Adaptative Map Generation For Turn-based Strategic Multiplayer Browser Games

(extended abstract of the MSc dissertation)

Gonçalo Duarte Garcia Pereira
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisor: Pedro Alexandre Simões dos Santos
Co-Advisor: Rui Filipe Fernandes Prada

Abstract—This thesis addresses the problems of Random Map Generation which are strategically and visually interesting for turn-based strategy Massively Multiplayer Online Games (MMOG). First we present the results of the research regarding works related to this subject. Previously used methodologies are reviewed and the techniques used in some MMOGs analyzed. Next we describe a method to create strategically and visually interesting game map environments with a MMOG as a case study. The game used is Almansur which belongs to a sub-genre of the MMOGs, the turn-based strategy massively multiplayer browser games (TSMBG). Our method uses dynamic maps which expand according to the flow of players’ subscriptions and adapts to their choices. It also eliminates current game designer limitations and promotes the development of scalable strategy focused games. Furthermore it mitigates balancing problems in TSMBG. Finally we present some results from our method and the conclusions regarding the gameplay experienced by players and the scalability of the developed method.

I. 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\(^2\).

With a few exceptions multiplayer browser games fall into two main categories, namely Role-Playing Games (RPG) and strategy games\(^3\). In browser strategy games, the player usually starts with a single “village”, “castle”, “territory” or “planet”, and then must balance economic development with military development, in order to gain dominance over its neighbours. To increase variety, each player can choose some special traits that give him specific advantages and disadvantages. These trait sets can be represented as “races” or “tribes”. Strategy Browser Games have been evolving for more than a decade into more complex and graphically appealing games. A way to see this evolution is through the interfaces presented to players to access the game data and provide different game views such as a map. Some started as text-based games with no representative graphics whatsoever (1996 - Earth: 2025\(^4\)), then others appeared with simple maps that are used as a simple referencing environment without any actual relevant influence in the game (2003 - Travian\(^5\)) and in present time there are some which use fully detailed maps that influence the game (2007 - Almansur Battlegrounds\(^6\)).

In strategy games, player location, terrain type and richness in resources influence the strategic balance and player challenge which then affect the player’s experience. Beyond the map’s strategic influencing characteristics there is also the visual effect the map has. So the map’s variety of visual characteristics such as shape and detail also play an important role regarding the player’s experience and its desire to continue playing the game. A game where the map always has the same shape easily becomes uninteresting for players since it does not bring any new experience. Another factor which creates interest is the options the players have available to explore and create different gameplay experiences. Currently many strategy multiplayer online games are using a model where a map is created to accommodate a given fixed number of players and then they are placed as they subscribe in a specific ordered way which varies with the game. In these maps there is currently a tradeoff between the players’ choices/map complexity and the scalability for the multiplayer context. The most detailed and strategically complex maps are handmade and thus hard to scale for many players and the generated maps are usually very simple and strategically poor. Here the map is used just as a referencing environment for player placement.

Our goal is to create a map generation model which can

---

4http://games.swirve.com/earth/
5http://www.travian.com/
6http://www.almansur.net/
have both the advantages of the handmade and procedural generated maps and properly adapt to the TSMBG (Turn-based Strategy Massively Multiplayer Browser Game). To do this we create a procedural generation method which has the ability to create interesting maps but depends on the game designer for a very specific and detailed parameterization in order to be able to add the map detail. Furthermore, in order to adapt to the TSMBG context we generate maps dynamically as players subscribe, thus enabling the game a greater variety of options for the player to explore since the map terrain can be generated to accommodate the player’s choices. The maps created in this way should not only evolve according to the players and their choices but also be unbounded by the uncertainty of when will new players join, what choices will they make or how many will join. We aim to create maps that do not limit either the game complexity or game scalability, but simply enable the best strategic and diverse environment the game designer can provide for a given game.

In the following sections we’ll start by reviewing some of the current techniques of map creation and existing multiplayer strategic online games. Then we’ll present our approach to an adaptative map generation method for the TSMBG. This will be followed by some results and its analysis. Finally some discussion, future work and conclusions are presented.

