Procedural Content Generation and Evolution within the EvoTanks Domain

[Extended Abstract]

Adam Pace
School of Computing and Mathematics
University of Derby
Derby, UK
a.pace1@unimail.derby.ac.uk

Tommy Thompson[*]
School of Computing and Mathematics
University of Derby
Derby, UK
t.thompson@derby.ac.uk

ABSTRACT

In this extended abstract we highlight ongoing work in developing procedural generation for the purposes of games. This research focusses on our recent development of a procedural generation framework developed in the EvoTanks domain, a re-imagining of the Atari 2600 title *Combat*. In addition, we highlight current work that utilises evolutionary algorithms not only to procedurally generate offensive and defensive content for the game, but its use to ensure game-balance is retained.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications & Expert Systems—*Games*; I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning, Connectionism and neural nets*

General Terms

Algorithms, Design, Experimentation

Keywords

Procedural Content Generation, Genetic Algorithms, Games

1. INTRODUCTION

As video games increase in size and scope, the pressure builds on developers to create content that ensures a fresh and interesting experience for the player. This content can range from core gameplay elements such as quests or levels, to more open-ended topics such as terrain, weapons, items or even characters; pretty much anything the player can see or interact with. One way to reduce the amount of content that must be developed — and by extension the associated costs — is through Procedural Content Generation (PCG): algorithmically generating content using a given procedure and a set of rules that define its creation. While academic research has grown in this area, very few 'AAA' games employ PCG — arguably developers cannot risk releasing untested content into their products. However, provided generated content is optional, these risks are somewhat reduced since the player will simply ignore it [8].

[*]Tommy Thompson is a member of the Distributed and Intelligent Systems Research Group (DISYS).

One area PCG is suited for is weapons, specifically guns. Provided the game continually offers new guns, this fits the rule of being optional content: provided PCG guns can fire, they are never broken - they may simply be bad. The only notable example of a AAA title adopting PCG for weapons is the *Borderlands* series [1, 2]. While guns have received some attention in commerical products, shields are often neglected or deemed secondary [1]. Typically these are a fixed mechanic rather than something the player can change to better suit their play styles. Given that gameplay balance is driven by the ability to not only attack, but defend, we feel there is opportunity to explore what PCG could achieve when developing weapons and shields based upon gameplay.

In this extended abstract, we highlight current work investigating PCG for weapons and shields. By adopting evolutionary algorithms, we hope to create a prototype for a framework that could be adopted in other games. This initial phase is conducted within the EvoTanks [7, 6] domain. We also highlight recent work to apply co-evolution to test the flexibility of this content: exploiting the arms race dynamic between weapons and shields to ensure gameplay balance is retained whilst providing interesting products for gameplay.

2. RELATED WORK

The *Borderlands* series generates an excessively large number of weapons for the player. The first game [1] holds the Guinness World Record for the most guns in a videogame: 17,750,000 [3]. This feature is a driving mechanic of the game, with the prospect of new weaponry maintaining player interest. However, *Borderlands* weaponry is not customised for the player.

By contrast, Galactic Arms Race (GAR) [4] explores player-driven PCG in games. Weapons are developed using the content-generating NeuroEvolution of Augmenting Topologies (cgNEAT) algorithm: a form of neuro-evolution that

[1]While shields are procedurally generated in *Borderlands*, the emphasis is placed upon the weaponry

not only tailors weights but network configurations [5]. In GAR weapons are generated based upon player preferences both locally and across the server. This serves two purposes, firstly broken weapons are filtered out by players due to lack of use. Secondly, if players enjoy weapons with unconventional methods: such as spirals of particles the game will start to provide more weapons matching those properties. The neural network for each gun controls how fired particles look and behave, creating an interesting array of weaponry both functionally and cosmetically. The behaviour of these particles directly dictate how the gun performs. One drawback to this is the limited domains it can be applied to. This form of weaponry is ill-fitted to the popular genre of military first-person shooters (FPS); it wouldn't fit the theme. This is where we feel an approach that models gun behavior and properties could work well and have greater flexibility.

3. OVERVIEW OF DEVELOPMENT

We have built a weapon and shield framework that fits into the existing game developed in C# using *XNA* 4.0. These are represented by properties that determine effectiveness. For example weapons are measured by damage per shot, accuracy, fire rate and range. Meanwhile, shields are represented by maximum health, damage threshold, recharge rate and delay and recharge type. The latter determines if the shield recharges when fully depleted or can start to recharge whenever it hasn't been hit for an amount of time greater than the delay. This is similar to how a number of modern FPS's would represent their weapons and shields, albeit somewhat simplified.

Through co-evolution tournaments inspired by work in [7], tanks fight one another within the population as means of assessing fitness of not only tank performance but by extension the gun and/or shield employed. To ensure a varied learning process, both elitism and roulette selection are employed to dictate the subsequent generations lineage.

Having completed initial testing to prove the frameworks validity, we ran tests to ensure that weapons and shields both are evolved to adapt to the continued shifts in population trends. In addition, we number of tests were conducted to experiment with learning parameters; crossover and mutation rate, population size and tournament size in order to ascertain the most effective or interesting outcomes for future tests. In addition, a number of different fitness algorithms have been devised to try and balance the type of weapons created and find an algorithm that best suits our requirements. This was conducted upon observing that assessment based solely on damage dealt produces a completely different weapon to one scored on how many shots fired to kill the opponent.

3.1 Current Work

Currently we are focussing on two areas for further development: investigating the impact of player behaviour on weaponry and shields and the challenges raised in balancing this content.

Initial tests showed that with different types of AI controlling the tanks, the weapons adapt to different situations. One avenue being considered is to evolve not only the weapon and the shield but also the tank too; with the intention of creating bespoke content for an evolved NPC.

In addition, we are continuing to experiment with the results of the co-evolution and applying it as means to re-evaluate the balance of this content. Recent experiments indicated that shields are capable of compensating for types of weapons evolved, almost to the point where they render the weapons impractical, despite their capabilities. Conversely, initially it was possible to evolve 'super weapons' would defeat the opponent instantaneously. We are currently addressing this by linking attributes: for example as weapon damage rises, the fire rate falls. Or by introducing new properties that impact on their effectiveness in combat, such as clip size or reload rate. Alternatively, chromosomes could be limited to have a maximum amount of 'points' to spend across the weapon properties. This would ensure the emphasis on properties is distributed fairly and prevent unreasonable content being created.

One distant prospect is to add 'effects' to our content. This is adapting the technique used with *Borderlands*; allowing developers to add unique that wouldn't be possible through attributes. An example of this could be that when a weapon fires a projectile it explodes on impact. These could be purely cosmetic or actually alter the behaviour.

4. CONCLUSIONS

With our framework proving effective, we are continuing to experiment with results garnered and refine those the constraints of the framework. In time, we hope this proves effective in creating a range of content that proves effective in our testing scenario.

5. REFERENCES

[1] Gearbox Software. Borderlands, 2009.

[2] Gearbox Software. Borderlands 2, 2012.

[3] Guiness World Records Ltd. *Guinness World Records 2012: Gamers Edition*. Guinness World Records Limited, London, UK, first edition edition, 2012.

[4] E. J. Hastings, R. K. Guha, and K. O. Stanley. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245, 2009.

[5] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[6] T. Thompson and J. Levine. Scaling-up Behaviours in EvoTanks: Applying Subsumption Principles to Artificial Neural Networks. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 159–166, 2008.

[7] T. Thompson, J. Levine, and G. Hayes. EvoTanks: Co-Evolutionary Development of Game-Playing Agents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 328–333, 2007.

[8] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):172–186, 2011.