Procedurally Generated History: building a game ecosystem through autoplay

Gabriele Trovato
Waseda University
#41-3-04/05, 17 Kikui-cho, Shinjuku-ku,
Tokyo-to, 162-0044, Japan
+81-3-3203-4394
gabriele@takanishi.mech.waseda.ac.jp

Soren Johnson
Mohawk Games
1966 Greenspring Dr, Suite 105
Timonium, MD 21093, USA
+1 443-564-9814
soren.johnson@gmail.com

Pieter Spronck
Tilburg University
Dante Building, Warandelaan 2
5037 AB Tilburg, Netherlands
+31 13 466 2695
p.spronck@uvt.nl

ABSTRACT
Generating interesting game worlds is an ongoing challenge within Procedural Content Generation (PCG). While the use of algorithms for automatically generating content could potentially bring many advantages, the content itself is often random and not well-crafted. In this paper we explore a potential solution to this issue by creating meaning through procedurally tracing a "history" of the game world while constructing it. We introduce a novel approach, called Procedurally Generated History (PGH), which consists of building a game world history through controlled autoplay. The resulting game ecosystem is expected to be engaging and balanced as hand-made content, while still preserving replayability. PGH was implemented in a modpack for *Sid Meier's Civilization IV*. Tests with the modpack demonstrated the soundness and balance of the results of the PGH approach.

Keywords
Procedural Content Generation, game design, strategy games

1. INTRODUCTION
The ability to explore a new environment is one of the main attractions to players of open-world games, such as RPGs and strategy games. Game designers can either generate a world automatically (which happens in, for instance, "rogue-likes"), or craft it by hand. Crafted worlds are usually interesting, but there is only one world for the player to explore. Generated worlds are usually much less interesting because there tends to be little meaning behind the world's features.

A world that feels interesting to the player has to be well-structured. One way of imposing structure on a world is to make sure that there is a reasonable history behind the world's construction. Our goal in this paper is to create a system that procedurally traces a history to end up with a well-structured, automatically generated world. The term *history* is used in a broad sense, as a series of events which contribute to the making of the "ecosystem" in which the human player enters the game. It may encompass, for instance, the Earth's history, or the background stories of a fantasy world. Through the generation of a diverging history, the game experience will be different too.

Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015), June 22-25, 2015, Pacific Grove, CA, USA.
ISBN 978-0-9913982-4-9.

1.1 Related work
Programmatic generation of game content using a process that results in a range of possible game play spaces is known as Procedural Content Generation (PCG) [3]. PCG is useful for creating game content such as levels, maps, quests, textures, characters, vegetation, and even rules, allowing game designers to save time and produce content that because of the randomness, gains replayability, without changing the core of the game.

PCG has been attempted since the 1980s, and it is argued that the use of PCG may lead in the near future to a rethinking of the level designer's job [3]; however, it is still a challenging topic nowadays. In almost all commercial games that feature large open worlds, the environments are mostly designed manually, as it is recognized that procedurally generated environments tend to be less interesting than manually designed ones.

Togelius et al [10] suggest three main goals and nine challenges that should be achieved in PCG. Among those challenges, the search space construction is particularly related to the concept that we are presenting in this paper.

Several types of PCG methods exist, most of which are search-based [9]. They can be categorized as: online or offline generation; the extent of generated content; and the extent of how an algorithm can be parameterized [9]. Most methods focus on generating small maps or levels, for which the main requirements are high variety and the satisfaction of some spatial constraints.

A challenge is to make any generated content "interesting" or "meaningful". A typical solution to this challenge is to pre-define the "required content" of the world, and only generate the connections in between [1,7,9].

1.2 Procedurally Generated History
In this paper we describe a novel approach to the generation of meaningful game worlds, which we call *Procedurally Generated History* (PGH). The idea behind the approach is that the game content is generated by initializing the game world by populating an undeveloped landscape with small initial factions, then let the world evolve by having the factions use autoplay (self-play) to grow and develop, under supervision of a management system. The approach is particularly appropriate for strategy games, but can be applied to any game in which there is a map to explore,

and in which there are a multiple players pursuing final victory. It is reminiscent of the world generation system in the indie game *Dwarf Fortress* (Bay 12 Games, under development since 2002). Here, and in the rest of the paper, we use the term "player" for any AI or human player who participates in the game as a faction. If we refer to human or AI players specifically, it will be clarified.

This paper is constructed as follows. We explain the concepts behind PGH in Section 2. The concepts were implemented in a mod for the commercial turn-based strategy game *Sid Meier's Civilization IV*, which we discuss in Section 3. We performed several tests of the mod, which are described in Section 4. Section 5 evaluates and discusses the results. Section 6 concludes and outlines future work.