II. RELATED WORK

There are two main ways of creating maps. Manually or through procedural generation[1]. The manual creation method is characterized by requiring the creator’s input for every detail of the map, whether it is a terrain type or a resource distribution. This need makes it very time consuming. The bigger and more complex the map is the more time it requires to be created. Examples of this method can be found among many games, for example Weewar7 and Battle for Wesnoth8. The application of this method to the context of map creation for multiplayer online strategy games then imposes a restriction in terms of size since the time spend on map creation is limited. If the number of players is large (over a few hundred) then manual generation becomes impractical. So this method limits the amount of players that can join in a single map but allows a creatively unbounded creation. This last characteristic is very important because it allows the creation of historically themed maps. On other maps it also allows the creation of strategic challenges like mountain ranges or rivers with specific strategic points of passage. There are several games using this type of maps, for example in Weewar we have very detailed maps that were handmade such as a map representing Earth with its terrain characteristics accurately. However the game supports very few players. It can have a maximum of 6 players on a single map, and only if the map allows it, since the player places have to be specified on map creation. So it is possible to

7http://weewar.com/
8http://www.wesnoth.org/

have a strategically rich environment, but the players can only play in a fixed set of maps and with a fixed small number of players at the same time.

The procedural generation approach has the advantage of consuming much less time to create maps than the handmade method. The approach is composed of several stages which vary depending on the game. Some common stages are heightfield generation, resources distribution and terrain feature creation. The most common is heightfield generation as usually is the base for all the others. To do this there are many actual algorithms with many different approached which have been described in previous works [2][3][4][5][6][7][8][9][10][11][12], some of the most common algorithms used are Perlin Noise[13][14] and midpoint-displacement[15]. An example of complete map creation can also be found in [16]. These maps can then be adapted to accommodate as many players as the game needs, but due to the unpredictability of player’s choices and player entry times and amount, they must be generic enough to be able to satisfy every player’s option. Choices such as a faction (or race) can heavily influence the advantage or disadvantage a player experiences from the terrain in a map[17]. In Battle for Wesnoth the terrain type influences units on both movement cost and defense. So maps have to be created in order to satisfy the different player choices equally well which is achieved by making the player as much independent from the map as possible. This independence results in the decrease or elimination of map features and its influence in the game which results in a strategically poorer environment. The strategic component that a map can add is left out and it is mostly used as a simple board for the game to evolve or a place to reference players’ locations. This kind of maps can be seen in games such as Travian where generic game maps are created at a given server’s start and players are placed in the game starting from the center of the map. The map is then filled towards the map’s limits in a circular fashion where the main strategic influence the terrain creates is the players positioning. Another example is Civilization IV which has a map generator capable of creating detailed, balanced and strategic maps procedurally, but only for a limited number of players.

None of the approaches described above can both accommodate a rich strategic environment where the map is an important component that adds interest to the game and at the same time be able to accommodate a large and dynamic community (of at least hundreds) of players. The first approach is very time consuming and impractical for many players and the second one produces strategically poor maps in order to be able to accommodate a varying number of entering players and their choices.

III. FROM CLASSIC TO DYNAMIC MAPS

To create a solution to the problems of balanced random map generation and map growth, a game context was needed since such a solution is dependent on the game’s characteristics. Therefore, this work was developed in collaboration with the game Almansur where the problems we formalized
first emerged. Our solution is then divided into two different contexts:

- The procedural creation of interesting balanced maps with Almansur’s classic multiplayer game context;
- The procedural dynamic growth of a map with Almansur’s TSMBG game context, while maintaining interest and balance;

The first context is common in most strategy games which offer the multiplayer game mode and is the base work needed for the second. This intends to solve some of the problems known in massively multiplayer games while upholding the goals of map interest and balance.

We will present a generic part of our solution first, then we will proceed with some of the details needed in a complete specification of the method for Almansur’s case.

