an expert performs the task that the system needs to learn. According to the results, their model is easier to use since it gives game designers more authority on the NPC behavior than the traditional models.

Meng and Hyung (2022) combined BTs with the RL technique, Q-learning. They also used simulated annealing for the optimization of their method. The Q-learning algorithm is used to create the BT, which can evaluate the environment, choose appropriate actions, and then modify the action plan in the opposite direction to produce feedback that resembles

Table 3 The hybrid decision-making methods

Related study	Year	Method combination
El-Nasr et al. (2000)	2000	Decision tree, fuzzy logic
Orkin (2006)	2006	Finite state machine, goal-oriented action planning
Ohsone and Onisawa (2008)	2008	Decision tree, fuzzy logic
Shu and Chaudhari (2008)	2008	Artificial neural network, fuzzy logic
Florez-Puga et al. (2008)	2008	Behavior tree, case based reasoning
Fujii et al. (2008)	2008	Rule-based system, fuzzy logic
Perez et al. (2011)	2011	Behavior tree, grammatical evolution
Dey and Child (2013)	2013	Behavior tree, reinforcement learning
Madsen and Adamatti (2013)	2013	Finite state machine, artificial neural network
Schwab and Hlavacs (2015)	2015	Behavior tree, genetic algorithm, goal-oriented action planning
Fu et al. (2016)	2016	Behavior tree, reinforcement learning
Abiyev et al. (2016)	2016	Behavior tree, fuzzy logic
Sagredo-Olivenza et al. (2017)	2017	Behavior tree, programming by demonstration
Kopel and Hajas (2018)	2018	Decision tree, artificial neural network, genetic algorithm, reinforcement learning
Partlan et al. (2022)	2022	Behavior tree, genetic algorithm
Meng and Hyung (2022)	2022	Behavior tree, reinforcement learning, simulated annealing
Zhu and Feng (2022)	2022	Artificial neural network, genetic algorithm
Widhiyasana et al. (2022)	2022	Artificial neural network, genetic algorithm

that of a human. Their study improved the Q-learning algorithm via the concept of simulated annealing, and as a result, it suggests a suitable NPC BT.

Zhu and Feng (2022) combined GAs with ANNs to provide a new decision-making method of game NPCs. Their technique uses a GA to optimize the weights of a fixed ANN structure, enables the ANN to evolve unaided, and enhances the fitness function, selection, and crossover mutation techniques in a conventional GA to make them more suitable for an NPC AI. Their tests demonstrates that their system can enable intelligent behavior for decision-making process of NPCs.

Widhiyasana et al. (2022) also used a GA for ANNs, with the aim of determining the appropriate GA crossover and mutation rates for ANN weight adjustments. In their findings, the optimal crossover rate for the GA as an ANN weight adjustment, is determined as 0.6, while the optimal specific mutation rate is 0.09. The average win value is highest for the given crossover rate, and this value tends to improve with each generation. According to their results, the given mutation rate also has the greatest average win value.

Similar to previous studies, Partlan et al. (2022) developed a new tool, called EvolvingBehavior, to help the game developers to create a better NPC decision-making method. As such Rodrigues et al. (2021) have used, they also used the BT structure of Unreal Engine and enhanced it with genetic programming. When they have evaluated their tool in a 3D survival game, they discovered that EvolvingBehavior could generate NPC behaviors that are similar to the game designer's objectives.

Other than these studies, there are studies such as Perez et al. (2011) and Florez-Puga et al. (2008), in which the former uses the grammatical evolution with BTs, and the latter, a combined approach of case based reasoning and BTs. The studies that used hybrid decision-making methods are shown in Table 3.

6 Frameworks for decision-making

In the field of NPC decision-making, several frameworks have been established for various goals. Some are used for general decision-making of NPCs, some, as personality engines for virtual characters and others, as emotion engines for the appraisal theory.

We have already discussed some of the frameworks that were included in the research covered in previous sections. However, some of the frameworks that we mention here were not included earlier. The primary subjects of the studies that produced these previously unmentioned frameworks may be seen as being beyond our scope, insufficient for our purpose, or in some cases, irrelevant. We included them here, nonetheless, since these frameworks on their own are used for the decision-making of NPCs.

General decision-making of NPCs play an important part in games. *AI Graph Editor* (Miyake et al. 2019) is one the frameworks used for this purpose. The authors created this framework and used it in the game Final Fantasy XV, taking advantage of the BTs and the FSMs together. They designed a decision graph tool for finding the run-time problems easily. Their tool works both offline and online, and employs a layered model of BTs and FSMs.

As we described in Sect. 4.4 in more detail, Orkin (2006) and his colleagues created another decision-making framework for the game F.E.A.R as *F.E.A.R SDK*.[2] They also created other frameworks as a research group such as *GPGOAP*[3] and *ReGoap*.[4] The underlying method for all these frameworks is GOAP. The design of the *F.E.A.R SDK*, has already been described in detail, and we will not elaborate it further in this section.

There are also personality engines that allow individuals to design many of the psychological attributes of an NPC themselves. One example is Extreme AI[5] personality engine developed by Georgeson and Child (2016). It can be used to create adaptive NPC personalities that takes advantage of all thirty facets of the big five personality model (for details, see Rothmann and Coetzer (2003)).

Another area in which the frameworks are implemented for is cognitive appraisal (Folkman and Lazarus 1984). The researchers created GAMYGDALA (Popescu et al. 2013) to enable game developers to add emotions to the NPCs according to the context by using inputs based on goals and related events, according to the OCC model of emotions (see details in Steunebrink et al. (2009)).