2. CONCEPTS

Many strategy games offer two different modes of play: the standard game, which is based on automatically generated maps, with random variables, and the fixed scenario game (sometimes as a campaign). Scenarios feature the game map in a particular state, which is typically drawn manually by a scenario designer.

The standard game has high replayability, although its randomness can lead to boring playthroughs, as non-controlled variables may produce game situations that are less engaging. Conversely, the fixed scenario game provides a more balanced and engaging play experience, at the cost of diminished replay value: once the scenario has been beaten, there is little point in playing again, except for challenging one's own best score or for trying different strategies.

Our PGH approach aims to gain the advantages of both described modes of play (see Table 1). To go from the standard game to the PGH game, one needs to guide the random game generation through the addition of constraints (left to right in Table 1). To go from a fixed scenario to a PGH game, one needs to add flexibility through loosening constraints (right to left in Table 1). The synthesis between the two approaches is the main idea behind PGH: fix the constraints and let the game produce interesting content by itself, which can be accurate, balanced and engaging like a scenario, but is different in every instance.

PGH involves several aspects of game content: the main points mentioned in Table 1 are explained in the following subsections. Each one of these concepts can be implemented differently depending on the specific game. Section 3 offers our implementation for *Sid Meier's Civilization IV*.

2.1 Autoplay

In this paper, we are considering a subtype of videogames, which feature a map and a number of factions which pursue victory. In such games, usually one faction can be controlled by the human player, while the other factions are controlled by AI. This is typical for most strategy games (both RTS and turn-based). In such a setup, it is possible to replace the human player with an AI and let the game autoplay, i.e., let AI control all the factions in the game, and let the game progress as if a human is actually interacting with it, until a specific amount of (in-game) time has elapsed. Many game developers use such an approach for evaluation purposes: in PGH, we intend to use it as part of generation of the content.

Table 1. Differences between a standard game with random map, a fixed scenario, and our PGH approach

Standard game	PGH	Fixed scenario
Fixed starting time at game start	Autoplay until starting time	Fixed starting time
Standard number of players	Dynamically changing number of players	Scenario-specific number of players
Players strength depending on standard difficulty levels	Players strength controlled to prevent unbalance	Players strength depending on their role in the scenario
AIs with standard behaviour	Customized AIs which pursue sub-goals	AIs pursue fixed scenario goals
No scripted events	Events as a way to control autoplay	Scripted events
Random map	Either random or hand-drawn, map can be modified at runtime	Hand-drawn map
Results in: replayable game, no history	Results in: realistic and replayable history	Results in: realistic history, little replayability

Autoplay has a central role in PGH: the game plays itself until a certain time at which the human player enters the game. In every instance, the generated ecosystem will be different, possibly in unexpected ways, but always within the desired constraints.

For this purpose, autoplay alone is not enough; it must be guided by an integrated management system. During autoplay, PGH retains control over the world that is generated by managing the number of players (2.2), balancing player strength (2.3), ensuring personalization of AI (2.4), and addressing the course of history through events and world changes (2.5).

2.2 Management of complexity

A generated ecosystem should be self-sustainable, i.e., must ensure that its complexity does not grow beyond acceptable bounds. This is especially true in case of a large number of AI players. For example, in strategy games, each faction produces units, buildings, and other infrastructures that develop with time. Factions tend to grow, and as such the complexity of the game grows. As complexity grows, loading times may increase exponentially. Thus, PGH should ensure that complexity remains low enough that the system will be able to produce good results within a reasonable amount of time.

The number and capabilities of AI players should be managed dynamically, as while loading times may be reasonable at the start of the game, the world is likely to clog up in later stages. The extension of some AI players may grow out of control. The most important function of the PGH management system is to take care of keeping the complexity of the world as a whole in reigns.

The generic concept is based on parametric functions and on a dynamic system of rewards and penalties, as shown in Figure 1, which is explained as follows. Calculations are performed continuously or periodically, measuring the performance of players via some parameters. These performance criteria are not specifically measuring how strong a player is, but rather determining if the player is a candidate to be excluded from the

game or to be merged with another player. Ratings are produced as output: the player is made aware of such ratings and may take actions consequently, which will be used in the next set of calculations. When one player is defeated, other players may benefit from the change. Conversely, a new player may be added when number of players gets too low compared to the optimal number, which is set at the start of the PGH process.

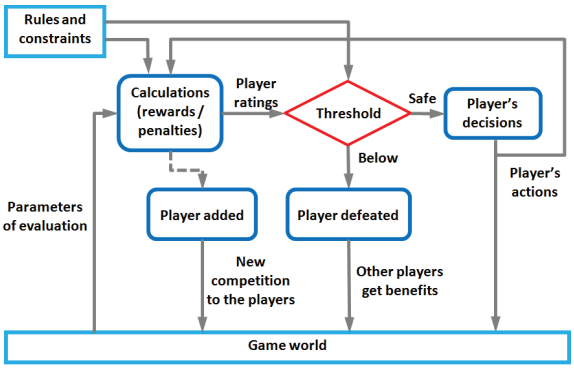

Figure 1. System for dynamic management of complexity.