A. Game Context

Throughout our work we will use a vertically aligned hexagonal grid with the hexagon as our smallest terrain unit, but the solutions presented can be adapted to other systems.

Our grid system choice was based on the advantages of hexagonal grids but also on the fact that our case study uses this system. Almansur 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 different troop movement speeds for each race to the difficulty and consequent 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. So as a basis for good initial development, game balance and visual coherence each land must have in its initial position terrains created specifically for its race. In Almansur 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 all alliances or expand by conquering neighbouring territories. To create our maps we use an iterative procedural generation method which has several input parameters. The most important to understand how maps grow in our solution is the terrain prototypes for each race that can be played in the game. A terrain prototype contains base values and offsets for each of the terrain’s characteristics which the game uses. Based on these prototypes a concrete game map terrain (represented as an hexagon) can be generated by randomizing its base values with the following formula (\( v \) stands for value):

\[
v_{\text{final}} = v_{\text{base}} - v_{\text{offset}} + 2 \times \text{random}_{0,1} \times v_{\text{offset}} (1)
\]

This randomized process enables that very few prototypes originate many different map terrains.

These prototype terrains are carefully crafted by the game designer, since they are the basis for an adequate and balanced map generation. Besides the base values of each prototype the game designer also groups prototypes by categories and evaluates the quality of each within their category. The categories will enable the use of the prototype for different purposes. These prototypes are grouped based on the use they will have, so we have one category for each race, for each terrain feature type and for neutral terrains. The evaluation of each prototype enables the generator to differentiate prototypes within a single category and use them differently. It is done based on the game designer’s heuristics to classify a given prototype’s quality inside that category. For example, in the category “race dwarf”, prototypes with a higher altitude value are more suitable for dwarf populations and so will be given a higher quality value than the ones with lower values for this characteristic.

To be able to support an ever growing map the idea is that it should expand by following a pattern of interleaved neutral and player zones. This aims to balance player expansion and military conflict, since the game permits military action from the moment the player enters the game (players are assigned initial armies). A zone is a set of terrain units (single terrains or hex) that are generated with a specific purpose based on one or several categories of terrain prototypes. Based on game tests, with manually created maps previous to this method, a player/neutral zone schema was chosen and is illustrated in figure 1. The schema shown represents a player 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, so in this case we chose to have three such points. Also notice that these zones are composed of two kinds of terrain, the player specific and the neutral territories.

The player specific terrains are created to be a combination of the most suitable ones for the player’s race. The neutral terrains are randomly generated from all the neutral terrain prototypes but a percentage of them are influenced to also be the most suitable (from the “neutral terrains” category) for the neighbouring player’s races. This enables to accommodate the players’ races well and encourages some initial expansion. The neutral zones have two main purposes. The first is the resources they contain, which are reason for disputes and create indirect points of conflict. The second is the strategic influence given by the terrain type, for example a lake neutral zone forces troops to move around it and might create choke points to the passage of troops. Another example is a mountain range which creates a barrier to the movement of troops since they spend more time to cross them.

**D. Growth Control**

The schema presented in figure 1 is the most tested but several variations can be easily achieved by varying the size of the player zone. This then influences the size of the neutral zones and the overall pattern created. Since the player zone is composed of two kinds of terrain we create two parameters which enable the variation of the zone size. The first is the player specific size (PSS), which represents the radius, in terms of terrain units, of the player specific terrain generated for the zone. The second is the neutral terrain size (NTS), which represents the radius of neutral terrain added around the player specific terrain. The implicitly defined neutral zone size (NZS) is then defined by:

\[
NZS = \begin{cases} 
PSS + NTS - 1 & \text{if } NTS > 0 \\
        PSS & \text{if } NTS = 0 
\end{cases}
\]  

(2)

The variation of the PSS and NTS values can attribute different levels of intensity and aggressiveness to a game. For example a game with a player zone composed by PSS=2 and NTS=1 (illustrated in figure 1) is less aggressive than another where the player zones is composed by PSS=2 and NTS=0. This happens because players in the later case are placed closer to each other and have less terrains to expand to.

