An Author-Centric Approach to Procedural Content Generation

Rui Craveirinha[1], Lucas Santos[2] and Licinio Roque[1]

[1] Department of Informatics Engineering
Faculty of Sciences and Technology, University of Coimbra
Coimbra, Portugal
[2] Universidade Estadual da Bahia,
Bahia, Brazil

Abstract. This paper describes an alternative approach for videogame procedural content generation focused on providing authors direct control on what gameplay ensues from the generated content. An architecture is proposed that allows designers to define, beforehand, target gameplay indicators, and then generates content for an existing base-design that achieves those same indicators in actual gameplay sessions with human players. Besides providing a description of this architecture, a trial intent on giving evidence of the approach's feasibility is presented. This experiment used an altered version of 'Infinite Mario Bros.' level generator, built to evolve design parameters so as to achieve 3 target gameplay indicators. Employing a Genetic Algorithm in generation of new parameter values, and using 25 players to test the end results, the platform was able to generate parameters that achieved, with precision, the values for those indicators. This result provides evidence of the approach's feasibility, hinting at its potential use for real-life design processes.

1 Introduction

Founding pillars of videogame theory come from different fields and are mostly general in nature, and while certainly useful for guidance in game design, they tend to lack a strong quantitative basis [Pedersen et al., 2010]. The past years have seen the rise of methods – such as Gameplay Metrics [Kim et al., 2008] – that can better inform creators during the videogame design process, by providing increasingly meaningful and objective data on how videogames are effectively played and experienced. Furthermore, procedural content generation aimed at improving user-experience (e.g. Experience-Driven Procedural Content Generation, EDPCG for short[Yannakakis and Togelius, 2011]) has become a reality, allowing designers access to tools that provide them active support through generation of game content. These advances deliver potential for creative processes that deliver more expressive designs. However, thus far, there are few examples of how these methods can be meaningfully incorporated in design contexts, as ways of facilitating or expanding production of videogame artifacts. What this paper describes then is an alternative procedural content generation approach,

created to help creative game design and production processes. It seeks to utilize automatic generation and evaluation of artifacts to increase both the efficiency and effectiveness of the design process, whilst empowering authorial exploration of the design space and furthering designers' creative agendas for their artifacts.

The remaining paper is thus structured: section 2 gives a brief overview of the main EDPCG literature, and how it was framed in this research; section 3 details this research approach, its architectural details, and how it is intended to use these new technologies, while section 4 details a first trial to implement this solution. Finally, in the last section, results are discussed and future work for this project is described.

2 Literature Review

Experience-Driven Procedural Content Generation is a family of methods that can aid game design, with a growing body of research tackling directly how game experience can be improved through procedurally generated content [Yannakakis and Togelius, 2011]. The general concept is to use a data source to create a player experience model (product of the interaction of players with the game system), and then generate/evolve game content that improves player experience. Experience quality is then assessed based on player experience data, whether it be the result of gameplay metrics, questionnaires, or even AI simulated game-play sessions. Once quality has been assessed, content is represented in computational form, and new game content is generated. Procedurally generated content can encompass level design, art assets, etc. According to [Yannakakis and Togelius, 2011], the rationale of this approach takes into consideration the fact that videogame markets have become heterogeneous, catering to increasingly fragmented segments of the population, relying on different structural aspects to produce similar affective experiences.

As of now, several pioneering efforts have been made with this approach. For example, in [Yannakakis and Hallam, 2004], Yannakakis et al. studied how to measure game interest in predator/prey games (such as Pac-Man), based on the assumption that interest is mostly determined by qualities of computer characters' behavior as opposed to other features (such as the game's graphical properties). To this effect, they proposed and then implemented a neuro-evolution learning algorithm that maximized interest criteria based on gameplay metrics, validating the technique with pair-wise player questionnaires [Yannakakis and Hallam, 2004, 2005, 2007]. In [Pedersen et al., 2010], this methodology was expanded to classify and predict a greater number of experience variables, covering up to 6 emotional categories: fun, challenge, boredom, frustration, predictability, and anxiety. Metrics data from several gameplay sessions was used to track correlations with player experience reports and different Super Mario Bros. levels properties. Their model's accuracy for the six emotion types was very high. Finally, Super Mario Bros' levels were adapted in real time in order to optimize fun, both for human players and AI simulated agents [Shaker et al., 2010].

