PSMAGE: Balanced Map Generation for StarCraft

Alberto Uriarte
Drexel University
Philadelphia, PA, USA
albertouri@cs.drexel.edu

Santiago Ontañón
Drexel University
Philadelphia, PA, USA
santi@cs.drexel.edu

***Abstract*—Designing a well balanced map for a real-time strategy game might be time consuming. This paper presents an algorithm, called PSMAGE, for generating balanced maps for the popular real-time strategy (RTS) game StarCraft. Our approach uses Voronoi diagrams to generate an initial map layout, and then assigns different properties to each of the regions in the diagram. Additionally, PSMAGE includes a collection of evaluation metrics, aimed at measuring how balanced a map is.**

I. Introduction

Real-Time Strategy (RTS) is a game genre where players need to build an economy (gathering resources and building a base) and a military power (training units and researching technologies) in order to defeat their opponents (destroying their army and base) and the players can perform actions at any moment (real-time). In this game genre, tournaments where players compete in one versus one games are increasingly popular. Usually these games provide a set of maps for multiplayer or tournament games, but the quantity of these maps is insufficient for the demands of the players. For this reason the fan community uses external tools to create more game content, in this case, more tournament maps. However, these tools require a significant amount of manual and time-consuming intervention, in order to ensure the resulting maps are balanced, and useful for tournament use.

In contrast to manual content production, Procedural Content Generation (PCG) refers to either fully or partially automated generation of game content. Designing an algorithm for PCG is a complex task for several reasons. For example, it is desirable that the algorithm allows a human user to influence the final content in order to avoid losing control over the design process. Also, in most cases a PCG algorithm must be able to automatically judge the generated content to filter out non-interesting or non-entertaining content. A significant amount of work can be found in the literature for generating different content types (bits, space, systems, scenarios, design, and derived), with a wide range of approaches. For a recent overview, the reader is referred to the work of Hendrikx et al. [1].

In this paper, we present an approach to generate balanced maps for RTS games. More specifically, we use a generative process to create maps suitable for tournament games, in the game StarCraft, where balance is key. We called our system PSMAGE (*Procedural StarCraft MAp GEnerator*). Additionally, PSMAGE incorporates a collection of evaluation metrics, that measure different properties of maps, and can help determine whether a given map is balanced or not.

This paper is organized as follows. The remainder of this section motivates our work and introduces our application domain, StarCraft. Then, Section II briefly describes related work in map generation, in order to provide context to our work. Section III states, in detail, which is the problem we are trying to address, and Section IV presents PSMAGE. After that, Section V presents a series of evaluation metrics used to evaluate the quality of maps, and Section VI describes our empirical results. The paper closes with a directions for future work.

A. Motivation

Creating balanced maps designed for competitive play can be time consuming for a designer, and difficult to do with conventional tools. A procedural map generation tool that can ensure maps resulting in balanced match ups between players can save a designer a significant amount of time, as well as guaranteeing a fair playing field for the players. For larger game studios, this can mean lower costs on newer content and for smaller studios, this could save time where resources are limited.

As the ease of procedurally generated map creation increases, so does the potential of non-professionals creating competitive maps. If the usability of a map creation tool is sufficient, the barrier of entry for designing maps is significantly lowered for gamers, not developers, to make their own content for practice, casual play with friends or competitions.

B. StarCraft

In this paper we use the game *StarCraft: Brood War* as the testbed for our research. StarCraft is an immensely popular RTS game released in 1998 by Blizzard Entertainment. StarCraft is set in a science-fiction based world where the player must choose one of the three races (*Terran*, *Protoss* or *Zerg*), gather resources, build bases, and train an army to defeat all of the other players. One of the most remarkable aspects of StarCraft is that the three races are extremely well balanced:

- *Terrans*, provide units that are versatile and flexible giving a balanced option between Protoss and Zergs.
- *Protoss* units have lengthy and expensive manufacturing process, but they are strong and resistant. These conditions makes players follow a strategy of quality over quantity.
- *Zergs*, the insectoid race, units are cheap and weak. They can be produced fast, encouraging players to overwhelm their opponents with sheer numbers.

