Adding Variety in NPCs Behaviour Using Emotions and Genetic Algorithms: the GENIE Project

Federica Agliata[†], Laura Anna Ripamonti[*], Dario Maggiorini[*], and Davide Gadia[*]

[†]Department of Computer Science
University of Milan
Email: {firstname.lastname}@studenti.unimi.it

[*]Department of Computer Science
University of Milan
Email: {firstname.lastname}@unimi.it

***Abstract*—In recent years we have been observing an increasing adoption of artificial intelligence in video games. With the availability of increasingly powerful hardware and advanced algorithms we can now scale up the quality of the AI available to every Non Playing Character (NPC). Thanks to this increased quality, NPCs can be made very believable and able to convey a compelling user experience. Despite this added quality and complexity, every NPC can get predictable with time and game designers are struggling to provide variety in games where many NPCs are present. Designing a specific, and unique, AI for every NPC can be a very time and resource consuming task. In this paper we propose GENIE (Genes Driven Decision Tree) as a tool to support game designers in the creation of a wide variety of behaviours. With GENIE, it is possible to define an NPC behaviour in term of reactions to its internal parameters. These parameters, in turn, are evaluated and generated using a genetic algorithm. NPCs can then evolve to different internal states and interact with the player using each time a different behaviour.**

***Index Terms*—Games, Artificial Intelligence, NPC Behaviour, Genetic Algorithms**

I. Introduction

As of today, we can observe an increasing adoption of artificial intelligence inside video games. In modern video games, artificial intelligence techniques are used to achieve various tasks. These tasks cover a wide range from pathfinding to procedural content generation, to machine learning. Among all the possible tasks, providing complex and believable behaviour to *Non Playing Characters* (NPCs) is strategic to convey a rich and compelling player experience. With the availability of increasingly powerful hardware, it is now possible to provide NPCs which are able to perform complex operations and take elaborate decisions while interacting with the player.

Despite the aforementioned potential to describe very articulated NPCs behaviour, every NPC will get predictable with time and game designers are usually struggling to provide variety in games where many NPCs are present. NPCs are getting predictable to players whenever the same behaviour and/or decisions pattern is proposed over a long time: a human counterpart is not going to find dealing with the artificial intelligence agent amusing or challenging on the long shot. Another problem is about providing variety when a population of NPCs is involved: as a matter of fact, having all the NPCs behaving in exactly the same way will provide a poor player experience and, once again, predictability will take over quickly. Designing a specific, and unique, behaviour for every NPC can be a very tedious activity for game designers and may also require a large amount of resources to store the descriptions of independent behaviours with small differences.

To support game designer in providing behavioural variety on a population of NPCs we are proposing here a system based on genetic algorithms. We baptised our system GENIE (*Genes Driven Decision Tree*). GENIE allows a game designer to define a set of possible states for every NPC and represent them in term of internal parameters. These internal states can also be seen as the emotional states of each NPC: in the same vein as human behaviour is affected by emotions felt while taking decisions, the behaviour of an NPC is affected by its current state. From a more technical standpoint, the internal parameters can be used to drive a decision trees representing the NPC tactical and/or strategical behaviour. Even a small change in the emotional state will provide a different behaviour from the same NPC. Of course, shifting the problem from defining multiple behaviours to defining a set of internal states is not a real solution. GENIE uses a genetic algorithm to generate internal states and provide multiple, and changing, behaviours to a population of NPCs. The fitness function can be tuned to follow the player reactions and adapt the game difficulty for an optimal user experience.

GENIE has been implemented using the Unity game engine.Inside the unity editor, the game developer can use an ad-hoc graphical user interface to design the decision tree and give a weight to every parameter belonging to the internal, emotional, state.

In this paper we are going to describe in detail how GENIE works and, to prove the effectiveness of our solution, we will present results obtained by testing GENIE on four games of different genres. Experiments have been performed on a first person shooter, a stealth, a role-playing, and a roguelike games. Preliminary results are promising with respect to the general technique of genetic evolution applied to parametric decision making.

The remainder of this paper is organised as follows. In Sec. II relevant related work on the same topic is presented while GENIE is described in Sec. III. Section IV provides a report of the experiments we performed to validate our approach. Section V concludes the paper and outlines future extensions of the current work.

II. Related Work

When discussing related work, we have to remember classical AI and AI for video games use similar techniques to achieve different goals. While classical AI aims to emulate a human being and recreate a perfect behaviour to the purpose to solve a given problem, AI in video games needs to recreate a believable behaviour in a specific context while posing a challenge to the player. Moreover, this challenge must be the result of an imperfect behaviour in order to let the player win with a reasonable effort and progress in the game.

The first historical example of AI applied to games where the player was supposed to confront NPCs exposing different behaviours is the game *Pac-Man* in 1980 [1]. After Pac-Man, to observe significant improvements in commercial products we have to wait almost twenty years. In late 90's, agents in games started using information from the surroundings to influence decision making such as in the case of *GoldenEye 007* [2], *Thief: The Dark Project* [3], and *Metal Gear Solid* [4] where allies' status was taken into account by NPCs. Also in the late 90's, the newborn genre of *Real-Time Strategy* (RTS) introduced the adoption of a very large number of NPCs on the playfield. With RTS games, NPCs started using interaction with one another to implement strategies (like in *Myth; The Fallen Lords* [5]) and movement in formation (like in *Age of Empire II* [6]).