Several benefits have been studied and verified in relation to user-centered design philosophies, namely in terms of increased system quality thanks to more accurate requirements, heightened production efficiency, improved user acceptance, amongst others [Kujala, 2003]. But we share with other researchers [Steen et al., 2007] reasonable concerns over how extreme user-focused approaches can impair innovation and creativity. Even proponents are careful in how they frame a user-centered approach to design, maintaining a focus on designer's intervening in the process [Kujala, 2003]. Indeed, user-centered design presupposes designers to enter into meaningful dialogical debate with users, in such a way that knowledge is shared from both ends so that design decisions are rightfully informed in a collaborative manner [Steen et al., 2007], never dictated exclusively by users' preferences.

Game designers and producers, like other members of the design field, run the risk of *"reducing experience to the mere "pleasure due to the feel of the action"* [Hassenzahl, 2011], i.e., thinking of user satisfaction as opposed to user experience. For a number of reasons, players may not always like or feel something positive in their experience, despite designers' creative agenda being fulfilled. By focusing procedural tools on, for example, creating frictionless artifacts that can maximize 'fun', researchers seem to be establishing not only the means to an end, but the very end itself of the creative process. Consumers can be superficial, dismissive, indecisive and incapable of adequately expressing their thoughts and emotions [Gladwell, 2005] – and as such to put EDPCG quality measurements dependent only on their subjective evaluation may not be the best approach for creators.

As such, this paper proposes an Author-centric approach to Experience-Driven Procedural Content Generation for games, where the quality evaluation is measured in reference to designer's expectations of what gameplay behaviors should be elicited. Artistic expression has, for the most part, been sustained by authorial pursuit and vision, despite not always having been well accepted by audiences or critics [Gombrich, 2009]. So we propose to put the focus back on authors' formal considerations instead of users' own, while trying to maintain the usefulness of using procedural generation tools. This view seeks to empower designers and award them greater creative control, while permitting more effective ways of tailoring their artifact to mediate specific gameplay patterns according to their agenda.

3 An Author-Centric Approach to Procedural Content Genration

The goal for this alternative is to provide a procedural generation architecture that provides designers, irrespective of their agenda, a more efficient and effective way of getting their game-artifact to mediate an intended gameplay experience. The focus here is on enabling designer choice to lead the end-result for gameplay, as opposed to subjective experience aspects. As such, only the elicitation of gameplay behavior is meant to be improved by this approach, so that player

experience becomes influenced by a dialogical relationship between designer and user, as is the case for other artistic media. In conforming to this view, the data source for the underlying experience model of the architecture is based on Gameplay Metrics, not utilizing any subject inquiries. Gameplay Metrics provide exactly the 'how' of gameplay, and have the advantage of being quantitative, objective, allowing also for large scale automatic data collection [Drachen and Canossa, 2009]. The analogy with other media is that, just as a set of brushes only allows painters to more accurately imprint forms and colors on a canvas, so should game-design tools only help better elicit intended gameplay behaviors to players. This view is in stark contrast with the notions of pre-selecting specific emotions, thoughts and valuations that audiences should take from the actual artifact, and preserves a strict interactionist view where the player, while being influenced by the artifact, retains voluntary translation of its meaning.

At this phase, this approach is only being considered to improve elements of an already existing game-prototype, namely, aspects such as the variables concerning physics or structural level components; though there is no apparent reason why this approach cannot be generalized to generate other aspects of a game's content. So, for designers to use this architecture, they would need to have a reasonably stable working prototype besides their design agenda for the gameplay. The proposed architecture can then support an approach for obtaining and validating improved variations of the original prototype, that elicit the gameplay agenda, as defined by the authors themselves.

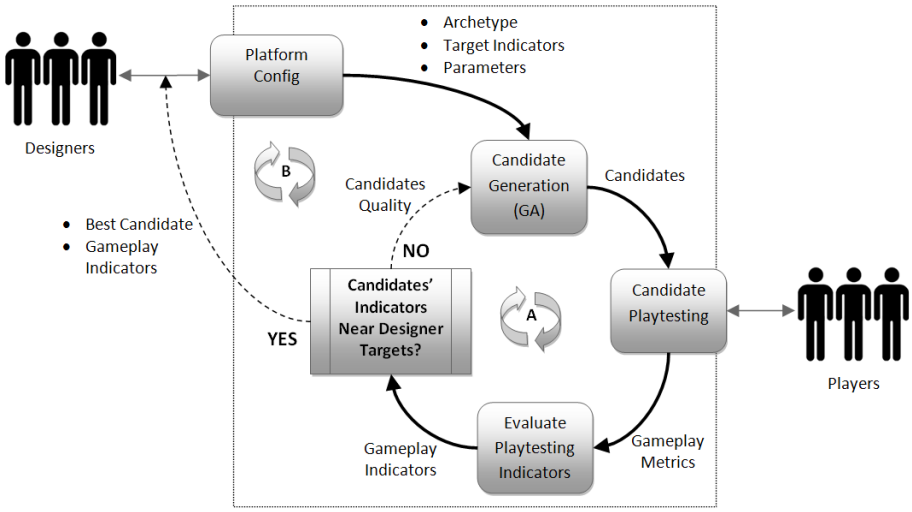

Fig. 1. Diagram of the Proposed Architecture for an Author-Centric approach to Content Generation for viedogames – see section 3 for further details.