978-1-4673-5311-3/13/$31.00 ©2013 IEEE

As in most RTS games, each player starts the game with an initial base (*starting point*) and needs to gather resources in order to build an economy to be able to produce more buildings and units. In StarCraft there are two types of resources: *minerals* and *Vespene gas*, initially distributed in the map. Initial base location, map geometry and the distribution resources, are all some of the key factors determining the strategy deployed by the players.

II. Related Work

Map generation techniques can be classified into two large groups: *constructive* approaches, and *generate-and-test* approaches. Constructive approaches are characterized by algorithms that are carefully designed to generate (in a single attempt) maps that have a desired look or structure. Generate-and-test approaches, on the other hand, consist of a generating component (that can generate different maps), and an evaluation component, which assesses the quality of each generated map, in order to select the best one. This section presents a brief overview of techniques using both approaches, focusing on those that have been designed for outdoor map generation, since those are more closely related to our work, which is generating maps for strategy games.

Constructive approaches to outdoor map generation typically work by first generating a *heightmap*, which is a two-dimensional matrix storing the height of every cell of a terrain, and then generating additional elements, such as water, or vegetation on top of it. Some of the most common methods to generate heightmaps are using noise, fractals, Voronoi diagrams or even a combination of the previous ones. For noise, the usual option is Perlin noise [2]. In the case of fractal approaches, good results have been accomplished by using the diamond-square algorithms [3]. Olsen presented a modification of the diamond-square (smoothed midpoint displacement) in combination with Voronoi diagrams [4], with improved results. Given a heightmap, the generation of bodies of water can be easily solved using flooding algorithms or defining a fixed water height level. But more sophisticated approaches have been used, such as fractal models [5] or water flow and erosion simulations [6].

Some other approaches give the designer a finer control of the process, like declarative modeling and interactive procedural sketching [7], procedural brushes [8] or software agents for individual tasks [9]. Moreover, except for a few notable exceptions, like [10], [4], constructive approaches have not been explored in depth for the specific problem of generating maps for strategy games.

Existing work on the second group of approaches, generate-and-test, can be divided in two groups: *Search-Based Procedural Content Generation* (SBPCG) [11] and *Multiobjective Evolution algorithms* (MOEA). In SBPCG a generated candidate content is evaluated on one numeric dimension and a search algorithm is used to explore the search space, looking for content that maximizes the evaluation (or "fitness") function. Frade et al. presented a genetic programming approach using an "accessibility" fitness function to evolve the heightmap [12]. Another typical fitness function family in the context of evolutionary approaches are those based in paths: Sorenson and Pasquier used a function that measures whether a traversable path from start to finish location exists [13]; Ashlock et al. maximize the distance between locations by placing obstacles (walls) in the map [14]. On the other hand, the idea in MOEA is to use more than one evaluation or fitness functions, aiming at finding Pareto front of Pareto-optimal solutions. Togelius et al. presented an application of this idea to generate strategy maps [15].

Another important aspect to consider for strategy map generation is how well the map is balanced. To the best of our knowledge, there is little work in this area. Togelius et al. [15] used some fitness functions to evaluate the maps produced and some of those functions are related to map balance. Lara-Cabrera et al. [16] analyzed the balance of generated maps for *Planet Wars* by running tournament games and promoting those games where players had similar planets and ships, and also those that took a long time to complete. A closely related problem is how to evaluate map quality. Olsen provided some additional functions, such us low average height and a high standard deviation for slope, to compute a game suitability score, and used it to evaluate generated terrains [4]. Perkins developed a terrain analysis library (BWTA) to analyze the space partitioning (detecting regions, choke points and base locations) in RTS maps [17]. The SC2 Map Analyzer[1] provides some spatial analysis and a graphical way to determine positional imbalance. More recently, Reddad and Verbrugge presented an approach to geometric analysis of strategy maps [18].

III. Problem Definition

This paper addresses the problem of procedural map generation of strategically interesting and balanced maps for strategy games. Specifically, we will use StarCraft as our application domain.

For the purposes of this paper, we will define a *balanced map* as one that satisfies two conditions: a) if all the players have the same skill level, they all have the same chances of winning the game, and b) in the case of StarCraft, no race has a significant advantage over any other race. And we will define a *strategically interesting* map, as one where: a) there is a significant number of strategies with which a player can win the game, and b) there is no dominant strategy. For example, maps that are too small might not be strategically interesting, since they only allow for early-game strategies.

