ADDING VARIETY IN NPCS BEHAVIOUR USING EMOTIONAL STATES AND GENETIC ALGORITHMS: THE GENIE PROJECT

Federica Agliata[†], Marcello Bertoli[†],
Laura Anna Ripamonti[*], Dario Maggiorini[*] and Davide Gadia[*]
Department of Computer Science, University of Milan
via Celoria 18, 20133 - Milan (Italy)
E-mail: [†]`{firstname.lastname}@studenti.unimi.it`, [*]`{firstname.lastname}@unimi.it`

KEYWORDS
Games, Artificial Intelligence, NPC Behaviour, Genetic Algorithms

ABSTRACT

In recent years we have been observing an increasing adoption of artificial intelligence in video games. With the increasing availability of powerful hardware and advanced algorithms, we can now scale up the quality of the AI available to every Non Playing Character (NPC). Despite this increased quality, 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 internal parameters representing its state.

INTRODUCTION

As of today, we can observe an increasing adoption of artificial intelligence in video games. In modern video games, artificial intelligence techniques are used in a wide range of activities; 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.
Despite the possibility to describe very articulated NPCs behaviour, every NPC will get predictable over time. 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 this amusing or challenging. 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.
To support game designers in providing behavioural variety on a population of NPCs we propose 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 be seen as emotional states for the 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, internal parameters can be used to drive a decision tree representing the NPC tactical and/or strategical behaviour. GENIE uses a genetic algorithm to generate internal states and provide multiple, and changing, behaviours to a population of NPCs. The fitness function of the genetic algorithm can be tuned to follow player's reactions and adapt game difficulty for an optimal user experience.
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 belonging to different genres.

RELATED WORK

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. 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*, *Thief: The Dark Project*, and *Metal Gear Solid* where allies' status was taken into account. 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.
Moving now to a more scientific ground, we can find a large number of contributions addressing the problem of changing behaviours using an algorithmic approach. To the best of our knowledge, despite this wide literature, only a subset seems to be actually related to gaming.
Generative approaches are usually applied to games with the aim to develop a human-like behaviour for NPCs. As an interesting application, (Arrabales et al., 2012) uses cognitive architectures to address the design

of believable bots for *First Person Shooter* (FPS) games. A more general approach is discussed in (Asensio et al., 2014), where the problem of believability is also taken into account trough Turing-like tests performed during live gameplay.

Despite their applicability, generative approaches do not provide a solution to the problem addressed in this paper: a new generated behaviour might be too different from the previous one and break continuity in use experience. A better solution could be to generate an initial (believable) behaviour and then evolve it. In (Lim et al., 2010), behaviour trees are evolved and recombined to raise the competition level of an NPC, while in (Schrum et al., 2012) a neural network is used to boost performances of an agent playing an FPS game. In (Floreano and Keller, 2010), a robot behaviour is evolved in order to improve survival probability, while in (Vaccaro and Guest, 2005) evolutionary computation is used to find optimal end moves for the tabletop game *Risk*. In all examples above, evolution is intended as a way to improve performances through generations and outperform a human player. While this is reasonable, to some degree, in games such as racing or chess, the resulting player experience will be poor in plot-driven games. NPCs must not be unbeatable: they are supposed to provide a reasonable challenge to the player and accompany her through the skill progression during the game.

The use of *Genetic Algorithms* (GAs) proved to be an interesting approach to the purpose of this paper. Since a genetic algorithm evolves a population by breeding *eligible* subjects over time, we should not observe sudden and abrupt behaviour changes between generations. Moreover, eligibility to reproduction can be tuned for optimal player experience.

Genetic algorithms have already been used to evolve soccer players (Whiteson et al., 2005) for the *RoboCup* competition, robots to be trained for space battles (Stanley et al., 2005), opponents in RTS games (Louis and Miles, 2005), tuning of FPS bots (Cole et al., 2004), and designing chess platers (Hauptman and Sipper, 2005).

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

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. Branching conditions for the generated tree will be triggered by variables defining the internal (emotional) state of each NPC. This way, each NPC will offer a slightly different behaviour depending on its state, starting from the template.

In particular, the internal state of each NPC is defined by a set of possible emotion that the NPC can "feel". To each emotion, we associate 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. The combined values of all emotions represent the emotional state of the NPC.

Inside the decision tree, decision nodes can define a threshold for each emotion. Emotions are not the only elements in play when selecting a branch: decision nodes may also need to check the surrounding environment, depending on the game.