Moving now to a more scientific ground, we should be considering two kinds of research contributions for this work: the ones coming from the field of computer science and those coming from the psychoanalysis field.

In computer science, we can find a large number of contributions addressing the problem of changing behaviour using an algorithmic approach. Despite this availability, only a subset of them are strictly related to gaming.

The field of *Dynamic Decision-Making* (DDM) has already been very well investigated [7], [8]. DDM is interdependent decision-making taking place in an environment that changes over time. The changes can be determined by the decision maker itself or from events that are outside of its control. Despite its complexity and interdependence of decisions, DDM is not exposing different behaviours when the same environmental conditions are met, but aims to define strategies to adapt to unplanned environment changes.

Since DDMs are not up to the task at hand, another opportunity is to offer different behaviours by generating them. Generative approaches are usually applied to games with the aim to develop a human-like behaviour for NPCs. As an interesting example, authors of [9] use cognitive architectures to address the design of believable bots for *First Person Shooter* (FPS) games. A more general approach is discussed in [10], where the problem of believability is also taken into account trough a Turing-like tests performed during live gameplay. In particular, many versions of the Turing test have been proposed in recent years in order to perform believability assessment in the recognition of human-like behaviours; see, e.g., [11] and [12]. Beside FPS games, behaviour generation has been exploited also for other purposes such as the creation of characters to perform interactive storytelling, as reported in [13].

Anyway, generative approaches do not grant a solution to our problem: a new generated behaviour could be too different from the previous one and break continuity in use experience. From this standpoint, a better solution could be to generate an initial (believable) behaviour and then evolve it. Many evolutionary approaches has already been applied to games in various situations. In [14], behaviour trees are evolved and recombined to raise the competition level of an NPC, while in [15] a neural network is used to boost performances of an agent playing an FPS game. Similar concepts has also been applied outside the field of digital entertainment but with similar goals, such as in [16], where robot behaviour is evolved in order to improve survival probability, and [17] to find optimal end moves for the tabletop game *Risk*. Nevertheless, in all the reported examples, evolution is always intended to improve performances through generations and/or outperform a human player. While this is reasonable in games such as racing or chess, the final player experience will be poor in heavily plot-driven games. Moreover, we are still failing in providing more variety to the gameplay.

Among all evolutionary strategies, the use of *genetic algorithms* proved to be an interesting approach to the purpose of this paper. Since a genetic algorithms evolves a population by breeding *eligible* subjects over, time we will not observe sudden and abrupt behaviour changes between generations. Moreover, reproduction eligibility can be tuned for optimal player experience and mutations may boost variety even more.

Genetic algorithms have already been used to evolve soccer players [18]–[20] for the *RoboCup* [1] competition, robots to be trained for space battles [21], opponents in *Real-Time Strategy* (RTS) games [22], tuning of FPS bots [23], and designing chess platers [24]. These approaches have included both the off-line and on-line optimisation of controllers and play strategies.

Unfortunately, all the aforementioned applications of GAs to games are still targeting the evolution of the best possible player. As already discussed, this is not always the request coming from a game designer. To the best of our knowledge, there are no contributions about genetic evolution in games with the purpose to evolve NPCs targeting a condition which is deemed optimal for player experience. In our vision, evolution should not be leading to a specific condition but provide continuous changes to follow the player skill and keep him in the game flow.

Moving now to the side of psychological studies, we already discussed in Sec. I the fact that NPCs internal states can be regarded as different emotional states. To design and tune these internal states, we can leverage on a wide existing literature related to analysis and representation of emotions.

The first studies on the *theory of emotions* date back to 1884 and are represented by the *James-Lange theory of*