Dias et al. (2014) developed FAtiMA framework, which is also an emotional appraisal engine. They implement and compare different appraisal theories in certain scenarios using their core algorithm and a set of components composed of different functionalities for appraisals or behaviors. They also described the usage of FAtiMA toolkit in more detail with a more recent study (Mascarenhas et al. 2021).

Comme il Faut (CiF) (McCoy et al. 2010, 2011) is one of the AI frameworks that deals with the social interaction between NPCs. Its main aim is to provide a playable model of social interaction with reusable representations of social norms and interactions. To simulate the interactions for the framework, they provide the necessary structures, such as social knowledge and the information on how this knowledge is used. The framework models the NPCs with a set of traits, feelings and relationships that can form intents, actions,

[2] https://github.com/xfw5/Fear-SDK-1.08.
[3] https://github.com/stolk/GPGOAP.
[4] https://github.com/luxkun/ReGoap.
[5] Unity Asset Store, https://bit.ly/3ezXtKj.

Table 4 The NPC decision-making frameworks

Framework type	Framework
General decision-making	AI graph editor (Miyake et al. 2019), F.E.A.R SDK, GPGOAP, ReGoap (Orkin 2006)
Personality engine	Extreme AI (Georgeson and Child 2016)
Emotion appraisal engine	GAMYGDALA (Popescu et al. 2013), FAtiMA (Dias et al. 2014; Mascarenhas et al. 2021), EmoBeT (Belle et al. 2022), EmotionalBT (Dell'Acqua and Costantini 2022)
Social interaction	CiF (McCoy et al. 2010, 2011), CiF-CK (Guimaraes et al. 2017)
Communication network	GVSN (Perrie and Li 2014)

remembering past events and cultural relations. They created a narrative game called Prom Week (McCoy et al. 2014) to demonstrate these abilities and it successfully employed complex and unique behaviors for the NPCs. Furthermore, Guimaraes et al. (2017) developed CiF Creation Kit (CiF-CK) to extend CiF framework by adapting it to a first person RPG game to use it for the interactions of the NPCs and successfully released this new application as a modification to this game.

Belle et al. (2022) created EmoBeT framework to provide NPCs that are more realistic. They found that, their approach makes it possible to design NPCs with psychologically motivated actions and personalities that influence their decisions. Their outcomes demonstrated that the player's choices had an impact on the NPC's feelings. Following that, these feelings affected the mood, which was crucial for making decisions. The results also demonstrated a connection between the NPC's default mood and its processing of specific emotions. This connection also showed how NPC decision-making was impacted by default mood.

Dell'Acqua and Costantini (2022) advanced the previous research from Johansson and Dell'Acqua (2012), proposing an EmotionalBT framework. In their view, AI agents should be equipped with an emotion recognition and management module, capable of empathy, in order to build and improve effective human–AI teams. To that end, they created a framework based on a strengthened understanding of BTs. They used a key example and a larger case study to demonstrate the efficacy of their suggested architecture.

Perrie and Li (2014) developed a gossip virtual social network for NPCs in games. In their framework, the NPCs can communicate with each other and interchange the knowledge acquired according to certain social-psychological rules. They demonstrated the success of their framework with both statistical analysis and a survey of real users.

We have listed all the NPC decision-making frameworks we have surveyed in Table 4.

7 Discussion and limitations

Overall, 106 studies were covered in this survey. The first 19 are either in the introduction section to give necessary background explanations about our aim, are in the methodology or the related surveys section. In contrast to previous related surveys, we set out to explain the basics and working principles of the decision-making methods in detail before presenting the studies in each category.

Table 5 The decision-making methods used in known games for NPCs

Game	Studio/year	Method	Source
Virtua fighter	Sega-AM2, 1993	RB	Yue and de Byl (2006)
Close Combat	Atomic games, 1996	FL	Sweetser and Wiles (2002)
Creatures	Millennium interactive, 1996	ANN	Yannakakis and Togelius (2018)
Half-life	Valve, 1998	FSM	Source codes[a]
Baldur's gate	BioWare, 1998	RB	Yue and de Byl (2006)
Unreal	Epic games, 1998	FL	Johnson and Wiles (2001)
Police quest: SWAT 2	Sierra entertainment, 1998	FL	Sweetser and Wiles (2002) and Johnson and Wiles (2001)
Civilization: call to power	activision, 1999	FL	Johnson and Wiles (2001)
The Sims	Maxis, 2000	RB, GOAP	Millington (2019)
Black & White	Lionhead Studios, 2000	DT	Yannakakis and Togelius (2018)
No one lives forever 2: a spy in H.A.R.M.'s way	Monolith Productions, 2002	FSM, GOAP	Orkin (2003)
Halo 2	Bungie Inc., 2004	BT	Yannakakis and Togelius (2018)
S.W.A.T. 4	Irrational games, 2005	GOAP	Pittman (2007)
F.E.A.R	Sierra Entertainment, 2005	GOAP	Orkin (2006)
S.T.A.L.K.E.R.: shadow of Chernobyl	GSC Game World, 2007	GOAP	Long (2007)
Deus Ex: human revolution	Eidos, 2011	GOAP	Sloan (2015)
The last of us	naughty dog, 2013	DT, BT	Panwar (2022)
Alien: isolation	SEGA, 2014	BT	Panwar (2022)
Final fantasy XV	Square Enix, 2016	FSM, BT	Miyake et al. (2019)
Mafia III	2K games, 2016	DT, BT	Holba and Huber (2021)