While the emotional state is fixed inside each NPC, changes will take place when breeding new generations using a GA. The adoption of a GA grants a smooth and uniform transition between generations and allows random, unpredictable, changes to be added thanks to mutations. The intensity of each emotion is used as a chromosome for the evolution. We implemented the genetic crossover using the *single-point crossover* algorithm and mutation by selecting a random chromosome (a random emotion) and setting it to a random valuer. The adoption of this lightweight algorithm is also helpful to improve scalability in games where thousands of NPCs are required, such as RTS games. While the purpose of the selection phase is very simple: picking genomes (NPCs) eligible for breeding, this operation is always tightly coupled with game mechanics and player experience. As an example, an FPS game might want 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 delegating the evaluation to the developer.

While implementing GENIE, a number of choices have been made to find an acceptable compromise between usability and complexity. The compromise we found is to have a general purpose tool but, to increase usability, we limited it 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. 1. Anyway, given the object-oriented nature of Unity, it is possible for the final user to easily extend these sets and support new game genres. Moreover, to make de designing phase easier, we also decided to implement binary decision trees. This is not going to be an actual limitation because it has been demonstrated that

Table 1: Emotions Associated to Genres Inside GENIE

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

the expressivity of binary decision trees is not a subset of the generic multi-branched version.

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.
We asked every volunteer to have a play session with each game and then fill in a feedback form. With this feedback form we aimed to understand if the player was actually perceiving a difference in the behaviours of the NPCs while playing. During analysis, for each experiment (genre), we classified the couples $[action, emotion]$ in three groups based on 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. 1. To complete the level, the player must survive all the events. For role-playing

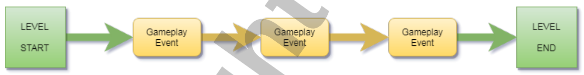

Figure 1: Linear Level Organisation for FPS and Stealth Experiments

and roguelike levels, where players have more freedom to roam the map, we adopted a non-linear approach as reported in Fig. 2. In this non-linear 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 beat the level.

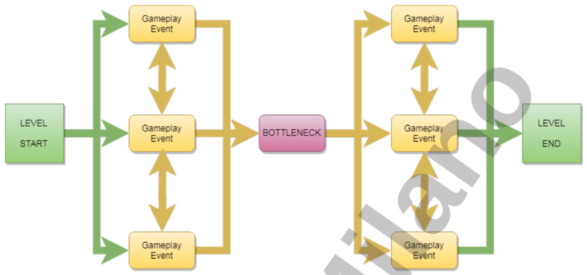

Figure 2: Non-Linear Level Organisation for Role-Playing and Roguelike Experiments

In all the games used for testing, fitness functions have been implemented using a scoring system. A score is associated to each NPC action: when the fitness function is run, the NPC history is evaluated and an integer number is returned. NPCs reporting the highest score are selected for breeding.

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 games, NPCs should usually vary the attack style, how they move around, and how they take cover.
For this experiment we implemented a level with a combat zone in a harbour. On the map, players and NPCs can find environmental elements offering shelter. Some of this elements are containers providing ammunitions or safe passage between zones. During gameplay, a player must confront with waves composed by five NPCs.
In the decision tree of the NPCs, 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 has always a rifle). A screenshot of the game when dealing with two tactical-inclined NPCs (pink particle effect) is shown in Fig. 3.

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

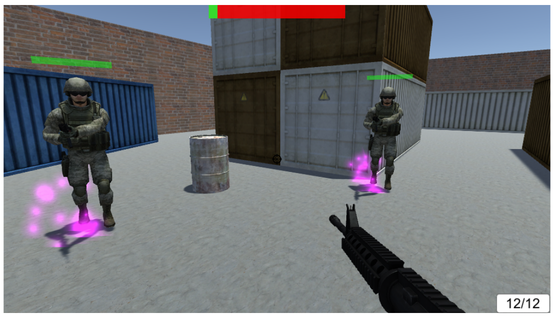

Figure 3: Screenshot During the FPS Experiment

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. In this museum, guards are deployed to protect the artworks. 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 the treasure and one to open the exit door. While doing this, the player must remain unseen from the NPCs and avoid generating sounds by bumping into obstacles. When the player is detected, all NPCs will converge to the point where 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 exits the building or is caught by a guard.
The decision tree for the stealth experiment is simpler than in the previous case but must include the possibility to coordinate with other NPCs. When the player is detected, the number of nearby NPCs comes into play and the guard might start chasing the player or move away and call for backup.

Role-Playing Experiment

A *Role-Playing Game* (RPG) is a game in which players assume the roles of characters in a fictional setting. A character must overcome a sequence of challenges in order to progress in experience and become more powerful. 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.
The RPG game we implemented is located in a dungeon made of rooms connected by corridors. This dungeonis populated by fancy creatures. In this game we implemented four different NPCs: *Minion*, *Melee*, *Mage*, and *Healer*. 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.
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.