[1]https://www.robocup.org/

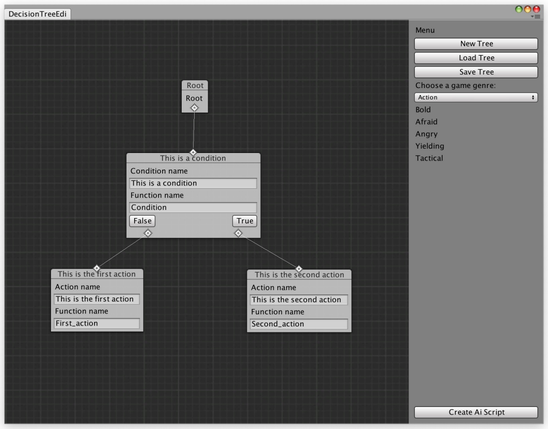
emotion from William James and Karl Lange [25]. Following the James-Lange theory, emotions occur as the result of physiological reactions to events: people have a physiological response to environmental stimuli and their interpretation of that physical response results in an emotional experience. The James-Lange theory has been disputed by Walter Cannon in favour of a more centralised approach known as the *Cannon-Bard theory* [26]. In the Cannon-Bard theory, emotional expression results from the function of hypothalamic structures and emotional feeling results from stimulations of the dorsal thalamus. As a result, the physiological changes and subjective feeling of an emotion in response to a stimulus are separate and independent. A step forward has then been made years later, in 1962, by Stanley Schachter and Jerome Singer. Schachter and Singer unified the James-Lange and Cannon-Bard theories in the *two-factor theory of emotion* [27]. According to the two-factor theory of emotion, when an emotion is felt, a physiological arousal occurs and the person uses the immediate environment to search for emotional cues to label the physiological arousal. When the brain does not know why it feels an emotion, it relies on external stimulation for cues on how to label the emotion. The missing piece in the two-factor theory is an explanation about what is actually generating the emotional experience. A possible answer was provided by Magda Arnold and Richard Lazarus with the *appraisal theory*. In the appraisal theory, emotions are extracted from our evaluations (appraisals or estimates) of events that cause specific reactions in different people. Essentially, the appraisal of a situation causes an emotional, or affective, response that is going to be based on that appraisal. Reasoning and understanding of one's emotional reaction becomes important for future appraisals as well. In particular, Lazarus stated that, to experience an emotion, thought is enough: emotional experience is always the result of a cognitive evaluation of surrounding events [28]. Anyway, a very important contribution, with respect of this paper, is the one provided by Ira Roseman in late 90's [29], [30]. Roseman's theory of appraisal extends the original theory in a structural way and holds that there are certain appraisal components that interact to elicit different emotions. In particular, the combinations of 5 appraisals can trigger 13 qualitatively different emotions in any given situation. The appraisals are: motivational state (rewarding/punishing), situational state (present/absent), probability (certain/uncertain), legitimacy (positive/negative outcome deserved), and causal agency (circumstanced other person/self). The emotions determined by these five appraisals are joy, relief, hope, liking (intended as friendliness), pride, distress, sorrow, fear, frustration, disliking (intended as unfriendliness), anger, regret, and guilt.

Modern psychology defines emotions as reactions – short in time, but strong in intensity – raising as an answer to external (i.e., from the environment) or internal (i.e., from the body) stimuli. These stimuli bring a specific cognitive message and serve the function to divert the attention. Researchers agree that emotions can be classified as primary (fundamental) and secondary. While secondary emotions are difficult to identify, primary emotions are inborn or acquired during the first stage of live. Moreover, lists of primary emotions proposed by various research groups are always covering at least a subset of Roseman's classification.

GENIE is leveraging in this approach and classifies emotions in order to describe NPCs internal state as a discrete elements set.

III. GENIE

GENIE (*Genes Driven Decision Tree*) is a software tool to be used within the Unity game engine. The purpose of this tool is to ease the design of multiple, variated, behaviours for NPCs in a video game. The designer/developer using GENIE can describe a basic behaviour of a template NPC by creating a decision tree using a visual interface (see Fig. 1). Branching



Condition name: This is a condition
Function name: Condition

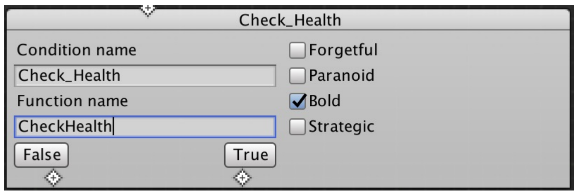
Action name: This is the first action
Function name: First_action

Action name: This is the second action
Function name: Second_action

Choose a game genre: Action
Bold
Afraid
Angry
Yielding
Tactical

Fig. 1. GENIE interface for decision tree editing.

conditions for the generated tree will be triggered by variables defining the internal state of each NPC (see Fig 2). This way,



Condition name: Check_Health
Function name: CheckHealth

☐ Forgetful
☐ Paranoid
☑ Bold
☐ Strategic

Fig. 2. GENIE panel to set emotional parameters.

each NPC will offer a slightly different behaviour depending on the internal state and starting from the template.

In particular, to describe the emotional state, we associate to each emotion a floating point value in the range $[0, 1]$. In this scale, the value 1 means maximum intensity for an emotion while 0 means that the emotion is absent in the NPC. Inside the decision tree, decision nodes can defines a threshold for each emotion. The current intensity of the emotions, i.e., the internal state, will contribute to the branch selection. Of course, emotions are not the only elements in play when selecting a branch: decision nodes, depending on the game, may also need to check the surrounding environment. From the environment, other contextual information will be extracted and used; e.g., collisions status, light intensity, or the player presence and distance.