As previously described, player zones have the player specific terrain and the neutral terrain and these are generated differently. In order to obtain a better balance between all the different races we create a configuration which aims to originate sets of player specific terrain equally useful for each player regardless of their race. It is called the player terrain composition (PTC) and the same PTC is used for all races. It qualitatively describes the player specific terrains by providing several associations between qualities of prototypes and influenced terrains.

**IV. CLASSIC CONTEXT SOLUTION**

In order to create the desired maps for the classic multiplayer context a sequential process is used where the map data is processed in several stages. The stages used are: creator input, create all player zones, populate players’s borders, randomize neutral zones, create map features, randomize outer coastline, player terrain adjustment and map created.

In these stages we faced several specific problems, however since we intend to focus on the dynamic context we will specify only the relevant method details that most contribute to the scalability from this context to the dynamic one.

In our solution we must ensure every player is connected to the rest and that the growth pattern is followed. To do this when player start places are needed, new ones are created based on the ones from the previous and already existent layer (the initial player layer is represented in figure 3). Based on the ancestor’s direction of a player start place and its position we can determine which new player start places are created. Considering the grid system of Almansur there are six possible situations which are illustrated in figure 2. Notice that with this system we can also easily determine the neutral zone places of the pattern. In the middle of every two new player zones there is a neutral zone represented in figure 2. We can then formalize the process of player places and neutral zone place generation by the following formula
In order to create dynamic multiplayer game maps an asynchronous sequential process is used where the map data is processed in several phases. The first phase is the configuration where all the necessary parameters needed for the generation process will be loaded into the generator. The second is the player generation, this phase adds a player to the map based on the choices he did. The third phase is the turn processing phase where the map generator updates the map based on the choices he did. The fourth phase is the player generation, this phase adds a player to the map based on the choices he did. The fifth phase is the turn processing phase where the map generator updates the map based on the choices he did.

1) **Configuration**: This phase of generation is very similar to the “Creator Input” stage in the Classic Multiplayer Context part of the solution. Comparatively to the previous context a new parameter is needed with the number of turns a player start place takes to expire which specifies how many turns can elapse before an unused player start place becomes unavailable. The general idea in this stage is that all the non-player configurations should be loaded and saved (persisted).

2) **Player Addition**: This phase is also very similar to the player creation stage in the classic context. However, there are two major differences, first only one player is added each time this generation phase is executed and second the actual map growth is different.

The map growth idea is still similar to the one explained in III-C but since we now intend to have a map expansion driven by the flow of player subscriptions, the player start place creation and usage will be different. Now we want the map to grow one player at a time and the expansion as formalized in 3 but only when a player enters and not on a complete layer basis (as in the classic context).

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 player start place 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 actual placement only happens when the next player zone is created. This spiral schema is illustrated in figure 4.

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 5.

In figure 5 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 number of turns to expire time influences the map shape and player concentration. For a fixed number of players and subscription times, a lower turns
to expire value leads to a more dispersed player distribution/map while a higher value approximates the growth to the normal, more agglomerated, evolution.

A. Turn Processing

In this phase the first step is to advance the number of turns from the generator. Next we check if the map is already closed, if it is then we report that it is already closed and there is nothing to be done. Next we must check for places that might have expired when we advanced the turn.

At the beginning of the turn processing we check the free player start places available for those that the number of turns to expire since their generation has passed. The next step is to check for neutral (or player) zones adjacent to players (or neutral zones) already placed that are not going to be generated and must be filled so the players does not stay in contact with ungenerated terrain for long. These problems are exemplified in figures 6 and 7.

VI. RESULTS

For our tests we gathered data in several ways, directly, by evaluation procedures (we created a land evaluation method) and player questionnaires. During the evaluation we analyzed the interest the maps create on players and the effects that different map growth parameters and a dynamic map generation process have on a game. Some of the effects we measured are the territories evolution of players and player survival amongst others. Regarding the map interest we expect players to experience interesting challenges in a balanced game and that they do not recognize the basic patterns or any unnatural shapes due to our map creation method. For the dynamic process we want to verify that it does not affect the gameplay for players, and that they can develop and attain the same objectives as if they all started the game at the same time.