A main assumption behind our work is that, assuming that the races are already perfectly balanced (i.e. there are no specific exploits that benefit a certain race), balanced maps are those where certain "strategic locations" in the map are distributed equally for all players. In other words, balanced maps are those in which all players have equal access to these strategic locations.

We will consider two different types of strategic locations: *regions* and *choke points*. Usually strategy game maps can be divided into a set of different regions connected through *choke points*. Intuitively, regions correspond to larger open areas, whereas choke points correspond to narrow passages that connect each of these areas. Regions can be characterized by the following properties:

[1] http://www.sc2mapster.com/assets/sc2-map-analyzer/

- *List of choke points*. Regions with more than one entrance are more difficult to defend. And, consequently, regions with only one choke point are harder to conquer.
- *Resources*. Areas in the map that contain resources are important to decide where to place a new base and start gathering resources. The strategic value depends on the amount of resources and the variety of resources.
- *Openness*. Measures the shape of a region, in relation to how easy a region is to defend or attack. For instance "thin" regions give an advantage to strong big units while "open" regions are better for weak small units. In our work, we compute the "openness" as the maximum value of the *distance transform* of the map in a given region.
- *Area*. The size of the region, which determines, for example, the size of a base that could be constructed in such area.

Choke points are characterized by the following properties:

- *Width*. Narrow choke points are easy to defend and perfect for ranged combat units, while wide choke points are more suitable for big confrontations and melee[2] units.
- *Ramp*. If the choke point is between regions of different height then it is a ramp. This can be strategically important in games where units in high ground get an attack or defense bonus like in StarCraft. PSMAGE does not generate ramps in its current version, but this is part of our future work. This means that in the generated maps, the regions with different elevations are not connected, but accessible by air transport. Moreover, although we do not generate ramps, we will consider them in our later balance metrics since we will analyze human made maps with ramps.

In our particular application domain, StarCraft, each player starts in a single location, called the *starting point*. Some of the regions of the map are selected as the starting points for different players. Therefore, in order to achieve balanced maps, we propose to generate maps for which the distributions of different types of regions and choke points are the same from each of the starting locations.

In addition to achieving a good balance, map geometry also affects the different strategies that players can employ. Small maps with few regions imply early conflict, which results in short games, where players are only able to exploit a subset of game mechanics. Maps whose structure forces the conflict is overly postponed, allow players to deploy a wider range of strategies. For instance, in StarCraft, it is considered that each player should be able to expand 3 times before conflict becomes inevitable in order for a map to generate interesting gameplay.

In Section V we will provide some evaluation metrics aimed at analyzing the balance of a map for strategy games, based on the ideas laid out in this section.

[2]Close range combat unit.

IV. Procedural Map Generation for StarCraft

The process that PSMAGE follows to generate balanced maps for StarCraft can be roughly divided in the following six steps:

1) **Region generation**: which generates the base layout of the map.
2) **Determine elevation**: determines the elevations of all the regions.
3) **Starting position placement**: assigns some regions as the starting positions for players.
4) **Addition of base locations**: adds resources to some of the regions of the map, making them potential regions for additional bases for players.
5) **Map symmetry**: through the use of symmetries, generate a final map likely to be balanced.
6) **Realization**: the map is translated into an actual StarCraft map.

Each step shall now be described in detail. For illustrative purposes, the result of each step will be shown.

A. Region Generation

As already described above, a strategy map can be seen as a collection of regions connected (or not) between them. In PSMAGE, we use an approach based on Voronoi diagrams [19] to generate such regions. A Voronoi diagram is a way to divide a given space in a set of regions, based on the proximity of each point in the space to a series of *seed points*.