[a] https://github.com/ValveSoftware/halflife

Table 6 Full taxonomy of the surveyed studies

DM methods	Applied			Used for				Game type/genre				
	On NPC	On Env.[a]	On other	Planning	Movement	Vehicle control	Interact	2D/platform	FPS	Strategy	RPG	Action/adven.[a]
FSM	Fu and Houlette (2002), Laird (2001), Syahputra et al. (2019), Fathoni et al. (2020), Fauzi et al. (2019), Sindhu et al. (2022) and Orkin (2006)	Laird (2001)	Miyake et al. (2019)	Fu and Houlette (2002), Laird (2001), Syahputra et al. (2019), Fathoni et al. (2020), Fauzi et al. (2019), Sindhu et al. (2022), Miyake et al. (2019) and Orkin (2006)	Fu and Houlette (2002), Laird (2001), Syahputra et al. (2019), Fathoni et al. (2020), Sindhu et al. (2022) and Orkin (2006)		Fathoni et al. (2020)	Fu and Houlette (2002) and Sindhu et al. (2022)	Laird (2001), Syahputra et al. (2019) and Fathoni et al. (2020)	Fauzi et al. (2019)		
DT	Mas'udi et al. (2021), Wexler (2002), Ohsone and Onisawa (2008), El-Nasr et al. (2000), Kopel and Hajas (2018), Quadir and Khder (2022) and Lie and Istiono (2022)			Mas'udi et al. (2021), Wexler (2002), Kopel and Hajas (2018), Quadir and Khder (2022) and Lie and Istiono (2022)	Mas'udi et al. (2021), Wexler (2002), Kopel and Hajas (2018) and Quadir and Khder (2022)	Mas'udi et al. (2021)	Ohsone and Onisawa (2008) and El-Nasr et al. (2000)	Quadir and Khder (2022) and Lie and Istiono (2022)	Kopel and Hajas (2018)	Wexler (2002)		Mas'udi et al. (2021), Ohsone and Onisawa (2008) and El-Nasr et al. (2000)

Table 6 (continued)

DM methods	Applied			Used for				Game type/genre				
	On NPC	On Env.[a]	On other	Planning	Movement	Vehicle control	Interact	2D/platform	FPS	Strategy	RPG	Action/adven.[a]
BT	Isla (2005), Simonov et al. (2019), Colledanchise and Ögren (2018), Lin et al. (2019), Rodrigues et al. (2021), Abiyev et al. (2016), Schwab and Hlavacs (2015), Dey and Child (2013), Fu et al. (2016), Sagredo-Olivenza et al. (2017), Partlan et al. (2022), Perez et al. (2011), Holba and Huber (2021) and Panwar (2022)	Johansson and Dell'Acqua (2012), Pyjas et al. (2022) and Holba and Huber (2021)	Miyake et al. (2019) and Sagredo-Olivenza et al. (2017)	Isla (2005), Simonov et al. (2019), Lin et al. (2019), Miyake et al. (2019), Abiyev et al. (2016), Schwab and Hlavacs (2015), Dey and Child (2013), Fu et al. (2016), Sagredo-Olivenza et al. (2017), Partlan et al. (2022), Pyjas et al. (2022), Holba and Huber (2021) and Panwar (2022)	Isla (2005), Colledanchise and Ögren (2018), Lin et al. (2019), Rodrigues et al. (2021), Miyake et al. (2019), Abiyev et al. (2016), Schwab and Hlavacs (2015), Dey and Child (2013), Fu et al. (2016), Partlan et al. (2022), Perez et al. (2011), Pyjas et al. (2022), Holba and Huber (2021) and Panwar (2022)	Holba and Huber (2021)	Johansson and Dell'Acqua (2012)	Colledanchise and Ögren (2018), Fu et al. (2016) and Perez et al. (2011)	Isla (2005), Schwab and Hlavacs (2015), Partlan et al. (2022), Pyjas et al. (2022) and Holba and Huber (2021)	Sagredo-Olivenza et al. (2017)	Lin et al. (2019), Rodrigues et al. (2021), Miyake et al. (2019) and Panwar (2022)	Johansson and Dell'Acqua (2012), Simonov et al. (2019), Abiyev et al. (2016) and Dey and Child (2013)
RB	Fujii et al. (2008)				Fujii et al. (2008)							Fujii et al. (2008)

Table 6 (continued)

DM methods	Applied			Used for				Game type/genre				
	On NPC	On Env.[a]	On other	Planning	Movement	Vehicle control	Interact	2D/platform	FPS	Strategy	RPG	Action/adven.[a]
FL	Fujii et al. (2008), Johnson and Wiles (2001), Shaout et al. (2006), Van Waveren (2001), Li et al. (2004), Soylucicek et al. (2017), Niewiadomski and Renkas (2014), Ohsone and Onisawa (2008), Abiyev et al. (2016), Shu and Chaudhari (2008) and El-Nasr et al. (2000)			Johnson and Wiles (2001), Shaout et al. (2006), Van Waveren (2001), Li et al. (2004) and Abiyev et al. (2016)	Fujii et al. (2008), Johnson and Wiles (2001), Shaout et al. (2006), Van Waveren (2001), Li et al. (2004), Soylucicek et al. (2017), Abiyev et al. (2016) and Shu and Chaudhari (2008)	Niewiadomski and Renkas (2014)	Ohsone and Onisawa (2008) and El-Nasr et al. (2000)	Shaout et al. (2006), Li et al. (2004) and Soylucicek et al. (2017)	Johnson and Wiles (2001) and Van Waveren (2001)	Johnson and Wiles (2001) and Shu and Chaudhari (2008)		Fujii et al. (2008), Niewiadomski and Renkas (2014), Ohsone and Onisawa (2008), Abiyev et al. (2016) and El-Nasr et al. (2000)
GOAP	Orkin (2003), Orkin (2006), Studiawan et al. (2018), Suyikno and Setiawan (2019), Schwab and Hlavacs (2015) and Johansen et al. (2022)			Orkin (2003), Orkin (2006), Studiawan et al. (2018), Suyikno and Setiawan (2019) and Schwab and Hlavacs (2015)	Orkin (2003), Orkin (2006), Studiawan et al. (2018), Suyikno and Setiawan (2019), Schwab and Hlavacs (2015) and Johansen et al. (2022)		Johansen et al. (2022)	Studiawan et al. (2018)	Orkin (2003), Orkin (2006) and Schwab and Hlavacs (2015)			Johansen et al. (2022)

