Self-Adapting Dynamically Generated Maps For Turn-based Strategic Multiplayer Browser Games

Gonçalo Pereira
INESC-ID / IST
Av. Rovisco Pais
1049-001 Lisboa
Portugal
gdgp@ist.utl.pt

Pedro A. Santos
IST
Av. Prof. Cavaco Silva - Taguspark
2744-016 Porto Salvo
Portugal
pasantos@math.ist.utl.pt

Rui Prada
INESC-ID / IST
Av. Prof. Cavaco Silva - Taguspark
2744-016 Porto Salvo
Portugal
rui.prada@gaips.inesc-id.pt

ABSTRACT
In this paper we describe a method to create strategically and visually rich game map environments for turn-based strategy multiplayer browser games. Our method creates dynamic maps which expand according to the players' subscriptions pattern and adapt to the players' choices. This method mitigates current level designer limitations and contributes to the solution of balancing problems in turn-based massively multiplayer browser games.

Categories and Subject Descriptors
I.3.6 [**Computer Graphics**]: Methodology and Techniques

General Terms
Algorithms, Design.

Keywords
Strategy browser game, dynamic maps, self-adapting maps, expanding maps, multiplayer game balance.

1. INTRODUCTION
Internet-based gaming has grown tremendously in the last few years and it is possible to find many games from different genres [1]. Among these are the browser games which are defined by only requiring internet users to have a browser installed in their computer in order to play. Some are multiplayer and take advantage of the internet's widespread nature to link many players into a group game experience. Communities in these games are large and can vary from several hundreds to thousands of players gathered for a collective game experience [4].

With a few exceptions multiplayer browser games fall into two main categories, namely Role-Playing Games (RPG) and strategy games [3]. In strategy games, the player usually can choose some special traits ("race", "tribe" or "faction") that give him specific advantages and disadvantages. These influence economic and military development and are sometimes influenced by the environment. In this game genre, player location, terrain type and richness in resources influence the strategic balance and player challenge. These combined with the map's visual detail affect the player's experience. Many strategy multiplayer online games currently use models of map generation which cannot provide both map complexity (visual and strategic) and scalability for the multiplayer context.

Our goal is to create a game experience where players are placed in a detailed and evolving map environment. This is accomplished by expanding maps dynamically as players enter. The idea is that maps should not limit either game complexity or scalability, they should simply enable the best strategic and diverse environment to be created and experienced.

2. EXISTENT MAP CREATION
There are two main ways of creating maps, manually or through procedural generation. The manual creation method is characterized by requiring the creator's input for every detail of the map. This makes the level designer creatively free, facilitating the creation of map features. However, the large amount of human input required makes it time consuming for a massively multiplayer online context. Another disadvantage is that if a map is created beforehand it cannot properly account for the player's choices, for example the race.

Procedural content generation (PCG) has been used in games for several decades (for maps, names and others)[1]. It enables to create great amounts of different game content dynamically and in a more practical way than handmade [8]. Several techniques are used to ensure that variety is present in the content created and at the core of many procedural methods are techniques based on pseudo-random number generation, probabilities and fractals [2].

One great advantage of the PCG approach is the time to generate a map. Usually the generation time is small enough that players can parameterize and generate the map on the fly like in Civilization[2] and many roguelike games[7]. This approach is composed of several stages that vary depending on the game [5]. Some common stages are heightfield generation [6], resources distribution and terrain feature creation.

[1] In 1984 the game Elite (http://www.iancgbell.clara.net/elite/) already used PCG extensively. These techniques continue to be used in games like Spore(2008) (http://www.spore.com/ftl)

[2] http://www.civilization.com/

However, on a massively multiplayer online context the same method cannot be used since the amount, subscription time and choices of players are unpredictable. Games like Travian[3] face this problem and solve it by statically creating generic game maps. Joining players are placed in the game expanding from the center of the map towards its limits. This generalization results on maps that have little influence on the game.

The existent map creation cannot simultaneously accommodate a rich/game influencing strategic environment and a large dynamic community (of at least hundreds) of players for the massively multiplayer online context. The obstacles are the unpredictability of the amount, subscription times and choices of players.

3. SELF-ADAPTING DYNAMIC MAPS

Our approach intends to give players an improved multiplayer online experience by creating a rich strategic environment where the map's characteristics are both influenced by player's race and time of subscription. This is done by expanding maps with each player subscription and generating territory specifically to accommodate the players' choices.