While the internal state is fixed inside an NPC, emotional changes will take place when spawning new ones. Emotional states change over time is achieved by means of a genetic algorithm. The adoption of a genetic algorithm guarantees a smooth and uniform transition between generations and allows random, unpredictable, changes to be added. The intensity of each emotion is used as a chromosome for the evolution. Since the main goal of this tool is to achieve variation, in our opinion it makes little sense to implement a very sophisticated, and CPU-intensive, genetic algorithm. For this reason, we implement the genetic crossover using the *single-point crossover* algorithm and mutation by selecting a random chromosome (a random emotion) and setting its value to a number in the range $[0, 1]$. The adoption of a lightweight algorithm is also helpful to improve scalability in games where thousands of NPC are required, such as RTS games. Anyway, a separate discussion is required for the selection phase. While the purpose of the selection phase is very simple: picking genomes (NPCs) eligible for breeding, this function is always tightly coupled with game mechanics and player experience. As an example, an FPS might want a fitness function to select NPCs with a long lifespan while avoiding those who already killed the player; this might be reasonable to make the population challenging for the player, but not too powerful. For these reasons, GENIE is not implementing any fitness function by itself, but is providing a $C\#$ delegate for the game developer to use. The delegated function will be used by the GENIE middleware to perform the selection phase.

The final result of the system described above when applied to an FPS game might be as follows. If the player is very skilled, all the NPCs not taking cover (having a low *fear* component in the emotional state) will die quickly. The system might start giving breeding priority to *fearful* NPCs, the coming generations will slowly take more cover and adapt to the player play style. This way, the average NPC lifespan will increase as a proof the game have become more challenging. If, on the other side, the game is getting too challenging, NPCs who killed the player might nor not be selected for breeding and the next generation will evolve from less dangerous NPCs. This is going to make the game easier and prevent player's frustration. If the player improves in skill, the system will adapt by selecting again strong NPCs for the following generation. Furthermore, from time to time, mutation may take place. When a mutation is performed, an NPC will behave in an unexpected way. If this new, unexpected, behaviour is successful (i.e., the NPC survives up to the next generation), it might get selected for breeding and contribute to the next generations.

While implementing of GENIE, a number of choices have been made to find an acceptable compromise between usability and complexity. One of the most crucial point was about making this GENIE prototype a completely general purpose tool or bind it to a specific game genre. On the one hand, a general purpose tool is much more flexible but can prove too difficult to manage due to the sheer number of parameters that comes into play. On the other hand, being specific to a genre would make it very powerful to use in a specifica case, but also much less useful in a real production environment. Moreover, being able to test the effectiveness of our methodology only in one single case did not make much sense to us. As a result, the compromise we found, was to have a general purpose tool but, to increase usability, limited to four mainstream game genres. The four game genres we selected for this implementation are: FPS, stealth, role-playing, and roguelike games. For each genre in this list, we are proposing (after thoughtful discussion with game designers) a reduced set of emotions deemed useful to evolve the specific NPCs. The emotions selected of each game genre are reported in Tab. I. Nevertheless, given the object-oriented nature of Unity, it is possible for the final user to easily extend these sets or support new games genre.

FPS	**Stealth**	**Role-playing**	**Roguelike**
Afraid	Bold	Anxious	Angry
Angry	Forgetful	Cautious	Coward
Bold	Paranoid	Considerate	Greedy
Tactical	Strategic	Panicked	
Yielding		Self-Assured	
		Shy	

TABLE I
EMOTIONS ASSOCIATED TO GENRES INSIDE GENIE.

Moreover, to make things easier to the developer, we also decided to implement binary decision trees. This way, the implementation of decision delegates will be easier, boosting code reusability, and the trees will be more manageable from inside the editor. Anyway, this is not going to be an actual limitation; because it has been demonstrated that the expressivity of binary decision trees is not a subset of the generic multi-branch version.

IV. Experimental Evaluation

To evaluate the actual effectiveness of GENIE, we performed experiments with games implemented using our tool. A game for every supported genre has been implemented and tested with actual players.

For the evaluation, we engaged a group of 20 volunteers. This group was made of Computer Science students with an age between 23 and 28 years. In the group, we had 18 males and 2 females. All the subjects declared to be active players and to be familiar with all the proposed genres.

To perform the experiments, we asked every volunteer to have a play session with each game and then fill in a feedback form. With the feedback form we aimed to understand if the player was actually perceiving a difference in the behaviours of the NPCs in game. Perceiving differences means, in this case, that a player is actually taking note of an NPC performing actions in line with its emotional status. To prove the effectiveness of GENIE, we need to have the majority of actions in their emotional context being apparent to a large majority of the players. During analysis, for each experiment (genre), we classified the couples $[action, emotion]$ in three groups based on the the share of players that perceived them: 66% or more, between 33% and 65%, and less than 33%.

In order to be able to compare results, demo levels have been designed following a common structure. For FPS and stealth games, where players are usually required to follow a given path, we adopted a linear level structure where three gameplay events (missions) are proposed in sequence, as depicted in Fig. 3. To complete the game, the player must survive all the events. For role-playing and roguelike levels instead, where

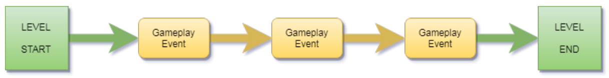


Fig. 3. Linear level organisation for FPS and stealth experiments.

the player has more freedom to roam on the map, we adopted a non-linear approach as reported in Fig. 4. In this non-linear

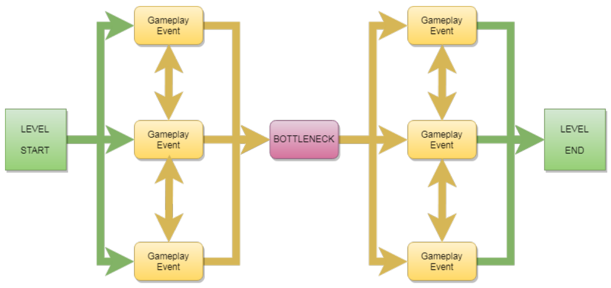


Fig. 4. Non-Linear level organisation for role-playing and roguelike experiments.

approach there are three events available in the first part of the level. After surviving all the events in any order, the player can access the second part of the level through a bottleneck section. In the second part, the same pattern encountered before the bottleneck is proposed again in order to end the level.