In terms of how this process is carried out, figure 1 shows the proposed architecture from a conceptual point of view. To start using the approach, the design team must provide 3 items: a game Archetype, a set of Target Indicators and respective target values, and a definition of Parameters that can be to generate new variations of the Archetype. These 3 elements are that which determines the overall specification of the problem they wish to solve in what regards to their game, and therefore the differentiating aspect of this approach. This is, in essence, the definition of the design domain in which the actual EDPCG optimization method will operate, and the notion of 'Quality' to be applied to the generated content.

Content quality is measured as the inverse of the difference between a set of Target Gameplay Indicator values chosen by Designers, and the values for those Gameplay Indicators that generated artifacts mediate to real players. Parameters are the set of design elements, implemented by designers, that they forfeit direct control of, and which a platform that implements this architecture will then use to generate new configurations. There is the assumption here that authors choose these in a sensible manner, so that they affect the desired target gameplay. The Archetype is the prototype they designed, deprived of the Parameter components, i.e. the base artifact of which all variations will evolve from. In continuing with the designer-centric stance, it is assumed that this Archetype is implemented by designers from the ground up. Naturally, all these elements involve a configuration process carried through a platform user-interface (determining, for example, the types of target indicators, their domain, etc.), and direct integration by means of computational interfaces (methods for publishing and subscribing metrics, for instance).

Once the 3 designer-defined elements are forwarded and integrated into a platform, its actual functioning begins – that is that A-cycle in figure 1. At that point, the platform runs without need of intervention from designers and it is at this stage that actual procedural generation tools are used. First, there is a phase of candidate generation, where new Parameter set values are found by means of a search algorithm; in this case, a Genetic Algorithm (GA) with its selection, recombination and mutation procedures. Secondly, by integrating new Parameters into the Archetype, new candidate artifacts become ready for playing, at which point players play these alternatives, and from their experience with them, metrics are extracted, so as to characterize their gameplay accurately. Thirdly, these are compiled into the Gameplay Indicators that designers defined, and compared with their target values, by means of a distance metric. Finally, if the best candidate's quality is not sufficiently close to the values, then this micro-cycle repeats with a new GA iteration, with each of the chromosomes being awarded a fitness proportional to candidate's quality values. Once the A-cycle ends, a candidate that matches the target indicators, as well as all gameplay indicators data, are given as output to designers, so they can continue their iterative design process, by choosing whether or not the end-result is satisfactory. If it is not, designers can fine-tune either of the elements provided, forwarding a new design specification to the platform, in a new iteration of the B-cycle.

GAs were selected for being a family of sub-optimal, stochastic, general search methods that can be easily adapted into a number of different problems [Haupt and Haupt, 2004], and that versatility suggested they can be fit into the proposed architecture. Note that the choice of the methods' generation algorithm and its technical specificities is not a major focus of this research, since, first and foremost, there is the aim to solve how to frame EDPCG-like methods for creative design use that maintains author choice. So it is implicitly accepted that this process can be carried out by more advanced search algorithms that can better fit the search landscape.

4 Experimental Trial

To put the approach into effect, a design case materialization was needed. The main goal for this research was to test the feasibility of this approach by implementing the architecture into a prototype software platform, and start to map its practical constraints. The idea of this first trial was to take the 'Infinite Mario Bros' level generator (also used in [Pedersen et al., 2009]) and impose a significant variation on the original 'Super Mario Bros.' design. Imagine a hypothetical author that wanted to make a hyped-up version of Mario, with high-speed and high-stakes gameplay, where player experience would be akin to that of doing a speed-run trial. So, this designer changed the core-physics of Mario accordingly: Mario jumps double the height, can shoot more fireballs, and his default state is Super Mario (though he is still turned to normal as he is hurt, as in the original game). Now, these subtle changes in the game's design affect the experience – the game becomes a caricature of the original, both more frantic, exaggerated and difficult to play. The core gameplay system is completed, even though the experience is far from polished; the altered 'Infinite Mario Bros.' base game thus serves as this trial's Archetype. The designer wants users to play as he envisioned: very fast, with a high rhythm of actions, all whilst maintaining it playable enough so that players can conquer levels without much trying.