2.3 Players strength balance

A world that develops on its own will produce a different history in every game. Because of this, the degree of challenge might vary considerably for a human player from game to game. During the autoplay process, the game must be rebalanced whenever some AI players are becoming too powerful or too weak. The calculations used for the management of complexity (Section 2.2) are useful for this purpose.

The performance of players within this management system indicates how suitable they are for inclusion in an "interesting game", and is different from their actual strength in the game. Conversely, the system will evaluate "being too strong" as a negative performance. As a result a strong player will have to face increasing challenges to manage its own strength. This does not mean punishing a player for being strong, but rather letting such a player face increasing difficulty in managing its achievements. Three cases can happen when a player gets increased challenges:

1. The player fails and gets defeated, and is replaced by a new player (see Section 2.2). The new player has to be given some advantage to catch up with the rest of the players and be at least "average". To achieve that, the new player can be constructed using a crossover operator that uses existing players as "parents".
2. The player struggles to manage the challenge and gets weaker, thereby "mutating" into a player that may remain in the game.
3. The player succeeds in mastering the challenge and stays in the game, which may lead to the player having to face new increased challenges.

In Section 3.3 an example of implementation of this approach will be discussed.

2.4 Personalization and roleplaying

A credible, sound history which can be recognized from a generated world triggers suspension of disbelief in the human player. As in the PGH approach the world's history is created by autoplay, AI players must base their decisions on "roleplaying" rather than "choosing effective actions".

Generally speaking, depending on the game, AI can be designed as a simple replacement of a human player, striving to achieve some goals as effectively as possible, or be designed as an entity that contributes to the human player's experience by exhibiting a "personality" [4]. The two roles are not necessarily a dichotomy. Through the personalization of the AI for each player, and the design of specific sub-goals that match the personalization, it is possible to make AIs pursue goals while at the same time appearing to be roleplaying. PGH provides AI players with personalization, and goals that match their personalization, thereby forcing the AI to "roleplay".

2.5 Addressing the course of history

In order to help the history develop in a certain direction, specific events may need to occur. A "non-invasive" way to address historic needs is to add events that are triggered under certain conditions. Events are also useful for adding variation, or for balance purposes. For example, a plague that decimates a large army of soldiers or a fluctuation of prices due to an economic crisis, are events that can manipulate history in a certain direction.

In some cases it may be necessary to modify the world in order to accommodate events. If such changes are made while the human player is already interacting with the game, they should go unnoticed (for example, changes could be done in an area covered by fog of war). This way of doing modifications resembles a "Wizard-of-Oz" technique in robotics [6]. A Wizard-of-Oz experiment is one in which robots are used as puppets to overcome their lack of autonomy, while subjects interact with them believing they are really autonomous. Same as in robotics, the designer's hand should not be revealed to avoid players feeling cheated.

3. IMPLEMENTATION

The PGH approach was implemented as two mods for *Sid Meier's Civilization IV*. We describe their different features below.

3.1 Mods for *Sid Meier's Civilization IV*

Sid Meier's Civilization IV is a turn-based strategy game in which players develop a cilization, of which the final goal is to dominate the world through some victory conditions. In a standard game mode, all civilizations start at in-game time 4000 BC, and the game unfolds through the ages. Maps are generated randomly. Next to the standard mode, the game also features static scenarios on a small portion of the timeline, with players' cities and units already placed.

The two mods we discuss here are *Rhye's and Fall of Civilization* (in short RFC), which plays on a map of Earth, and its random-map version RFC RAND. These mods, developed by the first author, feature all the concepts mentioned in Section 2, and are implemented as shown in Table 2. The first column in Table 2 corresponds to the central column of Table 1. Each of the features will be discussed in the following paragraphs.

Note that in these mods, the PGH approach will continue to regulate the environment even after the human player has entered the game, to ensure that the history of the world not only feels "natural" upon the moment of the human's entrance, but also afterwards.

Table 2. Implementation of the main PGH features in modpacks RFC / RFC RAND

PGH	RFC / RFC RAND
Autoplay until starting time	Autoplay for generating a late start for each civilization
Dynamically changing number of players	Number of players changing through the stability system
Players strength controlled to prevent unbalance	Players strength controlled through the stability system
Customized AIs which pursue sub-goals	Modifiers for AI expansion and differentiated victory conditions
Events as a way to control autoplay	Plague; World Congresses; Conquistadors; Great Depression
Either random or hand-drawn, map can be modified at runtime	Starting locations hidden by fog of war are cleaned up

3.2 Autoplay

In the standard version of *Sid Meier's Civilization IV*, all the players start at the same time in the Ancient Age, 4000BC. In RFC, each civilization has a different starting year. The human player, choosing a civilization, will start playing in a specific year in which the civilization is born: around 753 BC in case the Romans, in the Middle Ages in case of European nations, etc. Until the year of entry, the game is in autoplay mode.