In all the games used for testing, fitness functions have been implemented using a scoring system. A score is associated to NPC actions; when the fitness function is run, the NPC history is evaluated and an integer number is returned. NPCs with the highest score are selected for breeding. As an example, the scoring system for NPCs in the role-playing experiment is reported in Tab. II

Action	**Score**
Kill the player	+2
Cure allies	+1
Inflict basic damage	+1
Inflict ability damage	+1
Recover life	+1
Survive combat for less than 3 seconds	-2
Survive combat for 3 to 5 seconds	-1
Survive combat for 5 to 7 seconds	+0
Survive combat for more than 7 seconds	+1

TABLE II
SCORING SYSTEM FOR THE ROLE-PLAYING EXPERIMENT.

In the remainder of this section, we will first describe the four games we used for evaluation and then experiment results will be discussed.

A. First Person Shooter Experiment

In an FPS, the player is engaged in a weapon-based combat simulation using a first-person perspective. During the game, a sequence of missions must be completed while fighting waves of NPCs. The artificial intelligence driving the skills of the NPCs is significative of the level of difficulty offered by the game. In this kind of game, NPCs should usually vary the attack style, how they move around, and how they take cover.

For this experiment we implemented a level about a combat zone in a harbour (Fig. 5). On the map, players and NPCs

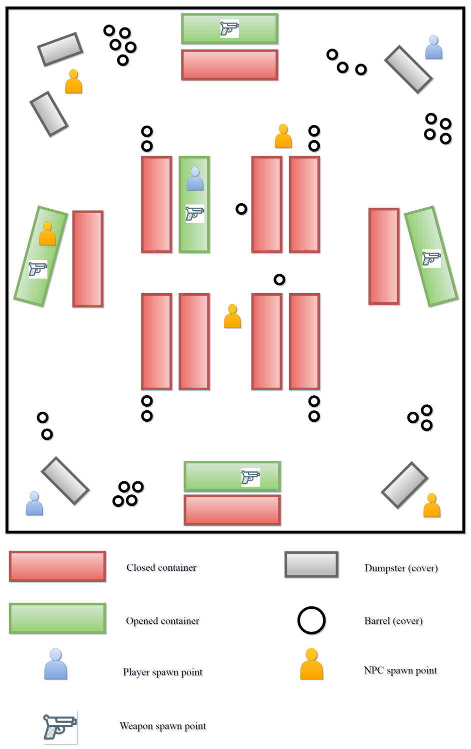


Fig. 5. Annotated map for the FPS level.

can find environmental elements offering shelter. Some of this elements are containers providing also ammunitions (red rectangles) or safe passage between zones (green rectangles). Player spawn points are placed under cover and far away from NPCs spawn points.

During gameplay, a player must confront with waves composed by five NPCs. The player will hold a weapon with infinite ammunitions while each NPC will have random equipment.

The decision tree for the NPCs is really complex because a lot of conditions must be evaluated. First, we have to check if the player is in the range of visibility, then considerations about the current weapon and other equipment available on the fields are drawn. Just to give a couple of examples, taking cover takes precedence if the NPC is afraid to fight while being bold is pushing to engage fighting even if the NPC have a melee weapon (the player have always a rifle). A screenshot of the game when dealing with two tactical-inclined NPCs (ping particle effect) is shown in Fig. 6.

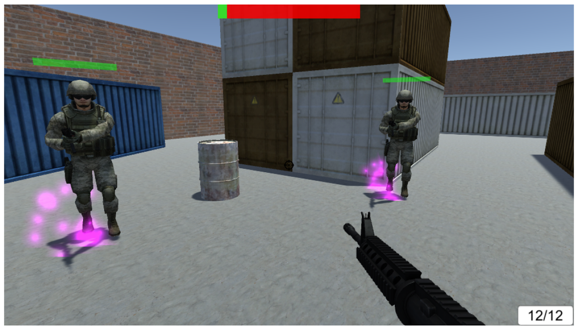


Fig. 6. Screenshot during the FPS experiment.

B. Stealth Experiment

In a stealth-based game, the player is required to move while staying undetected across an area guarded by NPCs. The player is usually subject to a swift death when confronting an NPC directly. In this kind of games, the difficulty offered to the player depends on the movement pattern, lever of awareness, and sensing sensibility of the NPCs. To put variety in a stealth game, NPCs should vary their pause/movement pattern as well as their policy about looking for (or chasing) the player.