Now he must define target indicator values for gameplay metrics: Number Tries for conquering a level, as a measure for player difficulty and challenge; number of player triggered Actions per Second, as a measure of the experience's rhythm; and level play-through Time, as a measure of play-speed. He establishes a target mean value for each of these metrics, as well as suitable maximum range value that defines acceptable variation for these metrics; see table 1 for the tested values. That is his experience ideal, materialized into target indicators for gameplay behavior. As a measure of success for this first research trial, it was established beforehand that the system would be deemed feasible if it was shown to be capable of generating a best candidate that, at least, could statistically guarantee indicator values between the target and a maximum of a quarter of the accepted range for a population of players (see table 1).

Eight Parameters with a varied range of design objectives were chosen to be permitted to be altered by the platform: game speed (24–48 cycles per second), number of cannons (0–5), number of goomba enemies (0–30), number of koopa

Indicator	T. Value	Max. Var.	Sucess Range
Actions	2	±4	1–3
Tries	1	±4	0–2
Time	40	±80	20–60

Table 1. The three gameplay indicators that the algorithm meant to optimize, their target value, and the maximum variation that was considered. Also, the minimum condition for success for this trial – to have the best candidate's indicator values nested in the success range.

enemies (0–30), number of holes (0–7), number of item-blocks (0–15), number of coins (0-100) and number of hills (0–5). Apart from speed, all parameters were given a range from 0 to as high a number as possible, keeping in mind that these values had to produce levels that could not break the constraint of the level size of 480 tiles. This is why the number of holes and cannons, structural level properties that occupied several tiles, had to be kept relatively short. This Parameter set encompasses one physics parameter – Game Speed – which radically alters game experience, as well as impacting game difficulty (as a faster game becomes harder to play). This parameter was allowed to fluctuate between the games' normal speed of 24 cycles per second (which resembles Super Mario's original speed) to double that, 48 cycles, which is so fast it borders th nigh unplayable. The speed variable is bound to affect all 3 of the chosen target indicators, as it makes both faster and more difficult to play. Four parameters were chosen that determine adversarial forces to hinder player progress (cannons, goombas, koopas and holes). These are expected to affect all three indicators, as more enemies and holes impact difficulty, as well as delay player traversal of the level, while increasing opportunities for actions (jump, stomp and shoot). Two parameters were chosen that allow explorative rewards, coins and item power-ups. Coins only affect rhythm and traversal time, as collecting them increases the number of actions, while taking time. Coins do not have impact on difficulty, as they do not improve chances of conquering a level, and because in this prototype the player is given 999 lives to play the game, there is no tangible benefit to collecting 100 of them so as to gain lives. Item-power ups should provide the reverse set of indicator impact: mushrooms and flowers ease player difficulty in clearing a level, while only marginally affecting action rhythm and traversal speed (as they are few and disperse in a level). Finally, the number of hills parameter was expected to be quasi-neutral to all indicators, as it is a structural feat that provides a mostly cosmetic effect (hills are just platforms where player and enemies can go to). It is possible that it could slight increase the number of actions (as it provides a jump affordance) or that it could hinder progress (as it could present a slightly more complex level structure for players to overcome).

4.1 Genetic Algorithm Implementation

Parameter set candidates are evolved by means of a simple Genetic Algorithm with the following configuration. Each individual is represented by a chromosome that is a vector of real values, each a gene corresponding to a given game design parameter which can be varied. For each generation, the population size was of 12 individuals, plus one extra slot reserved for elitist maintenance of the best individual. Uniform crossover was used as a recombination method, with a rate of 90%. Mutation operated by replacing a gene to a new random value, and had a starting probability of 5%, and that was increased 0.5% in each generation (these seem large values, but one must consider that for this test, there were only 8 genes and 12 individuals, so lower values would practically exclude mutation from ocurring). Selection was made by tournament for pairs of chromosome. Evaluation of each candidate's fitness is as follows: in each evaluation phase, all the chromosome population is converted into full sets of design specifications that are stored in the server. Players use a client application which automatically downloads the next available set, which users then play and experience, after which the metrics data from their play session is uploaded to the server. It is from the metrics data that fitness is directly calculated, using the formula 1:

$$fitness = \sum_{i}^{N} \frac{1}{N} q(i) \tag{1}$$

$$q(i) = \begin{cases} (1 - \frac{\left|tgt(i) - avg(i)\right|}{max(i)}), & \left|tgt(i) - avg(i)\right| \leq max(i) \\ 0, & \left|tgt(i) - avg(i)\right| > max(i) \end{cases} \tag{2}$$

where N is the total number of indicators, $tgt(i)$ the target value for a given indicator i, $avg(i)$ the average value for indicator i measured from the available sample of gameplay sessions, and $max(i)$ is a value determined by designers on what the maximum variation they are accepting in terms of the end sample, before the usefulness of the candidate becomes 0.