This approach was developed for the game Almansur Battlegrounds[4]. It *is a strategy game of politics, economy and war set in the early middle ages or, in some scenarios, fantasy world*. The game is turn-based and the map has a deep influence in the game, ranging from path-planning to the cost and productivity that the same building can have in different terrains. In this environment there are several races for fantasy games: Barbarian, Dwarf, Elf, Human and Orc. Each one has different needs in terms of terrain characteristics, for example Orcs need swamps for better development but Elves need forests. The player starts as a lord from a race of his choice with a small army and territory which he can then develop economically, make alliances or expand by conquering neighbouring territories. Until now this game has relied on manually created maps for a fixed number of players. Our goal is to provide growing game maps while maintaining the game balance and keeping the existent strategically rich environment.

3.1 How to Create the Maps

At the core of any map is its representation model. Almansur Battlegrounds uses a vertically aligned hexagonal grid where the terrain unit is the hexagon, but the method could be adapted to other grid models.

To create the maps we use an iterative procedural generation method which has several input parameters. The most important are the terrain prototypes for each race that can be played in the game and the number of turns a player placeholder takes to expire. A *terrain prototype* contains base values for each of the terrain's characteristics which the game uses. Examples of terrain characteristics are altitude and fertility but any others can be used.

These prototype terrains are carefully crafted by the level designer, since they are the basis for an adequate and balanced map generation. Besides the base values of each prototype the level designer also groups prototypes by categories and evaluates the quality of each within their category based on heuristics. The categories will enable the use of the adequate prototypes for the different terrain purposes. Examples of categories are "race dwarf", "feature lake" and "neutral terrains". The evaluation of individual prototypes in a group also enables their differentiated use, based on their quality regarding that group.

A *player placeholder*(PP) is a specific position in the map that can be a starting point for any entering player. As we will see below, when there are no players joining for a preset period of time, the generator must avoid upcoming players to be placed adjacent to others much more developed. The parameter with the number of turns a player placeholder takes to expire specifies how many turns can elapse before an unused PP becomes unavailable.

In order to support an ever growing map, it is necessary to define how it will grow. The idea is that the map should expand in an interleaved way between neutral zones and player zones which balance conflict and expansion. A *zone* is a set of terrain units generated with a specific purpose based on one or several categories of terrain prototypes. A player/neutral zone schema was chosen[5] based on game tests, with manually created maps previous to this method. It is illustrated in figure 1.

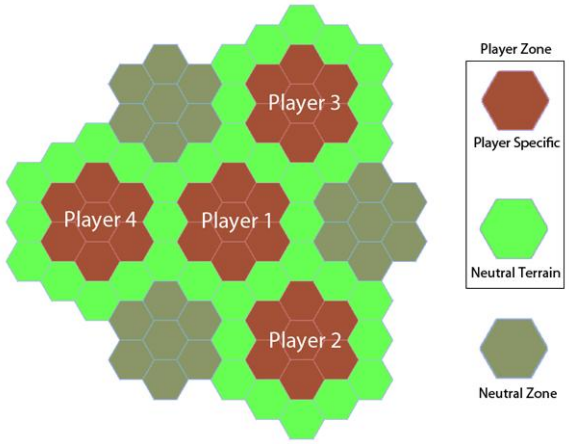

Figure 1. Player/Neutral zones schema

The schema shown represents player 1 which is surrounded by three player zones and three neutral zones. These player zones represent other players which cause direct points of conflict with the player in the center. Notice that these zones are composed of two kinds of terrain, the player specific and the neutral territories.

Based on this player/neutral zone distribution we can then scale our generation model to grow as needed. The creation of new zones then depends only on player entries, where we impose the constraint that one player must always be placed near an already existent one. Once a new player is created the remaining possible neutral and player zones surrounding it are made available for the generation process allowing more players to join. The growth of the map follows a spiral order for both player and neutral zones. This ensures that a joining player will be placed in the oldest unused PP available so the elapsed game time between new players and others already playing is globally the smallest possible. Other similar expansion ordering schemas can be used if they retain the same player adjacency property. Note that the neutral zones placement also follow the spiral scheme, but their

[3] http://www.travian.com/

[4] http://www.almansur.net/

[5] Personal communication with Marco Quinta from Almansur.

actual placement only happens when the next player zone is created. This spiral schema is illustrated in figure 2.

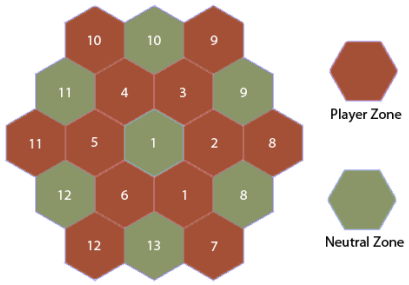

Figure 2. Spiral expansion

The figure illustrates a normal evolution, where there is always players entering. Notice that zones are now represented as just one hexagon for image simplicity. In this example each different number represents one expansion iteration. When the first player joins one neutral and player zone are created, when the second player joins a player zone alone is created, and so on.