Our stealth game is set inside a small museum (Fig. 7). In this museum, guards are deployed to protect the artworks.

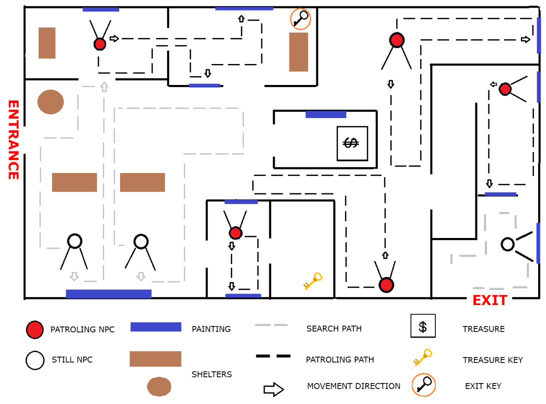


Fig. 7. Annotated map for the stealth level.

The player must traverse the map to steal a treasure and then leave the premise. Along the way, the player needs to retrieve two keys: one to access treasure and one to open the last exit door. While doing this, the player must remain unseen from the NPCs and avoid generating sounds bumping into obstacles. When the player is detected, all NPCs will converge to the point where the something has been spotter or a sound heard.

During gameplay, NPCs can be static or patrolling an area using a pre-determined path. The player is equipped with a torch to increase environment visibility and a crowbar to stun guards from the back; both, when used, increases the chances to be detected. The game ends whenever the player is caught by a guard.

The decision tree for the stealth experiment is simpler than in the previous case due to a reduced number of decisions to take. Nevertheless, it must now also include the possibility to coordinate with other NPCs. Moreover, decisions are taken also based on the level of detection (*Hi*, *Medium*, or *Low*, depending on distance) that each NPC has about the player. When the player has been detected the number of nearby NPCs comes into play and the guard will start chasing the player only when bold; otherwise, it will move away and call for support.

C. Role-Playing Experiment

A *Role-Playing Game* (RPG) is a game in which players assume the roles of characters in a fictional setting. In this kind of video games, each player controls the actions of a character immersed in a well-defined world. A character must overcome a sequence of challenges in order to progress in level and become more powerful. These challenges may be posed by other players (in *Player versus Player*, or PvP settings) or come from the surrounding environment (in *Player versus Environment*, or PvE settings). Dealing with NPCs sets us in a PvE setting.

In RPG games, each NPC usually falls in a specific category and has its own statistics. Artificial intelligence must be specialised for each category, which should also evolve independently. Differently from the first two genres, NPCs in a RPG game may also interact peacefully with the player and are ofter part of the narrative. This means that, beside combat, the NPC emotional state should have effects on conversations and relationship.

The RPG game we implemented is located in a dungeon made of rooms connected by means of corridors (Fig. 8). This

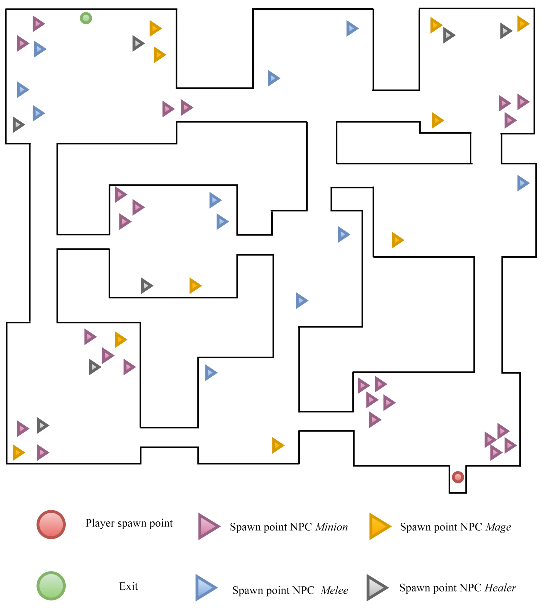


Fig. 8. Annotated map for the RPG level.

dungeon, in the style of fantasy roleplay games, is populated by fancy creatures. In this game we implemented four different NPCs: *Minion*, *Melee*, *Mage*, and *Healer*. The Minion and Melee are close-combat unit with different attack power, while Mage is a ranged combat unit, and the Healer will support other NPCs in the area by healing them.

Like many titles in this genre, the map layout is such that the player will be required to face enemies of increasing difficulty. The game ends when the health of the player is depleted.

In this experiment we deal with multiple decision trees: one for each NPC class. Even if the emotional traits are shared, each class will behave in its own way, with respect to the same emotion, during the game. As an example, a cautious Healer will try to get out of danger before healing others while a cautious Meele will pay more attention to its own health level.

D. Roguelike Experiment

The roguelike genre is a sub-category of Role-plan genre. A roguelike game is usually set in a fantasy, or medieval, environment and requires the player to crawl through a dungeon to kill monsters (NPCs), collect treasures, and interact with the environment. This kind of games are usually featuring a procedural level generation and a permanent death mechanic.

In roguelike games, artificial intelligente of NPCs is typically limited to movement and attack strategies. With respect to movement, an NPC should understand when (and how) to chase or retreat from the player. For the attack phase, the artificial intelligence is in charge to choose which weapon to use and if allies are required to engage the player.