The evaluation phase goes on until every population member has had at least 5 evaluations, i.e. 5 different play-throughs of the same level. Once this condition is fulfilled, the GA will run its evolutionary mechanisms, and a new generation will be evaluated. In between, if at any moment the best chromosome produces a candidate whose fitness is above 0.9, an external validation phase is carried out, where it is played at least 30 times, to validate its fitness level with greater levels of credibility. Only after this process does the GA resume. Candidates in the form of game parameter sets are forwarded to client-applications sequentially, which means that once a player plays, he will likely experience a given generations' entire population in sequence. However, the system is connected asynchronously, with no control over who, how and when experiments a given set, so play-testers were free to access the game at their own leisure, throughout the one week span of the test.

Generation	0	1	2	3	4	5	6	7*
Sessions	76	176	138	107	105	106	184	26*

Table 2. The table shows the total number gameplay sessions evaluated per generation. Note that generation 7 was incomplete, and as such its data will not be further contemplated (with the exception of new evaluations of the best individual that happened in that final stage).

4.2 Population Sample

The playtester group was comprised of 25 Informatics Engineering and Multimedia and Design Master students, 21 male, 4 female, ages 22–26. They were asked to play the game as means of collecting data for research purposes. No incentives were given, and no questionnaires were collected from the population. There were a total of 696 gameplay sessions played on subjects' own time and convenience, giving 27.84 gameplay sessions per player on average. Data from these sessions was used for 6 GA generations, plus a few more evaluations that occurred afterwards that were not enough to complete a generation's cycle. In total, 68 different procedurally generated candidates were tested until the sixth generation, in an equal number of fitness calculations. This means an average of 9.85 gameplay sessions played per generated candidate ($stdev = 5.52$), distributed unevenly throughout generations and playtesters (see table 2).

4.3 Results

As can be seen from figure 2, the GA was able to improve solutions relatively steadily until the final generation, assuring us of its overall convergence. Both the population's average fitness, as well as the best individual's fitness increased moderately until the final generation. Fitness variance in the population shows a tendency for decreasing, though it increases slightly from 4th generation onwards (presumably from the increase in mutation rate, designed precisely to avoid stagnation). While the best results seemed close to ideal, several things are in need of improvement. For one, in this test, the initial setup seems overly positive, with average fitness levels already in 0.8 levels (on a 0–1 scale). To a certain extent this is to be expected, given that the platform's aim is to iterate on existing designs, and not necessarily to generate an entirely new design from a completely broken one. That being said, the improvement that was measured, while substantial in terms of player experience, did not allow a strong validation of this method's power to achieve the desired target indicators.

Also to take into account, when interpreting the graph, is that the fitness landscape is, in essence, dynamic, as it depends on players behavioral patterns during game sessions. This happens, for instance, when a chromosome passes from one generation to another, and is then evaluated again, which can cause significant fluctuations to its fitness level. One particular instance of this dynamic component affecting fitness levels, is easily perceived in the second generation,

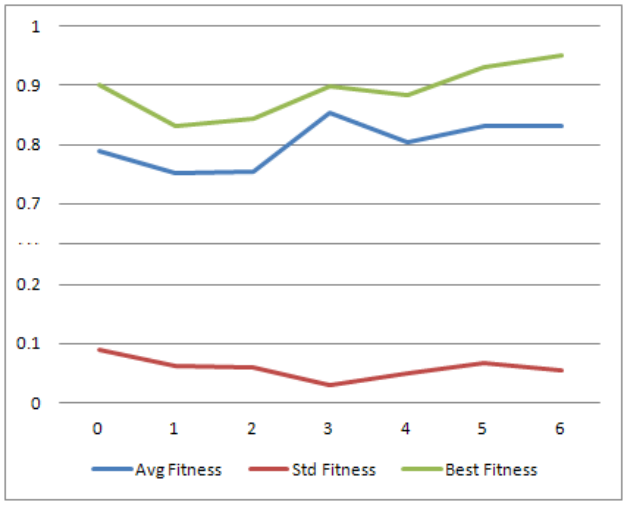

Fig. 2. Evolution of average population fitness, its standard deviation and the fitness of the best individual in the population. Note there is folding in the y-axis.

where a higher number of gameplay sessions seems to have led to a drift in values for all individuals, with a significant decrease in both average and best fitness (see chart 2 and table 2 side by side). This means that part of that initially excessive positive assessment of the population was likely to be due to statistical misrepresentation of the sample. Even so, overall, the test presented a good trend for improvement of solutions, though from these results, it seems it presented a low difficulty problem.