Table 6 (continued)

DM methods	Applied			Used for				Game type/genre				
	On NPC	On Env.[a]	On other	Planning	Movement	Vehicle control	Interact	2D/platform	FPS	Strategy	RPG	Action/adven.[a]
ANN	Suyikno and Setiawan (2019) Madsen and Adamatti (2013), Jolly et al. (2007), Shu and Chaudhari (2008), Zhu and Feng (2022), Kopel and Hajas (2018) and Widhiyasana et al. (2022)	Zhu and Feng (2022)		Madsen and Adamatti (2013), Jolly et al. (2007), Zhu and Feng (2022), Kopel and Hajas (2018) and Widhiyasana et al. (2022)	Madsen and Adamatti (2013), Jolly et al. (2007), Shu and Chaudhari (2008), Zhu and Feng (2022), Kopel and Hajas (2018) and Widhiyasana et al. (2022)			Zhu and Feng (2022)	Kopel and Hajas (2018)	Shu and Chaudhari (2008) and Widhi-yasana et al. (2022)	Madsen and Adamatti (2013)	Jolly et al. (2007) and Zhu and Feng (2022)

[a] "Env." is an acronym for "Environment", "Adven." is an acronym for "Adventure"

Using the reviewed studies, we aimed to gather published video games and the NPC decision-making methods employed in those games. For every game listed, we give a source material that details the decision-making methods used in each. The list for the NPC decision-making methods used in these video games is shown in Table 5.

We categorized the methods with respect to three important issues: whom the methods applied for, which area of decision-making they are used in, and which game genre that they belong to. Using this taxonomy, we summarized these findings in Table 6. We put the reference of the article into the correct place of the table if the mentioned studies used given decision-making methods in their experiments or simulations, or if they analyzed previously published games with respect to the usage of the methods.

According to our findings, decision-making methods can be applied to NPCs, to environment that NPCs interact or to other entities such as communities of NPCs. Decision-making methods of NPCs can be useful for planning or acting, spatial movement, vehicle/device control or interaction with other game elements such as players or other NPCs. The five main game genres to which the game environments in the studies belong are as follows: 2D/platform games, first-person shooter (FPS) games, strategy games, role-playing games (RPGs) or action/adventure games.

We gathered a comprehensive list of studies for our survey. Nonetheless, there remain some omissions and certain limitations in our study to be completed in future works.

Firstly, we only collected the studies that used traditional AI methods for decision-making, as discussed in Sect. 1. In recent years, machine learning (ML) technologies have also been implemented in this area. However, we decided to exclude those studies since most applied these ML techniques on the main player of game for training, but not on the NPCs. In future, our study can be extended with the inclusion of the applications of ML techniques on NPCs.

Secondly, the studies surveyed provided no information about how or why they selected one decision-making method over another for NPCs. In addition, there are no viable comparison metrics to investigate the advantages and disadvantages of each NPC decision-making methods. Thus, our survey can be extended with this additional information if the researchers create usable comparison metrics between the methods, and the game companies provide access to their data about these important decision criteria.

Our surveyed studies generally analyzed the decision-making methods of NPCs with respect to the player character, but the majority of them failed to consider the NPC-to-NPC interaction. In future studies, this aspect should also be investigated, due to the need for more dynamic decision-making methods when one NPC tries to anticipate the behavior of another.

Table 5 can be extended with more recent video games. In this context, there can be a more thorough investigation of video games, not only through academic articles, books and proceedings, but also through online materials, such as video articles from known publishers, and blog posts of game companies or acclaimed game magazines.

We categorized the decision-making methods as in Fig. 1 in line with our findings, but additions or changes could be made to these categories. For example, Millington (2019) proposed a different categorization in his book, but for some of the methods, there appear to be no academic studies. Therefore, we decide to create our own categorization. Our survey can be expanded by including studies we may have inadvertently overlook.

As one of the last limitations for our study, we can reassess the search keywords, and more accurately identify relevant studies. We aimed to access the correct information for every study included with the resources currently available. However, this task required us to analyze many sources. It will be possible to increase the number of keywords or make a more refined keyword analysis in future as new methods emerge.

There are four types of validity threats that need to be addressed concerning our survey.

- As a construct validity threat, we may have missed some important studies in our results due to the nature of our queries.
- As internal validity threats, since we survey a highly diverse topic and aim to include some studies only to explain the topics further, we may have shown bias towards particular concepts. Another issue is the adequacy of the academic databases and the academic search platforms used in our survey.
- One of our external validity threats is the possibility that some of our surveyed studies may have not included all the necessary information about the decision-making methods or the games in question.
- As our conclusion validity issue, while both authors collaborated on collection, review and analysis of the dataset, and writing of the resulting article, there may be alternative classifications for some studies surveyed. Some studies in one section may belong to a different section or sections, or they can belong to different categories for our resulting taxonomy table.