Since this genre is linked to RPGs, the level we implemented (Fig. 9) shares some traits (such as the environment setting and the NPCs deployment style) with the previous case. Differently from the RPG experiment, the player have to follow a specific path to reach the final treasure and it is now required to overcome all the enemies along the way.

The player, in this game, is equipped with a ranged magical weapon and can collect power ups as loot from slayed NPCs. A screenshot during gameplay is reported in Fig. 10.

Like in the previous case, in this experiment we are dealing with a number of decision trees. In particular, we defined here seven different classes of NPCs, each one with special abilities. As mentioned, due to the nature of the game, all decision trees are quite simple and combat-oriented.

E. Experimental Results

As already mentioned, we classified each action associated to a specific emotion in three groups based on the the share of players that perceived them. The outcome from the feedback forms is summarised in Tab. III. As we can see in the table, a vast majority of the actions relative to each emotion was apparent to more than two thirds of the players. From these numbers, it seems that the strategy adopted by GENIE is successful in producing different behaviours which are evident to the players.

In order to cross-check this result, we ran another set of experiments with a different feedback form. In this second set,

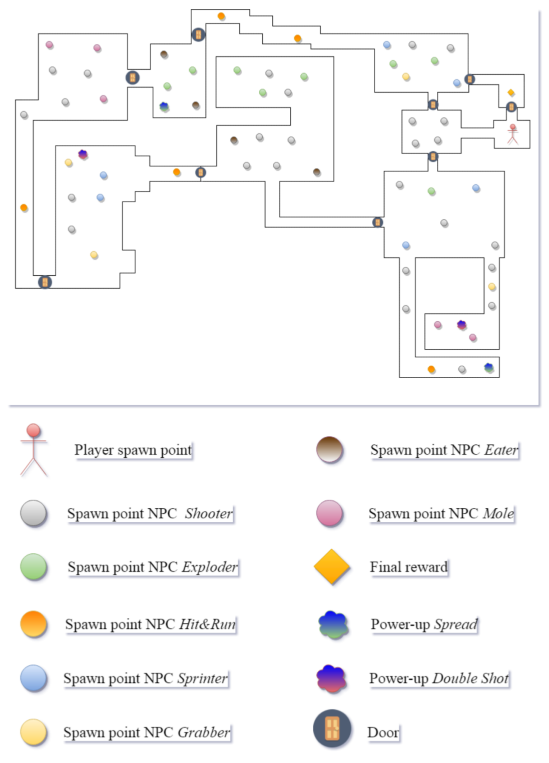

Player spawn point
Spawn point NPC *Eater*
Spawn point NPC *Shooter*
Spawn point NPC *Mole*
Spawn point NPC *Exploder*
Final reward
Spawn point NPC *Hit&Run*
Power-up *Spread*
Spawn point NPC *Sprinter*
Power-up *Double Shot*
Spawn point NPC *Grabber*
Door

Fig. 9. Annotated map for the roguelike level.

Fig. 10. Screenshot during the roguelike experiment.

we aimed to understand if the player could actually tell the different emotional states of the NPCs. Volunteers have been instructed in advance about the different emotions available in the game. Then, after the play session they reported how well the emotion was represented by the actions of each NPC.

Results show that there is actually a perceivable link between emotions and actions since the majority of feedbacks

Share of players	**Experiment**			
	FPS	**Stealth**	**Role-play**	**Roguelike**
less than 33%	1	0	0	0
between 33% and 66%	2	1	3	0
more than 66%	12	6	11	8

TABLE III
SUMMARY OF PERCEIVED ACTIONS DURING GAMEPLAY.

reported a good in-game representation, or better. Anyway, we observed one exception when analysing the FPS experiment, as reported in Tag. IV. As we can see, the yielding emotions

Emotion	Perception				
	Poor	Fair	Good	Very good	Perfect
Afraid	1	1	5	5	1
Angry	0	1	3	8	1
Bold	0	0	8	4	1
Tactical	1	1	7	2	2
Yielding	1	6	5	1	2

TABLE IV
AGGREGATED FEEDBACK ABOUT EMOTION REPRESENTATION FOR THE FPS EXPERIMENT.

seems to be the only one not well represented, trending to a *fair* rating. To understand this phenomenon, we did some oral interviews with the volunteers. After the interviews, our hypothesis is that, even if yielding is important for a game designer to characterise an NPC, the player is not expecting to actually see it on the battlefield; therefore, attention for that kind of behaviour was low during the experiment. Anyway, this is a clear indication that further research about how emotions and behaviours are attracting the player attention is required.

V. CONCLUSION AND FUTURE WORK

In this paper we presented GENIE: a tool to support game designers in the creation of a wide variety of behaviours. GENIE is based on discrete representation of emotional states used a genomes for a generic algorithm. Emotional states, used to drive decision trees, are evolved to adapt to the player needs and to enrich player experience by offering variety during gameplay.

Experimental evaluation on four games provided promising preliminary results supporting the validity of our approach.