Figure 3 shows how parameters varied, on average, across generations. Given the tendency for GA to improve the average fitness of the population (and therefore converge with target metrics), the major tendencies for parameters variation mirror an improvement of the candidates. The seemingly more meaningful trends in these parameter variations are: a steady value for the number of hills, a slow decrease in cannons, holes, goombas and game speed, a subtle increase in item blocks, and an initial sharp decrease in number of coins followed by sharp increase. The lack of variation in number of hills is unsurprising, given how little they impact gameplay other then providing players with a few extra platforms for jumping. These could have improved the number of actions, but in this test the effect proved negligible. The decrease in cannons, holes, goombas and game speed, hand in hand with the small increment to item blocks, can be seen as the algorithm seeking to decrease the game's difficulty and its action rhythm, both of which were initially higher than target levels (this trend is visible when comparing this graph with figure 4 and target gameplay indicators absolute values). The initial decrease of coins followed by increase around generation 3, are probably the result of trying to adjust the number of actions, as these are above target average before generation 3, and below afterwards.

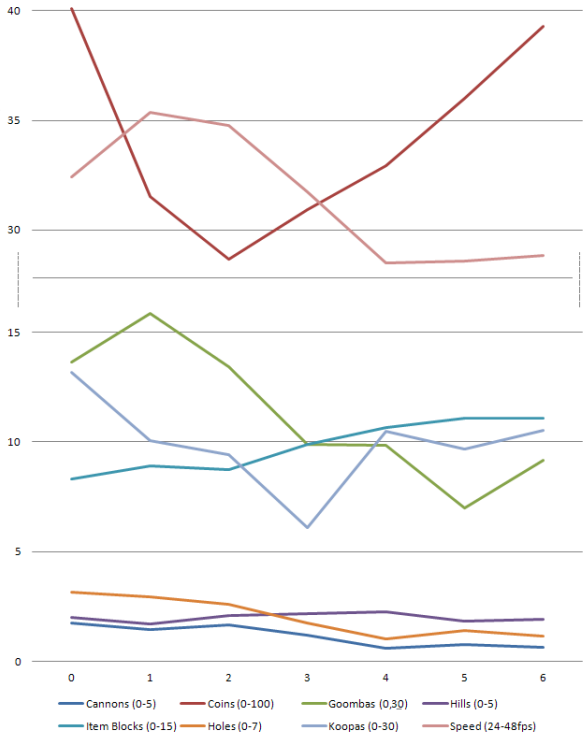

Fig. 3. Evolution of game parameters throughout generations. Note that there is a folding of the y-axis, between 15–30, for ease of visualization.

Looking at each of the quality vectors in the fitness variable in figure 4, one can discriminate how the algorithm fared at improving each of them. While 'Actions' and 'Tries' seem to follow a somewhat stable convergence curve, tentatively improving on average, 'Traversal Time' seems to oscillate with no clear tendency to improve on a population basis. The reason for this result only becomes clear once the relationship between parameters and indicators is analyzed further, per table 3.

The correlation table establishes clearly why the "Time to Traverse Level" indicator could not be radically improved across generations – none of the chosen game parameters seem to have a statistically significant relation with its variation. This indicates that the parameter set was insufficient for an adequate optimization of that particular indicator. This failure of one of the indicators to converge in terms of the overall population, hints at the dangers of a poor choice of 'Parameters' in finding an adequate solution, as well as the need for subsequent iterations of the cycle, by means of designer intervention (as introduced in the architecture). As to the other two indicators, it is clear that the set of 'Parameters' was sufficiently capable of producing some degree of optimization.

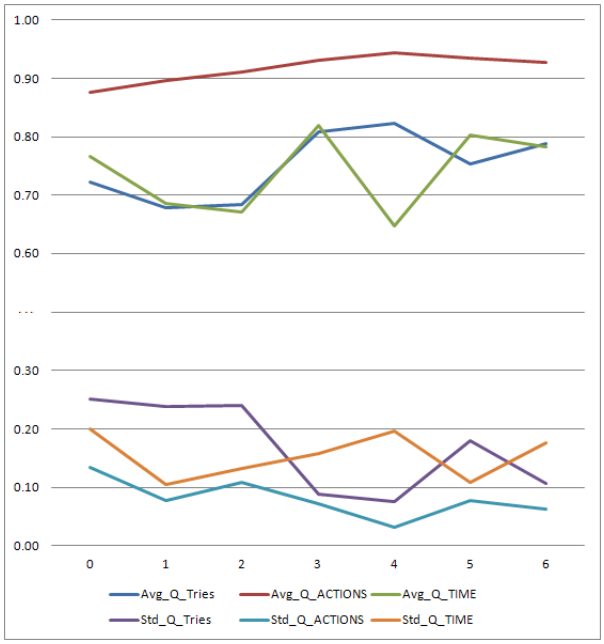

Fig. 4. Evolution of the three quality components, each for one of the indicators ('Tries', 'Actions' and 'Time') used to calculate the fitness function. There is folding in the y-axis between 0.3 and 0.6.