Even though utmost care has been taken to provide the best possible outcome, we believe that it is vital to list these validity threats as additional limitations to our study.

8 Conclusion

In this study, we examined multiple papers about decision-making of NPCs in computer games and categorized these according to different usage perspectives. At the same time, we also covered the developed frameworks and environments. Hence, our survey is unique and comprehensive in the sense that it specifically aims to analyze the decision-making mechanisms of the NPCs, and it brings different methods in more than one subarea together in the same study.

We are aware that there are studies that our survey was unable to include. Some were beyond of our scope, or others were the sole instances of their respective category. For the latter, we decided to omit these categories. In the future, newer examples in this area can cover more categories than at present, and more detailed studies can emerge.

Previous surveys in the area did not look the studies we surveyed from the same perspective as ours; they mostly used the general AI standpoint. In contrast, we attempted to detect which mechanisms the AI agents employed for decision-making, how these agents affected the game experience by employing them, and the diversity that the decision-making methods have shown over time. This was a necessity from our point of view due to the lack of surveys taking a similar approach. We believe that we contributed to an area where the previous work is limited and distributed over diverse studies. Our hope is that this study will lay down a foundation for researchers aiming to work further on the decision-making methods of non-player characters in computer games.

9 Supplementary information

The keywords we used to research previous studies for this article can be found in supplementary materials as a ".xlsx" file with the name of "MCU-KO-2022-Keywords.xlsx". The dataset we collected for our article can be found in Supplementary Materials as an ".xlsx" file with the name of "MCU-KO-2022-Dataset.xlsx".

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10462-023-10491-7.

Acknowledgements The authors thank the anonymous reviewers for their valuable comments and feedback which have improved the manuscript significantly. They also thank Simon Edward Mumford for his help in language editing and proofreading.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Abiyev RH, Günsel I, Akkaya N et al (2016) Robot soccer control using behaviour trees and fuzzy logic. Procedia Comput Sci 102(August):477–484. https://doi.org/10.1016/j.procs.2016.09.430

Belle S, Gittens C, Graham TN (2022) A framework for creating non-player characters that make psychologically-driven decisions. In: 2022 IEEE international conference on consumer electronics (ICCE), IEEE. pp 1–7. https://doi.org/10.1109/ICCE53296.2022.9730383

Bourg DM, Seemann G (2004) AI for game developers. O'Reilly Media Inc, Sebastopol

Cavazza M (2000) Al in computer games: survey and perspectives. Virtual Real 5(4):223–235. https://doi.org/10.1007/BF01408521

Charles D, McGlinchey S (2004) The past, present and future of artificial neural networks in digital games. In: Proceedings of the 5th international conference on computer games: artificial intelligence, design and education. The University of Wolverhampton, pp 163–169

Colledanchise M, Ögren P (2018) Behavior trees in robotics and AI: an introduction. CRC Press, Cambridge

Colledanchise M, Marzinotto A, Ögren P (2014) Performance analysis of stochastic behavior trees. In: 2014 IEEE international conference on robotics and automation (ICRA), IEEE. pp 3265–3272. https://doi.org/10.1109/ICRA.2014.6907328

da Silva GA, de Souza Ribeiro MW (2021) Development of non-player character with Believable Behavior: a systematic literature review. In: Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital. SBC, Porto Alegre, pp 319–323. https://doi.org/10.5753/sbgames_estendido.2021.19660

Danielson M, Ekenberg L (2011) Decision making in intelligent agents. In: Sierra AP, Rabunal JR, Dorado J (eds) Encyclopedia of artificial intelligence. IGI Global, Pennsylvania, pp 431–436

Dell'Acqua P, Costantini S (2022) Emotional behavior trees for empathetic human-automation interaction. In: WOA 2022: 23rd workshop from objects to agents

Dey R, Child C (2013) QL-BT: Enhancing behaviour tree design and implementation with Q-learning. In: 2013 IEEE conference on computational inteligence in games (CIG), IEEE. pp 1–8. https://doi.org/10.1109/CIG.2013.6633623

Dias J, Mascarenhas S, Paiva A (2014) FAtiMA modular: towards an agent architecture with a generic appraisal framework. In: Bosse T, Broekens J, Dias J et al (eds) Emotion modeling: towards pragmatic computational models of affective processes. Springer, Cham, pp 44–56

El-Nasr MS, Yen J, Ioerger TR (2000) FLAME—fuzzy logic adaptive model of emotions. Auton Agent Multi-Agent Syst 3(3):219–257. https://doi.org/10.1023/A:1010030809960

Fathoni K, Hakkun R, Nurhadi H (2020) Finite state machines for building believable non-playable character in the game of Khalid ibn Al-Walid. J Phys Conf Ser 1577:012018. https://doi.org/10.1088/1742-6596/1577/1/012018

Fauzi R, Hariadi M, Nugroho SMS et al (2019) Defense behavior of real time strategy games: comparison between hfsm and fsm. Indones J Electr Eng Comput Sci 13(2):634–642. https://doi.org/10.11591/ijeecs.v13.i2.pp634-642

Florez-Puga G, Gomez-Martin M, Diaz-Agudo B et al (2008) Dynamic expansion of behaviour trees. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment. pp 36–41. https://doi.org/10.1609/aiide.v4i1.18669