Possible extensions of the current work are the inclusion of other game genres, and a study about how it could be possible to evolve the emotional state together with the NPC, like in [31].

REFERENCES

[1] Toru Iwatani, "Pac-Man," Midway Games, 1980.

[2] Rare Limited, "GoldenEye 007," Nintendo, 1997.

[3] Tim Stellmach, "Thief: The Dark Project," Eidos Interactive, 1998.

[4] Hideo Kojima, "Metal Gear Solid," Konami Computer Entertainment Japan, 1998.

[5] Jason Jones, "Myth; The Fallen Lords," Bungie, 1997.

[6] Bruce Shelley, "Age of Empire II," Microsoft, 1999.

[7] W. D. Edwards, "Dynamic decision theory and probabilistic information processing," *Human Factors*, vol. 4, pp. 59–73, 04 1962.

[8] B. Brehmer, "Dynamic decision making: Human control of complex systems," *Acta psychologica*, vol. 81, pp. 211–41, 01 1993.

[9] R. Arrabales, J. Muñoz, A. Ledezma, G. Gutierrez, and A. Sanchis, *A Machine Consciousness Approach to the Design of Human-Like Bots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 171–191.

[10] J. M. L. Asensio, J. Peralta, R. Arrabales, M. G. Bedia, P. Cortez, and A. L. Pea, "Artificial intelligence approaches for the generation and assessment of believable human-like behaviour in virtual characters," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7281 – 7290, 2014.

[11] P. Hingston, "A turing test for computer game bots," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 3, pp. 169–186, Sep. 2009.

[12] ——, "A new design for a turing test for bots," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, Aug 2010, pp. 345–350.

[13] M. Cavazza, F. Charles, and S. J. Mead, "Planning characters' behaviour in interactive storytelling," *The Journal of Visualization and Computer Animation*, vol. 13, no. 2, pp. 121–131, 2002.

[14] C.-U. Lim, R. Baumgarten, and S. Colton, "Evolving behaviour trees for the commercial game defcon," in *Applications of Evolutionary Computation*, C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcazar, C.-K. Goh, J. J. Merelo, F. Neri, M. Preuß, J. Togelius, and G. N. Yannakakis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 100–110.

[15] J. Schrum, I. V. Karpov, and R. Miikkulainen, *Humanlike Combat Behavior via Multiobjective Neuroevolution*. Springer Berlin Heidelberg, 2012, pp. 119–150. [Online]. Available: http://nn.cs.utexas.edu/?schrum:believablebots12

[16] D. Floreano and L. Keller, "Evolution of adaptive behaviour in robots by means of darwinian selection," in *PLoS biology*, 2010.

[17] J. M. Vaccaro and C. C. Guest, "Planning an endgame move set for the game risk: a comparison of search algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 641–652, Dec 2005.

[18] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-evolving soccer softbot team coordination with genetic programming," in *RoboCup-97: Robot Soccer World Cup I*, H. Kitano, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 398–411.

[19] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone, "Evolving soccer keepaway players through task decomposition," *Machine Learning*, vol. 59, no. 1, pp. 5–30, May 2005.

[20] S. Luke, "Genetic programming produced competitive soccer softbot teams for RoboCup97," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds. University of Wisconsin, Madison, Wisconsin, USA: Morgan Kaufmann, 22-25 1998, pp. 214–222.

[21] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, Dec 2005.

[22] S. J. Louis and C. Miles, "Playing to learn: case-injected genetic algorithms for learning to play computer games," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 669–681, Dec 2005.

[23] N. Cole, S. J. Louis, and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, June 2004, pp. 139–145 Vol.1.

[24] A. Hauptman and M. Sipper, "Gp-endchess: Using genetic programming to evolve chess endgame players," in *Genetic Programming*, M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, and M. Tomassini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 120–131.

[25] P. J. Lang, "The varieties of emotional experience: A meditation on james-lange theory," *Psychological Review*, vol. 101, no. 2, pp. 211–221, 1994.

[26] W. B. Cannon, "The james-lange theory of emotions: A critical examination and an alternative theory," *The American Journal of Psychology*, vol. 39, no. 1/4, pp. 106–124, 1927.

[27] S. Schachter and J. E. Singer, "Cognitive, social, and physiological determinants of emotional state," *Psychological Review*, vol. 69, no. 5, pp. 379–399, 1962.

[28] R. Lazarus, J. R. Averill, and E. M. Opton, "Toward a cognitive theory of emotion," in *Proceedings - Third international symposium on feelings and emotions*, M. Arnold, Ed. New York: Academic Press, 1970.

[29] I. J. Roseman, "Appraisal determinants of discrete emotions," *Cognition and Emotion*, vol. 5, no. 3, pp. 161–200, 1991.

[30] ——, "Appraisal determinants of emotions: Constructing a more accurate and comprehensive theory," *Cognition and Emotion*, vol. 10, no. 3, pp. 241–278, 1996.

[31] A. Guarneri, D. Maggiorini, L. Ripamonti, and M. Trubian, "Golem: Generator of life embedded into mmos," *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, no. 25, pp. 585–592, 2013.