All parameters with exception of number of 'Koopas' impacted the 'Actions Per Second' in a statistically significant manner, though only three could explain more than 0.3 of its variability: Goombas, Coins and Game Speed, three parameters that we previously saw were probably varied by the GA to improve the number of actions. Number of 'Holes' and 'Cannons' affect negatively the action rhythm, though this is most likely only due to the increase in difficulty – more challenge means a more cautious gameplay style which then means slower actions' rhythm. As for the 'Tries' indicator, difficulty seems to have been mostly dictated by number of Holes, and on a smaller level, by Game Speed and number of Cannons, and as expected, counter-weighed by number of item-blocks. Unexpectedly, there is a weak, barely significant relationship between more coins and less 'Tries', though it eludes us how more coins could improve survivability in a play-through.

4.4 Best Solution

The best Individual found by the GA had a final fitness value of 0.96, calculated out of a total of 15 evaluations. Level retries: $m = 1.13, std = 1.85$; actions per second: $m = 2.14, std = 0.63$; time to complete level: $m = 45.26, std = 15.83$. Its parameter set was: 0 Cannons, 30 Coins, 15 Goombas, 1 Hill, 4 item blocks, 2

Correlations								
		Cannons (0-5)	Coins (0-100)	Hills (0-5)	ItemBlocks (0-15)	Holes (0-7)	Koopas (0-30)	Speed (24-48fps)
Actions per Second	Pearson Correlation	**-.191**	**.319**	**-.164**		**-.172**	.059	**.333**
	Sig. (2-tailed)	.000	.000	.000	.001	.000	.117	.000
Time Traverse Level	Pearson Correlation	-.015	-.019	-.001	-.015	.049	-.016	-.077
	Sig. (2-tailed)	.694	.613	.988	.693	.195	.675	.042
Tries	Pearson Correlation	**.257**	**-.102**	-.071	**-.199**	**.420**	.043	**.197**
	Sig. (2-tailed)	.000	.007	.058	.000	.000	.254	.000

Table 3. Table showing statistical correlation between parameters and target indicators. Significant values are highlighted in red and greater font size means greater variance prediction.

Holes, 9 Koopas and speed of 28 cycles per second. However, despite the closeness of indicator results to their intended values, the distributions are uneven, thus failing to provide a strong validation for the parameter set being able to generalize this behavioral pattern. To study the extent of the generalization possibilities for this candidate, one must turn to statistical indicators. The standard error for skewness in a population of this small size is 0.580 and kurtosis is 1.121, so any skewness and kurtosis that is greater than the double of these values is indicative of a non-normal distribution. The 'Number of Tries' distribution of the best candidate has skewness $+1.739$ and kurtosis 2.431, therefore being both skewed and leptokurtic, i.e. significantly non-normal. Skewness and kurtosis values for Actions Per Second were -0.679 and -0.844, and for Time To Traverse Level were -0.697 and 1.791 respectively, therefore not significantly non-normal in both these aspects. Consequently, now it is possible to use the Shapiro-Wilk test of normality for both distributions, verifying that it is not possible to reject their normality ($sig = 0.203$ and $sig = 0.478$ respectively). Given these facts, it can be asserted to be somewhat likely that these two indicators are samples of a normal distribution. With this in mind, and assuming the playtester sample to be representative, it is possible to calculate how close this game was to achieving the target metrics for the population at large. The confidence interval, at 95% confidence levels, for the"Actions Per Second" distribution is $[1.79, 2.48]$; for "Time To Traverse Level", $[36.50, 54.03]$, at 14 degrees of freedom. Both these ranges are well within the conditions of success for this trial, and while it is not possible to extend this calculation for the 'Tries' distribution, at least its average did not veer off course (see table 1 for success ranges). Admittedly, the pre-established values for success turned out to be underwhelming considering the results, in line with the higher than expected fitness levels for candidates. To conclude, of the three target indicators, the best candidate achieved appropriate mean levels for all, though only 2 of these produced sound distributions that hint at gen-

eralization of this success. Only these 2 can, with an adequate confidence level, be seen as statistically trust-worthy representatives of the candidate's indicators imprint. So, this method was a success for the two indicators that the Parameters were capable of provoking variations (as statistical correlations attested to).