However, player entries are not predictable, so we can have different evolutions. If there are no player entries for a given amount of game time (in our case turns) the generator marks as expired zones the neutral and player zones for which the initially set parameter number of *turns to expire*(TE) has been reached. If players could enter in these places there would be a high probability that the older players were much more developed and could easily overrun the new ones. The *expired zones* are generated as neutral impassable zones where map expansion stops. To accomplish this they are created as natural barriers, like a lake, impossible for game units to cross. As a consequence the map will not grow in a pure spiral fashion but will only continue to expand in a few places. This system allows the map to expand or contract the places for expansion depending on the flow of players. An example of such a situation is illustrated in figure 3.

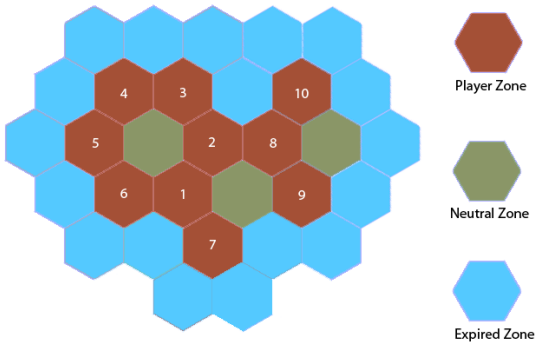

Figure 3. Spiral expansion with expirations

In figure 3 we have a situation where the first eight players entered without having any expiration. Then all the neighbour places from the first seven players expired and two more players joined. Finally no more players entered and the map closed itself.

The variation of the TE time influences the map shape and player concentration. For a fixed number of players and subscription times, a lower TE value leads to a more dispersed player distribution/map while a higher value approximates the growth to the normal, more agglomerated, evolution.

3.2 Zone Generation

Based on this schema the level designer can influence the gameplay by varying the size (radius in hexagons around the player zone center) of the player specific terrain (PSS) and/or the neutral terrain (NTS). For example a game with PSS=2 and NTS=1 is less aggressive than another with PSS=2 and NTS=0.

To generate a concrete terrain a prototype is picked and then all its base values are randomized by adding or subtracting a random offset to enable the creation of many different terrains from a single prototype. The terrains in the player specific terrain are generated with the prototypes from the category of the player's race. The neutral terrains and normal neutral zones are generated with the "neutral terrains" category while feature zones (in neutral zone places) are created based on the category of prototypes for that feature.

3.3 Balancing

With this method a great part of the game balance is the responsibility of the level designer for the generator parameterization. Especially important is the correct definition and evaluation of the terrain prototypes of all the races, since the player specific terrain generation picks prototypes based on a preset ordered combination of different prototype qualities.

The fact that players enter at different game times also creates problems to the game balance since it originates significant development differences between players. A first step to mitigate this is the player/neutral zone schema where the chosen configuration balances the number of direct conflict points a player has to manage, which previous player tests have proven it to be successful (see footnote 5). Also the spiral expanding schema with PP expiration ensures that a new player never enters near a much older and developed player by discarding unused PPs that have passed the acceptable number of turns a player can have as a head start from another.

4. PRACTICAL RESULTS

In this section we show some examples of different maps created by our method in the implementation developed for Almansur. The generation of these examples followed the same zone sizes described in section 3. Also notice that players start only with a single hexagon (further discussed in the next section).

The example in figure 4 is a map with eleven players that is partly closed because of expired zones since there are few players entering but still has some expansion places. In figure 5 we can see the evolution of the map view for the first player to join the game in the previous example. In (a) we have the view right after the player joined and in (b) the view after all his neighbour PPs have been filled. The example in figure 6 is a map with twenty one players and is a case of a closed map, since no players joined for a long period of time. The terrain visual detail is abstracted and each isolated colored hexagon represents a different player.

Another important factor we tested was the amount of players that we could add with an "acceptable" generation time for the subscribing player. After several load tests we found that with the current implementation we can add between 1000 and 1100 players with waiting times always under 4 seconds and running the game on a laptop computer (Intel T7700 - 2.4GHz).