3.3 Stability system

The core of the PGH management system introduced in Section 2 is implemented in RFC as a "Stability System". This system is integrated in the game, running in the background using measured data, and also provides feedback to the players, enhancing their play experience. The key concept is to determine what can make a civilization "stable" or "on the verge of collapse". This a process that is a persistent feature of world history: complex societies have to increase their level of complexity in order to survive new challenges, while the returns on their investments reduces with time [8], a mechanic that mirrors the players' strength balance explained in Section 2.3. The calculation of each civilization's stability is proportional to several factors regarding expansion, government, economy, city management and foreign relations.

The generic formula (Equation 1) is calculated every game turn T:

$$S_{TOT}^{(T)} = S_b + S_p = \sum_i b_i^{(T)} w_i + S_p^{(T-1)} \qquad (1)$$

S_{TOT} is the total stability, sum of a base stability S_b and of the permanent modifier S_p. Base stability is the sum of all the weights w_i associated to the base stability factors b_i. These factors are information about the civilization taken directly from the game. Examples include: number of cities, government type, population, religion, percentage of unhappy citizens, and GDP. Each b_i is multiplied by its weight w_i, which normalizes the value, resulting in a positive or negative term. The sum of the terms is calculated every turn, as in Table 3 ("New base stability"). Civilizations affect each other: having an unstable neighbour is a factor that brings instability. Interaction between the factors is also possible: wars may have a negative impact on civilizations, but can be positive for civilizations under a totalitarian regime.

While S_b is recalculated every game turn, the permanent modifier S_p is updated when certain events related to stability are triggered. Equation 2 expresses such an adaptation to the permanent modifier at turn T_e when event e occurs. Examples of events are: war declared, city razed, peace signed, and building constructed.

$$S_p^{(Te)} = S_p^{(Te-1)} + w_e \qquad (2)$$

An example is given in Table 3. During a game, at turn 200, one civilization's modifiers S_b and S_p are respectively 23 and 12. The civilization founds a city during turn 200, triggering a permanent bonus (+3). However, during the next calculations of the base stability, the empire size turns out to be quite large, and among other factors b_i, the one related to the number of cities adds a malus (-4). At turn 202, the new city is under attack: enemy units within the borders are regarded as a negative factor (-2) for base stability. The following military victory triggers a +1 permanent bonus, and subsequently enemy units retreat: at turn 203, base stability is back to 19.

Table 3. Example of calculation of stability

Turn	Calculation	S_b	S_p	S_{TOT}
200	New base stability	23	12	35
200	City founded (+3)	23	15	38
201	New base stability	19	15	34
202	New base stability	17	15	32
202	Military victory (+1)	17	16	33
203	New base stability	19	16	35

Six levels of stability are assigned depending on S_{TOT}: [very solid / solid / stable / shaky / unstable / collapsing]. Periodically, one of the civilizations with a value of S_{TOT} in the *unstable* or *collapsing* zone will see a part of their territory split up, or even the whole empire descend into civil war and materially leave the game. This may happen to any civilization regardless of their strength: for example, a strong empire, facing a temporary period of economic depression may cause the stability to be ranked as low. Controlling a rival civilization's motherland may also cause trouble. As explained in Section 2.3, preventing a player of becoming too strong is part of this mechanic. Weak civilizations are also likely to end up in unstable zone by losing wars or having poor base factors b_i. These civilizations will be at risk of breaking up or being absorbed by neighbouring civilizations.

The whole system, in symbiosis with the game world, will make the game develop within certain constraints during autoplay, and keeps on regulating complexity also after the human player enters the game. The feedback to the players (in Figure 1, the arrow entering the "Player's decisions" block) is provided displaying the current level of stability, the numerical value S_{TOT}, and an arrow pointing up or down in case there has been some recent change. Further information is provided displaying the breakdown into five main categories (shown in Figure 2, at the bottom) which are rated from 1 star (minimum) to 5 stars (maximum).

Figure 2. Stability feedback panel

Players can effectively improve their stability by taking decisions according to this feedback. It is important to give to the player additional tools for management, in the form of extended government options, which in the mods are provided through an additional panel. In the previous example, we mentioned that control of a rival's motherland may cause instability. Treating those lands as part of the empire, or as viceroyalty, as commonwealth or as occupied area, results in significantly different stability calculations.