5 Conclusions and Future Work

Given the obtained results, it is with confidence one can conclude that in this instance, it was possible to develop and implement a platform that subscribes to the proposed Author-centric approach to EDPCG. It proved capable of achieving sub-optimal values for gameplay indicators, though only as long as the design space was well defined (meaning, when Parameters were shown to be capable of significantly affecting said indicators). This entails that the experimental trial achieved its goal: to find sufficient evidence to sustain the feasibility of this approach. Moreover, this points towards the relevance of the approach herein proposed, and specifically, towards the importance of incorporating forms of design-space specification by designers in EDPCG methods.

In terms of the implementation details, results showed that some elements are in need of revising. The simple GA that was used for generation purposes did achieve moderate improvement to the game instances, though the fact remained that convergence was achieved from an exceedingly positive starting point. Further, both the high variability and dynamic nature of the fitness landscape impacted significantly its optimization process. To solve this issue, in the future the minimum number of gameplay sessions to evaluate candidates will need to increase (therefore decreasing the likelihood of more drastic fitness variations), and the selection process will be adapted to account for the varying nature of the landscape. Coming trials must also see an improvement to the definition of target gameplay indicators that provides designers with some form of control over how much variability they intend to have in their candidates, as a measure of quality introduced in the algorithm.

If future trials repeatedly show the approach's capacity to find parameter sets that can mediate specific gameplay metrics imprints to a playtester population, it is our opinion that this will be a huge advantage for the videogame production process. This test however, only showed the method's plausibility. Coming experiments shall study how to better interface this approach with designers while also providing further evidence of the method's capability to generate good candidates, and the constraints in which such is possible. Only once its efficacy, specifications and interface are fully studied can this approach be considered validated.

6 Acknowledgements

This work was financed under PhD Scholarship number SFRH/BD/75196/2010, awarded by the Portuguese FCT - Fundacao de Ciencias e Tecnologia and Project ICIS with reference CENTRO-07-0224-FEDER-002003.

Bibliography

Anders Drachen and Alessandro Canossa. Towards gameplay analysis via gameplay metrics. In *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, MindTrek '09, pages 202–209, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-633-5. doi: http://doi.acm.org/10.1145/1621841.1621878. URL http://doi.acm.org/10.1145/1621841.1621878.

M. Gladwell. *Blink: The Power Of Thinking Without Thinking*. Little, Brown and Company, 2005.

E.H. Gombrich. *The story of art*. Phaidon, 2009.

Marc Hassenzahl. User experience and experience design. In Mads Soegaard and Rikke Friis Dam, editors, *Encyclopedia of Human-Computer Interaction*. Interaction-Design.org, 2011.

Randy L. Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. John Wiley & Sons, Inc., second edition, 2004.

Jun H. Kim, Daniel V. Gunn, Eric Schuh, Bruce Phillips, Randy J. Pagulayan, and Dennis R. Wixon. Tracking real-time user experience (true): a comprehensive instrumentation solution for complex systems. In *Computer Human Interaction*, pages 443–452, 2008. doi: 10.1145/1357054.1357126.

Sari Kujala. User involvement: A review of the benefits and challenges. *Behaviour & Information Technology*, 22(1):1–16, 2003. URL http://dx.doi.org/10.1080/01449290301782.

C. Pedersen, J. Togelius, and G.N. Yannakakis. Modeling player experience for content creation. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):54 –67, march 2010. ISSN 1943-068X. doi: 10.1109/TCIAIG.2010.2043950.

Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling player experience in super mario bros. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, CIG'09, pages 132–139, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4814-2. URL http://portal.acm.org/citation.cfm?id=1719293.1719323.

N. Shaker, G. N. Yannakakis, and J. Togelius. Towards Automatic Personalized Content Generation for Platform Games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, October 2010.

Marc Steen, Lottie Kuijt-Evers, and Jente Klok. Early user involvement in research and design projects – A review of methods and practices. In *23rd EGOS Colloquium (European Group for Organizational Studies)*, 2007.

Georgios Yannakakis and John Hallam. A generic approach for generating interesting interactive pac-man opponents. In *In Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 94–101, 2005.

Georgios N. Yannakakis and John Hallam. Evolving opponents for interesting interactive computer games. In *Simulation of Adaptive Behavior*, 2004.

Georgios N. Yannakakis and John Hallam. Towards optimizing entertainment in computer games. *Appl. Artif. Intell.*, 21:933–971, November 2007. ISSN 0883-9514. doi: 10.1080/08839510701527580. URL http://portal.acm.org/citation.cfm?id=1392651.1392653.

Georgios N. Yannakakis and Julian Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 99(PrePrints), 2011. ISSN 1949-3045. doi: http://doi.ieeecomputersociety.org/10.1109/T-AFFC.2011.6.