Folkman S, Lazarus RS (1984) Stress, appraisal, and coping. Springer, New York

Forgy CL (1979) On the efficient implementation of production systems. PhD thesis, Carnegie Mellon University

Forgy CL (1989) Rete: a fast algorithm for the many pattern/many object pattern match problem. In: Mylopolous J, Brodie M (eds) Readings in artificial intelligence and databases. Morgan Kaufmann, San Francisco, pp 547–559

Fu D, Houlette R (2002) Putting AI in entertainment: an AI authoring tool for simulation and games. IEEE Intell Syst 17(4):81–84. https://doi.org/10.1109/MIS.2002.1024756

Fu Y, Qin L, Yin Q (2016) A reinforcement learning behavior tree framework for game AI. In: 2016 international conference on economics, social science, arts, education and management engineering. Atlantis Press, pp 573–579

Fujii S, Nakashima T, Ishibuchi H (2008) A study on constructing fuzzy systems for high-level decision making in a car racing game. In: 2008 IEEE congress on evolutionary computation, CEC 2008, IEEE. pp 3626–3633

Georgeson J, Child C (2016) NPCs as people, too: the extreme AI personality engine. https://doi.org/10.48550/arXiv.1609.04879

Guimaraes M, Santos P, Jhala A (2017) CiF-CK: an architecture for social NPCs in commercial games. In: 2017 IEEE conference on computational intelligence and games (CIG), IEEE. pp 126–133. https://doi.org/10.1109/CIG.2017.8080425

Gupta N (2013) Artificial neural network. Netw Complex Syst 3(1):24–28

Harel D (1987) Statecharts: a visual formalism for complex systems. Sci Comput Prog 8(3):231–274. https://doi.org/10.1016/0167-6423(87)90035-9

Harper PR (2005) A review and comparison of classification algorithms for medical decision making. Health Policy 71(3):315–331. https://doi.org/10.1016/j.healthpol.2004.05.002

Holba J, Huber G (2021) Open-world enemy AI in Mafia III. In: Game AI pro—online edition 2021. Game AI pro, chap 16

Iovino M, Scukins E, Styrud J et al (2022) A survey of behavior trees in robotics and AI. Robot Auton Syst 154(104):096. https://doi.org/10.1016/j.robot.2022.104096

Isla D (2005) Gdc 2005 proceeding: handling complexity in the halo 2 AI. In: Game developers conference 2005. https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai

Johansen NS, Kær LB, Stolberg JAB, et al. (2022) Towards believable non-player characters with domain-independent planning. In: 2022 workshop on scheduling and planning applications workshop

Johansson A, Dell'Acqua P (2012) Comparing behavior trees and emotional behavior networks for npcs. In: 2012 17th international conference on computer games (CGAMES), IEEE. pp 253–260. https://doi.org/10.1109/CGames.2012.6314584

Johnson D, Wiles J (2001) Computer games with intelligence. In: 10th IEEE international conference on fuzzy systems. (Cat. No. 01CH37297), IEEE. pp 1355–1358. https://doi.org/10.1109/FUZZ.2001.1008909

Jolly K, Ravindran K, Vijayakumar R et al (2007) Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks. Robot Auton Syst 55(7):589–596. https://doi.org/10.1016/j.robot.2006.12.011

Justesen N, Bontrager P, Togelius J et al (2020) Deep learning for video game playing. IEEE Trans Games 12(1):1–20. https://doi.org/10.1109/TG.2019.2896986

Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4:237–285. https://doi.org/10.1613/jair.301

Kitchenham B (2004) Procedures for performing systematic reviews, vol 33. Keele University, Keele, pp 1–26

Kopel M, Hajas T (2018) Implementing AI for non-player characters in 3D video games. In: Nguyen NT, Hoang DH, Hong TP et al (eds) Intelligent information and database systems. Springer, Cham, pp 610–619

Laird JE (2001) It knows what you're going to do: adding anticipation to a quakebot. In: Proceedings of the fifth international conference on autonomous agents. pp 385–392. https://doi.org/10.1145/375735.376343