3.4 Roleplaying

As stated in Section 2.4, AIs can be competitive and at the same time roleplay their personality. Each civilization AI has been customized in order to expand in a way that is appropriate to the the civilization's role in real history. For example, a seafaring civilization, such as Portugal, will be active in colonising some locations in the East Coast of South America and in other continents, while maintaining a small mainland territory. This kind of behaviour is easy to achieve when the map is fixed, which it is in RFC: the implementation consists of assigning modifiers, rewards or penalties to the desired locations on the map. RFC RAND, however, uses generated maps, which makes it harder to achieve as there is no "South America". For RFC RAND, each civilization calculates modifiers based on latitude, longitude and other geographical data of the generated world, and on relative position of other civilizations cities. The calculations can be stored in tables (in case of RFC) where each cell represents the modifier of a map tile, or (in case of RFC RAND) can be done online when necessary, storing only geographical parameters.

In line with their personalization, different AIs have different victory conditions. For example, Portugal may have to found a certain number of colonies, or to monopolize overseas spice markets, in order to win. In this way, each AI is playing its own game with different goals, and the generated history will feature a situation in which the human player will need to stop some of those AIs from winning according to their victory conditions.

Personalization of AIs and monitoring of AI goals is performed in an integrated way by the PGH management system, as all data regarding geography, as well as achievement of goals, are providing feedback to the stability system.

3.5 Events and changes at runtime

A few events have been added to RFC. Among them:

- Plague that decimates units which are considered excessive (to restrict complexity / loading times);
- World Congresses are held where cities are traded, and AIs try to pursue their own goals;
- Bonus units ("conquistadors") are given when a civilization from Eurasia meets the New World civilizations for the first time;
- Great Depressions can be triggered based on economic calculations.

All these events, similar to the implemented concepts of Section 3.4, are managed in an integrated way with the stability system.

Direct intervention at runtime is also needed. In RFC RAND, there are cases in which later civilizations spawn in weak starting locations, which are calculated online. This may happen as the map is randomly generated (with some constraints), and the game may have been developing in unpredictable ways. In such cases, additional help might be needed in "cleaning" and improving the starting location, in order to give the civilization a balanced start.

An example of such changes is show in Figure 3. In the upper image, a location has to be assigned as starting point for a new civilization. The area is small, near tundra (grey tiles) and very close to another civilization's border (in red). Once the best tile in the area is selected, improvements (marked with yellow arrows in the bottom image) are automatically done, turning tundra into grassland (green terrain); mountains into hills; and adding iron, a strategic resource. These changes are not seen by the human player, and result into a more balanced starting location, which will eventually generate a more balanced history.

Figure 3. Changes made to a new player's starting location at runtime: the territory before the changes (top), and modified terrain (bottom) with arrows indicating changes

4. TESTS

We evaluated the effectiveness of the implementation explained in Section 3 by running tests. The goal of the mods is to generate a history that is interesting to play. In order to be considered interesting, it has to be judged as sound (realistic) and fun (well-balanced). These attributes are verified in the following sections.

4.1 Soundness of history

In Civilization, realism is intended to be "historical realism", which is something difficult to measure numerically. We tested this goal through a qualitative analysis of fifteen trials of RFC (i.e., on the Earth map).

We let the game autoplay until the birth of the United States of America, in late game. A player starting to play in 1775 AD would expect to find a realistic world coherent with of geopolitical situation of 1775 AD in Earth history (Table 4, second row). A scenario fixed at that date would feature such a situation through manual design. Conversely, RFC should achieve similar results through autoplay only (results in Table 3, from the third row down). During development of the mod, it was ensured that the civilization expand according to Earth history, and was

checked for all the possible human player's starting times, not just 1775 AD. As can be seen in Table 4, the generated games often show remarkable similarity to Earth history, with some interesting variations (more on this in Section 5.1).

Table 4. Dominating civilizations in each continent after autoplay until 1775 AD

Trial n.	North America	South America	Europe	Africa	Asia	Oceania
Earth history	UK / FRA	POR / SPA	FRA / HRI	TUR / POR	CHI	HOL / UK
1	FRA	POR / SPA	HRI	HRI	CHI	HRI
2	FRA / HOL / SPA	SPA	RUS / GER	ARA / SPA	ARA	HOL
3	UK / FRA / TUR	POR / SPA	RUS / HRI	ARA	TUR	HOL
4	UK / SPA	FRA / SPA	RUS / GER	UK / SPA	ARA / RUS	HOL
5	UK	POR / SPA	RUS / TUR	ARA / HOL	RUS / TUR	HOL
6	UK	SPA / HRI	HRI	ARA	RUS / CHI	
7	FRA	POR / SPA	FRA / RUS	UK	JAP / RUS	UK
8	UK	INC / POR	UK / FRA	UK / HOL / ETH	CHI / IND	UK / HRI
9	UK	SPA	GER	ARA	CHI	
10	FRA	POR / SPA	FRA	POR	JAP / IND	
11	UK	SPA	UK / RUS	UK / ETH	RUS / IND	UK
12	FRA	POR / SPA	FRA / HRI	ARA / FRA	RUS / FRA	UK
13	UK / FRA	FRA / POR	HRI	ARA / HOL	MON / ARA	UK
14	UK	FRA / SPA	FRA	ARA / FRA / HOL	MON / ARA	HOL
15	UK / FRA	INC	TUR / GER	TUR	TUR/ RUS	UK