Roguelike Experiment

The roguelike genre is a sub-category of the RPG genre. A roguelike game requires the player to crawl through a dungeon to kill monsters (NPCs), collect treasures, and interact with the environment. In roguelike games, artificial intelligente of NPCs is typically limited to movement and attack strategies. Differently from the RPG experiment, the player has to follow a specific path to reach the final treasure and it is required to overcome all the enemies along the way. The player is equipped with a ranged magical weapon and can collect power ups as loot from slayed NPCs. A screenshot during gameplay is proposed in Fig. 4.

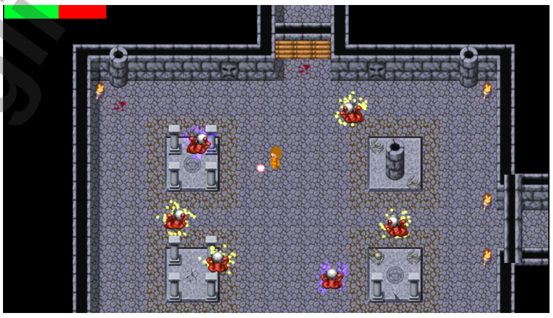

Figure 4: Screenshot During the Roguelike Experiment

Like in the previous case, in this experiment we are dealing with multiple decision trees. In particular, we defined seven different classes of NPCs, each one with different special abilities.

Experimental Results

As already mentioned, we classified each action associated to a specific emotion in three groups based on the share of players that perceived them. The outcome from the feedback forms is summarised in Tab. 2. 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.
In order to cross-check this result, we ran another set of experiments with a different feedback form. In this second set, we aimed to understand if the player could actually tell apart the different emotional states of the NPCs. Volunteers have been instructed in advance about the different emotions available in the game. Then, after

Table 2: Summary of Perceived Actions During Gameplay

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

each 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 reported a good in-game representation. Anyway, we observed one exception when analysing the FPS experiment, as reported in Tag. 3. As we can see, the

Table 3: Aggregated Feedback About Emotions Representation for the FPS Experiment

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

yielding emotion 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.

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 as genomes for a generic algorithm. Emotional states, used to drive decision trees, are evolved to adapt NPCs to the player's 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 for validation of other game genres and a study about how it could be possible to evolve the emotional state together with the NPC skills and features, like in (Guarneri et al., 2013).

REFERENCES

Arrabales R.; Muñoz J.; Ledezma A.; Gutierrez G.; and Sanchis A., 2012. *A Machine Consciousness Approach to the Design of Human-Like Bots*, Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-32323-2, 171–191.

Asensio J.M.L.; Peralta J.; Arrabales R.; Bedia M.G.; Cortez P.; and Pea A.L., 2014. *Artificial Intelligence approaches for the generation and assessment of believable human-like behaviour in virtual characters. Expert Systems with Applications*, 41, no. 16, 7281 – 7290. ISSN 0957-4174.

Cole N.; Louis S.J.; and Miles C., 2004. *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, 139–145 Vol.1.

Floreano D. and Keller L., 2010. *Evolution of Adaptive Behaviour in Robots by Means of Darwinian Selection.* In *PLoS biology.*

Guarneri A.; Maggiorini D.; Ripamonti L.; and Trubian M., 2013. *GOLEM: Generator Of Life Embedded into MMOs. The 2018 Conference on Artificial Life*, 585–592.

Hauptman A. and Sipper M., 2005. *GP-EndChess: Using Genetic Programming to Evolve Chess Endgame Players.* In *Genetic Programming.* Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-31989-4, 120–131.

Lim C.U.; Baumgarten R.; and Colton S., 2010. *Evolving Behaviour Trees for the Commercial Game DEFCON.* In *Applications of Evolutionary Computation.* Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-12239-2, 100–110.

Louis S.J. and Miles C., 2005. *Playing to learn: case-injected genetic algorithms for learning to play computer games. IEEE Transactions on Evolutionary Computation*, 9, no. 6, 669–681. ISSN 1089-778X.

Schrum J.; Karpov I.V.; and Miikkulainen R., 2012. *Humanlike Combat Behavior via Multiobjective Neuroevolution*, Springer Berlin Heidelberg. 119–150. URL `http://nn.cs.utexas.edu/?schrum:believablebots12`.

Stanley K.O.; Bryant B.D.; and Miikkulainen R., 2005. *Real-time neuroevolution in the NERO video game. IEEE Transactions on Evolutionary Computation*, 9, no. 6, 653–668. ISSN 1089-778X.

Vaccaro J.M. and Guest C.C., 2005. *Planning an endgame move set for the game RISK: a comparison of search algorithms. IEEE Transactions on Evolutionary Computation*, 9, no. 6, 641–652. ISSN 1089-778X.

Whiteson S.; Kohl N.; Miikkulainen R.; and Stone P., 2005. *Evolving Soccer Keepaway Players Through Task Decomposition. Machine Learning*, 59, no. 1, 5–30. ISSN 1573-0565.