Lara-Cabrera R, Nogueira-Collazo M, Cotta C et al (2015) Game artificial intelligence: challenges for the scientific community. In: CoSECivi 2015. https://doi.org/10630/9942
Li Y, Musilek P, Wyard-Scott L (2004) Fuzzy logic in agent-based game design. In: IEEE annual meeting of the fuzzy information, 2004. Processing NAFIPS'04, IEEE. pp 734–739. https://doi.org/10.1109/NAFIPS.2004.1337393
Lie CSK, Istiono W (2022) How to make npc learn the strategy in fighting games using adaptive AI? Int J Sci Tech Res Eng 7:4
Lin J, He J, Zhang N (2019) Application of behavior tree in AI design of MOBA games. In: 2019 IEEE 2nd international conference on knowledge innovation and invention (ICKII), IEEE. pp 323–326. https://doi.org/10.1109/ICKII46306.2019.9042660
Long E (2007) Enhanced NPC behaviour using goal oriented action planning. Master's thesis, University of Abertay Dundee, Dundee
Loyall BA, Bates J (1991) Hap a reactive, adaptive architecture for agents. In: Citeseer
Madsen CABCW, Adamatti DF (2013) Using artificial neural networks in NPC decision-making process. Int J Comput Inf Technol 2(6):1009–1013
Mas'udi NA, Jonemaro EMA, Akbar MA et al (2021) Development of non-player character for 3d kart racing game using decision tree. Fount Inform J 6(2):51–60. https://doi.org/10.21111/fij.v6i2.4678
Mascarenhas S, Guimarães M, Santos PA et al (2021) FAtiMA Toolkit—toward an effective and accessible tool for the development of intelligent virtual agents and social robots. Preprint at http://arxiv.org/abs/2103.03020
Mateas M, Stern A (2002) A behavior language for story-based believable agents. IEEE Intell Syst 17(4):39–47. https://doi.org/10.1109/MIS.2002.1024751
McCoy J, Treanor M, Samuel B et al (2010) Comme il Faut 2: a fully realized model for socially-oriented gameplay. In: Proceedings of the intelligent narrative technologies III workshop. Association for Computing Machinery, New York, pp 1–8. https://doi.org/10.1145/1822309.1822319
McCoy J, Treanor M, Samuel B et al (2011) Comme il Faut: a system for authoring playable social models. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment
McCoy J, Treanor M, Samuel B et al (2014) Social story worlds with Comme il Faut. IEEE Trans Comput Intell AI Games 6(2):97–112. https://doi.org/10.1109/TCIAIG.2014.2304692
Meng F, Hyung CJ (2022) Research on multi-npc marine game ai system based on q-learning algorithm. In: 2022 IEEE international conference on artificial intelligence and computer applications (ICAICA), IEEE. pp 648–652. https://doi.org/10.1109/ICAICA54878.2022.9844648
Millington I (2019) Artificial intelligence for games, 3rd edn. CRC Press, Cambridge
Miyake Y, Shirakami Y, Shimokawa K et al (2019) A character decision-making system for FINAL FANTASY XV by combining behavior trees and state machines. In: Game AI pro 360: guide to architecture. CRC Press, p 339. https://doi.org/10.4324/9781315151700
Moher D, Liberati A, Tetzlaff J et al (2009) Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. Ann Intern Med 151(4):264–269
Nareyek A (2000) Intelligent agents for computer games. In: International conference on computers and games. Springer, pp 414–422. https://doi.org/10.1007/3-540-45579-5_28
Neufeld X, Mostaghim S, Sancho-Pradel DL et al (2019) Building a planner: a survey of planning systems used in commercial video games. IEEE Trans Games 11(2):91–108. https://doi.org/10.1109/TG.2017.2782846
Niewiadomski A, Renkas K (2014) Hierarchical fuzzy logic systems and controlling vehicles in computer games. J Appl Comput Sci 22(1):201–212
Novák V, Perfilieva I, Mockor J (1999) Mathematical principles of fuzzy logic, vol 517. Springer, New York
OBrien L (1996) Fuzzy logic in games. Game Dev Mag 3(2):52–55
Ochs M, Sabouret N, Corruble V (2009) Simulation of the dynamics of nonplayer characters' emotions and social relations in games. IEEE Trans Comput Intell AI Games 1(4):281–297. https://doi.org/10.1109/TCIAIG.2009.2036247
Ohsone K, Onisawa T (2008) Friendly partner system of poker game with facial expressions. In: 2008 IEEE symposium on computational intelligence and games, IEEE. pp 95–102. https://doi.org/10.1109/CIG.2008.5035626
Orkin J (2003) Applying goal-oriented action planning to games. In: Rabin S (ed) AI game programming wisdom, vol 2. Charles River Media Inc, New York
Orkin J (2006) Three states and a plan: the AI of FEAR. In: Game developers conference 2006. CMP Game Group, San Jose, p 4
Panwar H (2022) The NPC AI of the last of us: a case study. Preprint at https://doi.org/10.48550/ARXIV.2207.00682