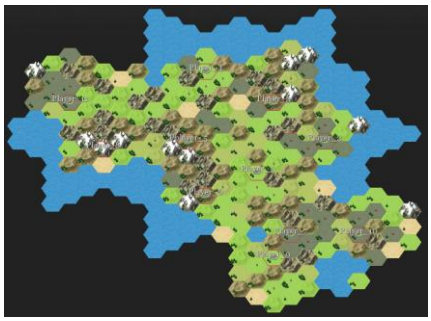

Figure 4- Example of a semi-closed map

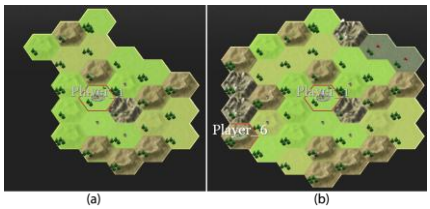

Figure 5 - Player view changes

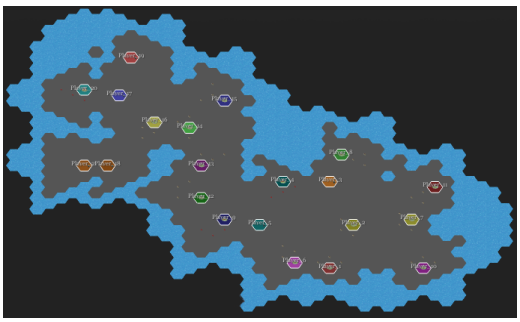

Figure 6 – Example of closed map

5. DISCUSSION

In our method of map generation there is a balance between the level designer's input and a procedural generation which enables us to create rich dynamic maps for a large number of players.

The problem of delayed player entries is addressed by our schema, which helps diminish the imbalances created by them. However, the problem is not eliminated. Two further possibilities of addressing the problem can be a protection period for new players or increased strength of new players. These would diminish the risk of delayed new players being overrun by any older players.

Beyond that, our schema also takes advantage of the procedural methodology combined with the randomness from the flow of players entering at a given time. The reaction of the method to the players' flow creates maps with random shapes which help avoid player boredom and promote replayability.

The consequence of playing in a dynamic map for players is that they must understand the nature of the map so they aren't surprised or frustrated. The nature of these maps is one of a growing map where expanding places contain ungenerated terrain for a short time, in which players cannot move. This terrain also cause player view update issues. To reduce its impact it is possible to generate all the terrain for a player, but let him only start with his most important one.

6. CONCLUSIONS

Our approach enables us to create strategically rich environments on maps for turn-based strategy massively multiplayer online browser games. It also contributes to the solution of delayed player entries and player unpredictability. Nonetheless, the problem is not eliminated and is subject of future work.

Regarding the game designer, he is freed from the limitations of having to create the game around the assumption of a generic or size limited map. Game maps with strategic complex situations can be procedurally created for the massively online context, but there is an increased responsibility regarding the prototype terrains and method parameterization.

From the player's perspective the risk of tiring from repetitive play is reduced since map generation depends on player entry pattern and choices so that each game has a completely different map. Using the techniques described in this paper, the massive multiplayer browser strategy game can attain a depth and strategic challenge unparalleled until now.

7. ACKNOWLEDGMENTS

Our thanks to Almansur for supporting our ideas and supply base knowledge about previous game experience. We would also like to thank PDM&FC for technical support.

8. REFERENCES

[1] *Analyst: Online Games Now $11B of $44B Worldwide Game Market*, 2009. Retrieved June 20, 2009, from Gamasutra: The Art & Business of Making Games: http://www.gamasutra.com/php-bin/news_index.php?story=23954

[2] Lecky-Thompson, G. W. 2001 *Infinite Game Universe: Mathematical Techniques*. Charles River Media, Inc.

[3] *List of multiplayer browser games*, 2009. Retrieved June 20, 2009, from Wikipedia: The Free Encyclopedia: http://en.wikipedia.org/wiki/List_of_multiplayer_browser_games

[4] *Massively Multiplayer Online Game*, 2009. Retrieved June 20, 2009, from Wikipedia: The Free Encyclopedia: http://en.wikipedia.org/wiki/MMOG

[5] Prachyabrued, M., Roden, T. E., and Benton, R. G. 2007. *Procedural generation of stylized 2D maps*. In Proceedings of the international Conference on Advances in Computer Entertainment Technology (Salzburg, Austria, June 13 - 15, 2007). ACE '07, vol. 203. ACM, New York, NY, 147-150. DOI= http://doi.acm.org/10.1145/1255047.1255077

[6] Olsen, J. 2004. *Realtime Procedural Terrain Generation: Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games*. Retrieved June 18, 2009 from Oddlabs: http://oddlabs.com/download/terrain_generation.pdf

[7] *RogueBasin*, 2009. Retrieved June 20, 2009, from RogueBasin: http://roguebasin.roguelikedevelopment.org/

[8] Doull, A. 2008.*The Death of the Level Designer: Procedural Content Generation in Games*. Retrieved August 30, 2009, from Ascii Dreams: A roguelike developer's diary: http://roguelikedeveloper.blogspot.com/2008/01/death-of-level-designer-procedural.html