Legend: UK = United Kingdom; FRA = France; HOL = Holland; SPA = Spain; POR = Portugal; INC = Inca; RUS = Russia; HRI = Holy Roman Empire; TUR = Turkey; ARA = Arabia; ETH = Ethiopia; GER = Germany; CHI = China; JAP = Japan; IND = India; MON = Mongolia; empty = unexplored

In Section 3.4, we mentioned roleplaying of AIs by making them aim to achieve specific goals. Although it is hard to demonstrate that numerically, an example of roleplaying behaviour can be seen in Figure 4. This figure is from an example game played by RFC RAND, in which the Dutch roleplay their colonizing behaviour on a random map. The motherland is located in the top middle of the map (the big, orange area), and it has founded several colonies in far away, exotic lands (the other orange areas).

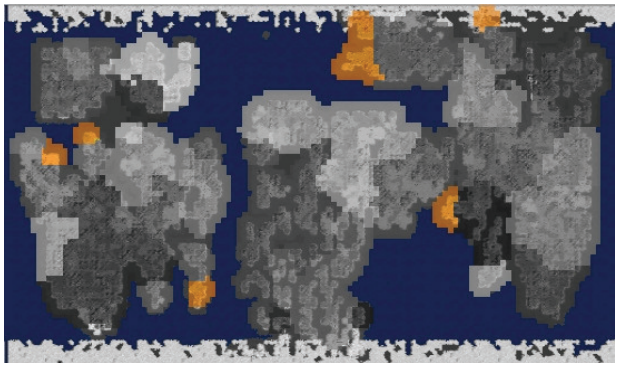

Figure 4. Expansion of the Dutch empire (in orange) in a randomly generated map

4.2 Strength balance

In Civilization, the most obvious way of measuring a player's strength is overall score. In Table 5 we report the results of 24 tests performed with RFC RAND, showing civilization scores at halfway point (250 turns) and the end of the game (500 turns). We ran four different games on random maps, stopped the autoplay and recorded the scores. For each of the four games, we ran three trials and compared the result with three different trials in which the PGH management system has been partially disabled (the code for triggering civilization spawns at later dates is still active). The games ran on a large sized map, which may contain a maximum of 24 players. We expect to see more competition for the victory in the former trials, whereas without proper management, there should be more risk of developing an unbalanced game.

Table 5. Scores of the top 3 civilizations, with or without management system

		Trial 1		Trial 2		Trial 3	
Half game scores		**Civ**	**Final score**	**Civ**	**Final score**	**Civ**	**Final score**
Game 1 GRE 1023 FRA 586 PER 519	w/	GER	3276	POR	3803	USA	3091
		RUS	2377	USA	3573	MAL	2445
		AZT	2157	GER	3279	RUS	2388
	w/o	USA	3586	GRE	4676	GRE	4162
		GRE	2861	RUS	3466	RUS	3228
		KHM	2569	USA	2898	KHM	3040
Game 2 CHI 575 ROM 396 PER 382	w/	VIK	3191	NED	3284	POR	3100
		FRA	3131	CHI	3045	INC	3031
		INC	3103	MAL	2908	FRA	2945
	w/o	CHI	3705	CHI	3524	CHI	4251
		FRA	3026	MAY	2825	INC	3991
		HOL	2884	ROM	2806	ROM	2965
Game 3 JAP 682 RUS 618 SPA 611	w/	RUS	3733	RUS	3434	CAR	3958
		MON	3130	CAR	3359	SPA	3909
		PER	2761	FRE	3085	KHM	3206
	w/o	USA	3067	SPA	3670	SPA	3445
		SPA	2968	USA	3256	FRA	3278
		MAY	2926	MAY	2979	USA	2924
Game 4 PER 717 RUS 683 IND 637	w/	ENG	3347	GER	2891	ENG	3546
		INC	3071	TUR	2763	MON	3047
		MON	2964	ENG	2754	TUR	2881
	w/o	IND	3429	GER	3487	RUS	3518
		ENG	3274	IND	3086	GER	3428
		PER	3120	PER	2938	IND	3174

In Table 5, results in the "w/" and "w/o" rows are quite different. Among the top three civilizations at half game, 11.1% and 47.2% (respectively with and without management system) are in the top three at the end of the game. The difference is significant (Mann-Whitney U-Test $p < .01$). Furthermore, within the three game endings, the same civilizations are present a higher number of times in the "w/o" case (70.8% against 41.6%), suggesting that there is more variety in the possible outcomes of a game by using the management system.