Partlan N, Soto L, Howe J et al (2022) EvolvingBehavior: towards co-creative evolution of behavior trees for game NPCs. In: Proceedings of the 17th international conference on the foundations of digital games. Association for Computing Machinery, New York. https://doi.org/10.1145/3555858.3555896
Perez D, Nicolau M, O'Neill M et al (2011) Evolving behaviour trees for the mario AI competition using grammatical evolution. In: Kaufmann P, Castillo PA (eds) Applications of evolutionary computation. Springer, Berlin, pp 123–132
Perrie J, Li L (2014) Building a dynamic social community with non playable characters. IEICE Trans Inf Syst 97(8):1965–1973. https://doi.org/10.1587/transinf.E97.D.1965
Pirovano M (2012) The use of fuzzy logic for artificial intelligence in games. University of Milano, Milan
Pittman DL (2007) Practical development of goal-oriented action planning AI. Master's thesis, Southern Methodist University, Dallas
Popescu A, Broekens J, Van Someren M (2013) GAMYGDALA: an emotion engine for games. IEEE Trans Affect Comput 5(1):32–44. https://doi.org/10.1109/T-AFFC.2013.24
Pyjas GM, Weinel J, Broadhead M (2022) Storytelling and vr: inducing emotions through AI characters. In: Proc EVA London 2022. pp 198–204. https://doi.org/10.14236/ewic/EVA2022.37
Quadir AM, Khder MA (2022) Exploring the potential of AI-driven opponents in video games. In: 2022 ASU international conference in emerging technologies for sustainability and intelligent systems (ICETSIS), IEEE. pp 464–469. https://doi.org/10.1109/ICETSIS55481.2022.9888909
Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106. https://doi.org/10.1007/BF00116251
Receveur JB, Victor S, Melchior P (2020) Autonomous car decision making and trajectory tracking based on genetic algorithms and fractional potential fields. Intel Serv Robot 13(2):315–330. https://doi.org/10.1007/s11370-020-00314-x
Rodrigues S, Rayat HK, Kurichithanam RM et al (2021) Shriek: a role playing game using unreal engine 4 and behaviour trees. In: 2021 4th biennial international conference on nascent technologies in engineering (ICNTE), IEEE. pp 1–6. https://doi.org/10.1109/ICNTE51185.2021.9487723
Rothmann S (2003) The big five personality dimensions and job performance. SA J Ind Psychol 29(1):68–74. https://doi.org/10.10520/EJC88938
Russell S, Norvig P (2002) Artificial intelligence: a modern approach. Prentice Hall, Hoboken
Sagredo-Olivenza I, Gómez-Martín PP, Gómez-Martín MA et al (2017) Trained behavior trees: programming by demonstration to support AI game designers. IEEE Trans Games 11(1):5–14. https://doi.org/10.1109/TG.2017.2771831
Schwab P, Hlavacs H (2015) Capturing the essence: towards the automated generation of transparent behavior models. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment
Sekhavat YA (2017) Behavior trees for computer games. Int J Artif Intell Tools 26(02):1730001. https://doi.org/10.1142/S0218213017300010
Shaout A, King BW, Reisner LA (2006) Real-time game design of Pac-Man using fuzzy logic. Int Arab J Inf Technol 3(4):315–325
Shu F, Chaudhari NS (2008) A chaotic behavior decision algorithm based on self-generating neural network for computer games. In: 2008 3rd IEEE conference on industrial electronics and applications, IEEE. pp 1912–1915. https://doi.org/10.1109/ICIEA.2008.4582852
Sielicki M, Daszuta M, Szajerman D (2018) Adaptation and application of goal oriented action planning in unreal engine. Comput Game Innov 1:103
Simonov A, Zagarskikh A, Fedorov V (2019) Applying behavior characteristics to decision-making process to create believable game AI. Procedia Comput Sci 156:404–413. https://doi.org/10.1016/j.procs.2019.08.222
Sindhu RM, Annabel LSP, Monisha G (2022) Development of a 2d game using artificial intelligence in unity. In: 2022 6th international conference on trends in electronics and informatics (ICOEI), IEEE. pp 1031–1037. https://doi.org/10.1109/ICOEI53556.2022.9776750
Sloan C (2015) Drive-based utility-maximizing computer game non-player characters. PhD thesis, Technological University Dublin, Dublin. https://doi.org/10.21427/D7KK57
Sloan C, Mac Namee B, Kelleher JD (2011) Utility-directed goal-oriented action planning: a utility-based control system for computer game agents. University of Ulster, MartinMcGinnity Intelligent Systems Research Centre, Coleraine
Soylucicek AE, Bostanci E, Safak AB (2017) A fuzzy logic based attack strategy design for enemy drones in meteor escape game. Int J Comput Theory Eng 9(3):167–171. https://doi.org/10.7763/ijcte.2017.v9.1132
Steunebrink BR, Dastani M, Meyer JJC (2009) The occ model revisited. In: Proc 4th workshop on emotion and computing. Association for the Advancement of Artificial Intelligence

Studiawan R, Hariadi M (2018) Tactical planning in space game using goal-oriented action planning. J Adv Res Electr Eng 2:1. https://doi.org/10.12962/j25796216.v2.i1.32
Suyikno DA, Setiawan A (2019) Feasible NPC hiding behaviour using goal oriented action planning in case of hide-and-seek 3D game simulation. In: 2019 fourth international conference on informatics and computing (ICIC), IEEE. pp 1–6, https://doi.org/10.1109/ICIC47613.2019.8985962
Sweetser P, Wiles J (2002) Current AI in games: a review. Austral J Intell Inf Process Syst 8(1):24–42
Syahputra M, Arippa A, Rahmat R et al (2019) Historical theme game using finite state machine for actor behaviour. J Phys Conf Ser 1:012122. https://doi.org/10.1088/1742-6596/1235/1/012122
Van Waveren JMP (2001) The quake III arena bot. Master's thesis. Delft University of Technology, Delft
West N (1996) Next generation lexicon A to Z: a definitive guide to gaming terminology. NEXT generation. https://archive.org/details/nextgen-issue-015/page/n39/mode/2up. Accessed 21 Nov 2022
Wexler J (2002) Artificial intelligence in games. University of Rochester, Rochester
Widhiyasana Y, Harika M, Hakim FFN et al (2022) Genetic algorithm for artificial neural networks in real-time strategy games. Int J Inform Vis 6(2):298–305. https://doi.org/10.30630/joiv.6.2.832
Yanase J, Triantaphyllou E (2019) The seven key challenges for the future of computer-aided diagnosis in medicine. Int J Med Inform 129:413–422. https://doi.org/10.1016/j.ijmedinf.2019.06.017
Yannakakis GN, Togelius J (2018) Artificial intelligence and games. Springer, New York
Yue B, de Byl P (2006) The state of the art in game AI standardisation. In: Proceedings of the 2006 international conference on game research and development. Murdoch University, Murdoch, pp 41–46. https://doi.org/10.5555/1234341.1234350
Zarozinski M, Than L (2001) Imploding combinatorial explosion in a fuzzy system. In: AI game programming wisdom, Vol. 2. Charles River Media, Inc., pp 342–350
Zhu M, Feng L (2022) Design and implementation of NPC AI based on genetic algorithm and BP neural network. In: Proceedings of the 14th international conference on computer modeling and simulation. Association for Computing Machinery, New York, pp 168–173. https://doi.org/10.1145/3547578.3547604
Zijie W, Tongyu W, Hang G (2021) A survey: development and application of behavior trees. In: Liang Q, Wang W, Liu X et al (eds) Communications, signal processing, and systems. Springer, Singapore, pp 1581–1589