A. Map Interest

The results observed in this subject were very satisfactory. The challenges experienced in these maps were considered (we offered a scale of 5 values, from uninteresting to interesting) interesting by 53% of the players and mildly interesting by another 47% of them.

Regarding the player placement distribution only 20% of the players did not notice the way players were distributed throughout the map. Additionally, some players were actually able to recognize the way players were distributed. Nonetheless, 93% of the players did not consider the distribution to be “unpleasant”. The same percentage 93% of players did not find unnatural shapes in the generated maps.

Still based on the player’s feedback, 73% reported that they always felt they could change the course of events in the game. From the remainder none felt helpless in the beginning of the game. These results support the conclusion that our dynamic method was able to maintain game balance.
Figure 8. Dynamic Game II analysis of the territories evolution based on subscription turn.

B. Territories Evolution

From the two games we analyzed in depth we reached a similar conclusion that there is a very low, and variable amongst games, influence of the subscription turn of players and the number of terrains they can own after a fixed number of turns. In the graph from figure 8 we can examine the influence of the subscription turn in the number of terrains a player manages to obtain in 5, 10 and 15 turns after he joined the game “Dynamic Game II”. After 5 turns we can observe that the influence of the turn is very small, since we have a linear regression with a very low slope value. After both 10 and 15 turns the influence is still very low, even though when comparing to after 5 turns the sign of the slope changed. The evolution of territories observed in game “Dynamic Small-Cap I” provided us with results similar to the previous ones.

C. Parameterization and Aggressiveness

From the several areas we evaluated the one with the most conclusive data relation between the parameterization and the effects is the aggressiveness of the game. The two games we studied had an important difference regarding the player zone configuration. The Dynamic Game II had an NTS=1 while the Dynamic Small-Cap I had an NTS=0. This caused players in the later game to be closer to each other and have less terrains to conquer which originated more conflicts. We can observe this behavior in figure 9. The Dynamic Game II always has less personal conflicts during the period which we collected data in both games. It is then possible to conclude that the difference in NTS (in our case 0 or 1) has a deep influence regarding the game aggressiveness.

D. Scalability

The scalability is a very important property of our method as it enables a massively multiplayer context. Since our dynamic generation process is asynchronous we must have a way to persist the map generation data. This process plays a very important role since a big part of the player subscription time is due to the persistence of data. During our work we evolved our approach to this problem through several methods of persisting the generator’s state. While doing so we observed a corresponding performance increase regarding the player subscription times. These can be observed in figure 10 where we compare the three methods used. Here we can observe (by both individual times and their polynomial approximations) a considerable difference in performance amongst the different methods explored, where the best performance was achieved with a database persisted model.

VII. DISCUSSION

In this section we will discuss some of the implications and consequences of our approach in both the game designer’s part of game creation and the player’s perspective while playing.

With our approach the maps created are unbounded by any imposed limit on either the number of players or game design characteristics. With it the game designer has available a procedural way to create interesting maps that are scalable for the massively multiplayer browser games. He can focus only on the map balancing while creating...
different player experiences without intensive work. Some of the characteristics that can be modified to provide different experiences are the size of the player zones, the balance between PSS and NTS of the player zones, the PTC and terrain feature creation and control. Our method then provides a good balance between the advantages of the manually and procedurally created maps.

The terrain prototypes definition plays an important part regarding the balance between manual work and procedural generation. The initial prototype values used by our method were based on previously existent terrain definitions from handmade maps. The prototypes and the difficult process of their evaluation and categorization can lead to imbalances between the several game races since each one has different distinct characteristics. In the results chapter we briefly referred a mechanism to assist the game designer in this difficult task. This process was merely briefly explored but we think it can be further tested and explored due to its importance in validating the game designer’s prototype definition work.