5. DISCUSSION

5.1 Discussion of the tests

Results in Table 3 confirm that for all the trials, the outcome is a believable alternative history which satisfies our expectations. An "alternative history" which would be regarded as realistic has to feature European powers (especially maritime ones) colonizing

the Americas and the coastlines of most territories, with other non-European civilizations, such as Ottoman Turkey and China, to dominate other areas.

Focusing on North America, it is considered plausible having the French monopolize the control of the colonies, as in Figure 5, or other European powers, as in Figure 6. In fact, the events in the history of the Earth could have been very different by just a small previous change (e.g., a different outcome of the Anglo-French war in North America). However, our generated history would not be realistic if in 1775 AD, North America were, for instance, dominated by Ethiopia through nuclear weapons. Such wildly divergent alternate histories will not be seen in the mods.

Regarding the effectiveness of the stability system for strength balance, results suggest that not only the balance is preserved from overpowering players, but also the whole ecosystem changes more dynamically. In Figures 7 and 8, strength ratings are plotted against time on the horizontal axis. Figure 7 shows an example of the development of civilizations' strength under PGH management, and it shows how civilizations' strengths can vary considerably. In comparison, Figure 8 shows an example of the development of civilizations' strengths without PGH management system. In the latter case, civilizations' strengths tend to grow without limits. This causes a clear divergence of strengths.

Figure 5. Alternative history: North America controlled by France, with three cities around the human player, in Trial #1.

Figure 6. Alternative history: East Coast of North America controlled by Holland (the city in orange, in the north east) and Britain (two cities in red, south and north), in Trial #5.

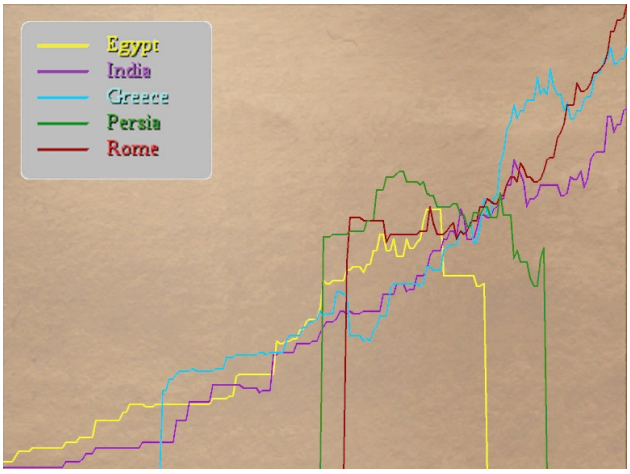

Figure 7. Record of civilizations power through the years, with rises and falls due to the management of stability.

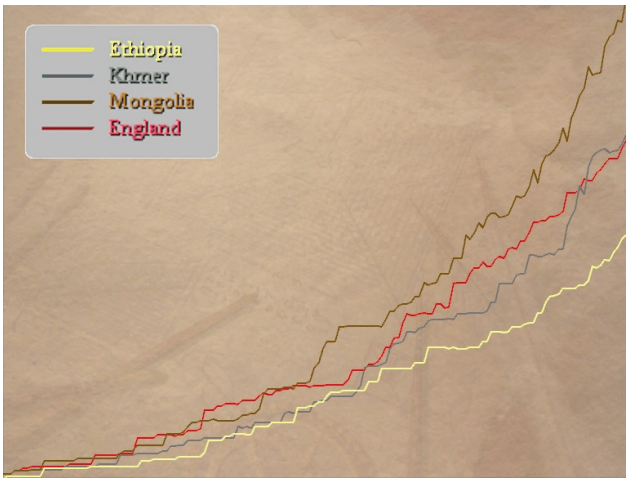

Figure 8. Record of civilizations power through the years without stability management.

5.2 Reception

The mod *Rhye's and Fall of Civilization* was praised by videogames press, labelled as "infinitely replayable" [2] and "a fresh new coat of paint to the core Civilization gameplay" [5]. High popularity followed in the gaming community, where it was regarded as an accurate history simulator. Alternative history threads appeared in forums to discuss the different outcomes of the autoplay and to post screenshots and stories. A version of the mod was included in the official expansion pack *Beyond the Sword*. Counting by number of downloads, it is the second most popular mod for *Civilization IV*, after *Fall from Heaven 2*.

5.3 Limitations

The biggest limitation of PGH is that, being an online process, it requires a loading time at the beginning of the game. In RFC and RFC RAND, depending on the amount of history to be generated, the process can take from less than one minute, to potentially up to one hour.

Such a long time is of course unacceptable in a product for casual users. We name a few methods to reduce loading times:

- Some aspects of the game which are irrelevant to the creation of the history may be disabled while autoplay is running. Content that is not significant to the human player can be broadly sketched. An example of this concept can be found in *Championship Manager* / *Football Manager* series: football players of lower leagues cannot be accessed and feature fictitious names, nevertheless they are part of the game's ecosystem.
- Autoplay can start at a later time, designing manually the intermediate starting point. In RFC, the first start corresponds to the year 3000 BC, and the intermediate start to 600 AD. Generation of a world in the Middle Ages or Renaissance is significantly faster using the intermediate start.
- It is possible to generate interesting worlds, place them on a central server and let users download them.

5.4 Comparison

In the introduction, we mentioned the game *Dwarf Fortress* as another example of world history generation. It features a very detailed process of map making, whereas RFC and RFCRAND are more focused on the subsequent development of civilizations, managing complexity and balance through the Stability System, as well as other tools to address the course of history. In general, mods for *Civilization IV* have more stringent constraints about realism (especially RFC on Earth map), in order to mimic the mechanics of Earth history.

Both *Dwarf Fortress* and the mods described in this paper have to face the same issue of processing time required for generation. It should be noted that, in both works, the act of just watching the results of the generation is regarded as enjoyable. Tracking world history events in Dwarf Fortress is interesting to the extent that a fan-made program was made to store all details in a database. For RFC, in forum discussions, users often express elation at the results they observed and share their experiences.

6. CONCLUSION

In this paper, we introduced a novel approach to Procedural Content Generation (PCG), called Procedurally Generated History (PGH). PGH consists in building a game world history through autoplay. Our purpose is to procedurally generate an interesting world to play, which we achieve through the use of autoplay to make the AIs generate the game's ecosystem in a natural way. A management system is necessary to force the game into a correct or reasonable growth through balancing players' strengths.

Our approach was implemented in a two mods for *Sid Meier's Civilization IV*. We evaluated the soundness and balance of the ecosystems generated by the mods, and found that they are generally realistic with regards to possible alternate histories, and tend to be well-balanced.

Future work includes improving generation time in order to be reasonable for any desired starting time. The concept of disabling certain features of the game in order to have a faster autoplay can also be considered spatially, using an adaptive map, where content is generated in detail near the player and only sketched in other areas, as long as they are not revealed to the human player.

ACKNOWLEDEGMENTS

This research is partially supported by the Japanese Society for the Promotion of Science. We also thank Firaxis Games for providing the source code of *Sid Meier's Civilization IV* and making possible to use the game as a testbed for our research.

7. REFERENCES

[1] Ashlock, D. and McGuinness, C. 2014. Automatic generation of fantasy role-playing modules. 2014 IEEE Conference on Computational Intelligence and Games (CIG) (Agosto 2014), 1–8.

[2] Chick, T. 2007. "Sid Meier's Civilization IV: Beyond the Sword". Yahoo! Games. https://web.archive.org/web/20081007042243/http://videogames.yahoo.com/pc/sid-meiers-civilization-iv-beyond-the-sword/review-523931-2 Retrieved 30 May 2010.

[3] Doull, A. 2008. The Death of the Level Designer: Procedural Content Generation in Games. http://njema.weebly.com/uploads/6/3/4/5/6345478/the_death_of_the_level_designer.pdf

[4] Johnson, S., Playing to Lose: AI and Civilization, Game Developers Conference, 2008. Available: https://www.youtube.com/watch?v=IJcuQQ1eWWI

[5] Park, A. 2007. "Civilization IV: Beyond the Sword Review". GameSpot. Available: https://web.archive.org/web/20090627021825/http://uk.gamespot.com/pc/strategy/civilizationivbeyondthesword/review.html? Retrieved 30 May 2010.

[6] Riek, L.D. 2012. Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. Journal of Human-Robot Interaction. 1, 1 (Aug. 2012).

[7] Sorenson, N. and Pasquier, P. 2010. Towards a generic framework for automated video game level creation. Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications), vol. 6024. Springer LNCS, 130–139.

[8] Tainter, J. A. 1990. The Collapse of Complex Societies; Reprint edition.; Cambridge University Press: Cambridge, Cambridgeshire; New York.

[9] Togelius, J., Yannakakis, G.N., Stanley, K.O. and Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. IEEE Transactions on Computational Intelligence and AI in Games. 3, 172–186.

[10] Togelius, J., Champ, A.J., Lanzi, P.L., Mateas, M., Paiva, A., Preuss, M. and Stanley, K.O. 2013. Procedural Content Generation: Goals, Challenges and Actionable Steps. Artificial and Computational Intelligence in Games. Simon M. Lucas and Michael Mateas and Mike Preuss and Pieter Spronck and Julian Togelius. 61–75.