The problem of delayed player entries is also addressed in a way which brings improvements in both balance and player experience. The method described has schemes that help diminish the imbalances created by the delay. However, since the problem is not completely eliminated we propose be put in place an initial protection period. Such a period with a duration of a few turns would enable new players to develop initially without the risk of being overrun by any older player, even if from just one turn older players. This would provide a safer initial resource development and give the new player time to prepare for the challenge presented by entering the game later than other players. Another solution to this problem would be to increase the strength of the players which enter later. This increased strength might refrain older players from immediately attacking recently added players or at least would enable the new player to have an opportunity in fighting the older player. However, this compensation scheme might create other problems, like older players’ insatisfaction with the fact that players which enter later are “benefited”. Moreover the balancing of such a system would be very hard and thorough player tests would have to be made. Some further discussion regarding this subject is presented in chapter VIII.

The schemes that help mitigate imbalances, specifically created by delayed player entries in a dynamic context, also have an effect on the way players see these maps. With our method the maps become evolving shapes which react to the flow of players’ subscribing the game by growing to properly accommodate the new players. This makes the map shape unpredictable (since player subscriptions are also unpredictable) reducing the map repetition because every map shape is different according to the number of players entering at a given game time. A map where more players enter at a given game time will be different from a map where fewer players are entering at the same game time. This promotes replayability since almost every map will be different from any of the previously created which originates different player experiences and avoids boredom from repetition.

The dynamic type of multiplayer game creates a different game experience for players where they play in an ever growing interesting map. However, for our approach to have the intended effect players must understand the nature of the map they are playing so they are not surprised or frustrated. This could happen when not yet generated map terrain appears in their views. The nature of our maps is that of a growing map and so expanding zones may contain ungenerated terrain for a short time. Therefore, players must understand that in a growing self-adapting map it is natural that for a few turns they can find terrain to where they cannot move or see because they have not been created yet. This can happen in two ways. The first is when a player is placed on the map, and some of the neighbour player zones have not been generated yet since the player enters in the map’s frontier of expansion. In this frontier the ungenerated player zones are the expansion points for new players. The terrain is then created in the following turns by either new players entering or by zone expirations, as a consequence players can see a terrain evolution for a few turns near them. A way to make this view updates have less impact on the player is to generate all the terrain for a player, but let him only start with his most important one, making him conquer or explore by himself the rest. The second is when a player explores the map to its borders. Note that in this case he only faces this if the map is still expanding and has not been closed yet. The evolution of this situation is then similar to first case where the difference is that it is caused by the player’s fast exploration. Once players understand this nature of a growing map they can fully enjoy the strategically rich multiplayer online browser game.

Regarding the results experimentally collected from the games which used the method developed we were able to observe some relations between the map growth parameterization and game properties and player feedback. It is important to notice we did not have as many test as we wanted (for a reinforced results analysis) due to the time it took to implement and integrate a stable version of our solution and the great amount of time each test game took to run its course from start to finish. Nonetheless, we were able to collect data which seem to confirm that we achieved our objectives of procedural and balanced map generation of interesting maps in both a classic and massively multiplayer online context. Additionally we were also able to measure more intrinsic effects of the developed method on gameplay, in parameters like the aggressiveness (measured with the amount of wars) of a game and the survival of players. Finally we stress the fact that more player tests are needed and they will be done as the game will be available to the general public.

VIII. Future Work

On top of our work there are still some areas that can be improved or explored.
As previously described, the map expands or contracts the size of the expansion zone and at a given point in time it can actually close itself not allowing more players to join. This can create an interesting player experience and even serve as a map size control method in case the game designers wish it. For example if a maximum number of players is set to the game, then the map will only expand until that number of players is reached and close itself without any direct input from the map creator (this control measure is implemented in Almansur). However, one might intend the map continues to expand indefinitely, even when the number of players joining is too low to keep zones from expiring. One way to enable this would be to create common neutral zones instead of impassible zones until new players joined. However, this approach imbalances the game, since if players do not enter for a very long period then one or several players end up having a large territory to explore and conquer only by themselves and acquire a great resource advantage over the others. How would it be possible to expand even when there are few players entering? Do we create poor terrain? Is it even really desirable? What are its consequences for the game? These are some of the questions which could be further explored in this area of dynamic map generation.

Regarding the map shape, the properties of the dynamic generation process can still be further explored to randomize it more. When the map grows rapidly we have many free player start places where hardly will all be occupied. Since the the growing schema tries to follow a spiral pattern, this commonly leads to some of the places being occupied in an orderly fashion and the others expire, creating a recognizable expansion pattern. In order to avoid this pattern and further randomize the map shape the generation process could be made aware of these situations and randomly expire some of the excessive free player start places.

On the balancing area there is also space for improvement. The player protection period would already be a good help, but could not we also have a development compensation system for players that enter later than others? Such a system could then eliminate disadvantages created by players entering at different game times and having different development levels at a given time. This could be achieved by creating terrains with different building levels or by assigning different amounts of initial terrain based on the already existent players on the map. But then how would other players feel about this? Possibly they would feel this measure as unjust, since they had had to work to acquire that development, and new players did not. And if one eliminates the disadvantages of entering later, why not wait until later to enter? Of course, this not intended in terms of game design. One wants players to enter as soon as they see and become interested by the game. A possibly more balanced solution would be to create a compensation system that offers less “strength” to the new player than the one achieved by the neighbouring players. This could however be hard to measure, even though the zone evaluation mechanism created could serve as a starting point for it.

Nonetheless, player tests would still be needed to validate it.

Regarding scalability (especially in a dynamic context) our method can also be improved. One of its main obstacles is the amount of data the generator has to persist. The more players subscribe the more data is persisted and slower the load/unload times of the generator are. During our dynamic generation process the generator keeps many information where some of it could be discarded for a better performance. For example it keeps a model of all the generated terrain where inner generated zones could be discarded for a better performance since they will not be used anymore. The decrease of the persistence times would lead to an even slower (than our current method) growth of player subscription time and increase the amount of players we could add.

IX. Conclusions

Our approach enables us to create interesting maps for turn-based strategy browser games in both a classic and massively multiplayer context. Our method can achieve these results due to its balance between the flexibility of the procedural generation method and the needed game designer input. For the massively context a dynamic map generation model was additionally needed in order to be able to adapt to the unpredictable nature of player subscriptions in online browser games and use a complex game environment in this context. The method developed also contributes to the solution of balancing problems created by delayed player entries and player unpredictability. Nonetheless, these problems are not eliminated and are 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, especially the massively multiplayer context. By using our method he benefits from the advantages of procedurally generating maps (ex. can quickly create several different maps) and also from some of the handmade maps (ex. create strategic situations with terrain features). Relatively to the handmade maps he cannot create historical games but themed games are still possible (ex. “The Orc War”) with a correct parameterization. However, this comes with an increased responsibility regarding the parameterization of the generation process since the intended player experience relies heavily on a good parameterization to ensure game balance. This means that all the parameters, especially the terrain prototypes, must be carefully adjusted for the game before hand, also meaning intensive game testing for a well balanced game.

From the player’s perspective the solution developed for the classic game context does not bring significant changes. However, on the dynamic context the players can now experience a complex game environments and with interesting maps. Additionally, the risk of players tiring from repetitive play is reduced since map generation depends on player entry pattern and choices resulting in very different game maps for each game.
Our case study was Almansur Battlegrounds and it provided a real game for us to apply the methods developed. After implementing the solution created in the game we were finally able to test our approach. From the results obtained we can conclude that we achieved our goals of balanced generation of interesting maps. Beyond this we were able to scale our map generation to the massively multiplayer context while maintaining balance and interest. For the classic game context is now provided with a procedural method that can even be further automated in order to automatically generate games as needed. For the dynamic context Almansur can now host multiplayer games where players can play against many others, do not have to wait for them in order to start playing and can always freely choose their race.

Using the techniques described in our work the TSBGs can evolve with their complex game environments while providing players with the flexibility player traits choice while still offering interesting and strategically challenging games.

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.

REFERENCES