Since PSMAGE generates maps for StarCraft, each map is a rectangular grid composed of $w \times h$ cells. Specifically, given a desired map size, the process works as follows:

1) Given a desired number of seed points, n (that can be specified by the user), PSMAGE generates n points in the map by using Poisson disk sampling [20]. This technique just needs an input parameter d_{min} (the minimum distance between the seeds) results in well spaced seed points in the map and it avoids areas of the map with a very high density of seed points.
2) Fortune's Algorithm [21] is used to generate the Voronoi diagram. This algorithm uses the idea of sweep line algorithms to efficiently compute the Voronoi diagram in only $O(n \log n)$ time complexity.
3) Finally, we need to clip the Voronoi diagram with the size of the desired map in order to ensure all the regions are closed. We can see the results of these three steps in Figure 1, notice that a level designer can decide the number of seeds n (which corresponds to the number of regions), the minimum distance d_{min}, and the size of the map $w \times h$.

B. Elevations

Each of the regions in RTS games might have different properties. In the particular case of StarCraft, walkable regions have a given *elevation*, which can be: *normal ground* or *high ground*. In its current form, PSMAGE cannot generate maps with non-walkable regions, such as *water* or *space*, but that is part of our future work.

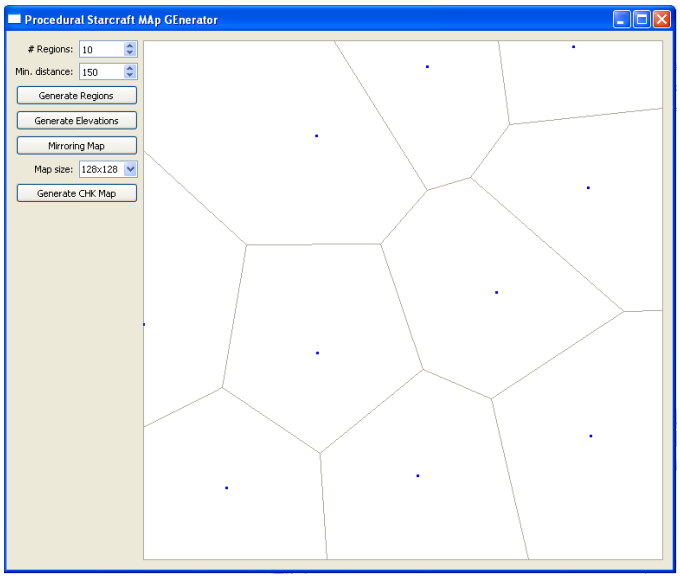

Fig. 1. Regions generated by PSMAGE using a Voronoi diagram with $n = 10$ seed points and $w = h = 128$.

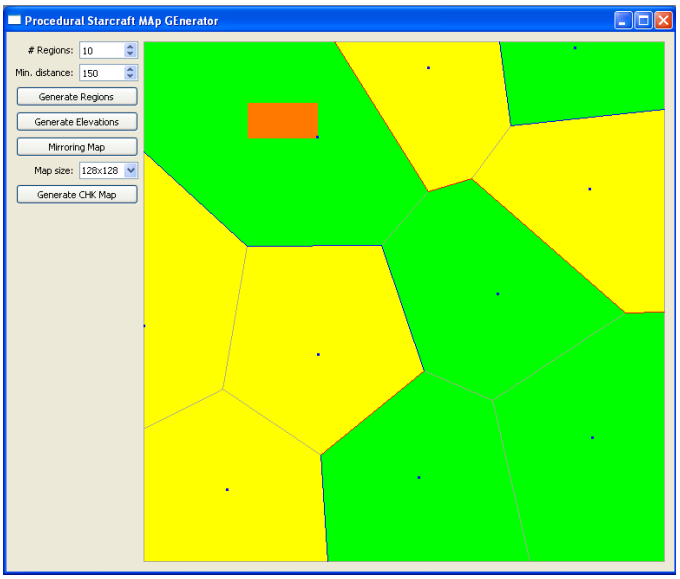

Fig. 2. Regions with elevations (green regions are normal ground and yellow regions are high ground) and starting location in orange.

PSMAGE just assigns a random elevation, respecting a user defined parameter α, representing the percentage of regions to be considered as high ground. Considering other forms of assigning elevation, and also adding the capabilities to define ramps between areas of different elevation is part of our future work. Specifically, we will experiment with local search methods (such as evolutionary approaches) in order to determine an elevation distribution that optimizes the evaluation metrics presented in Section V.

C. Starting Locations

The next step is to determine which are the regions where each of the players will start. As we will elaborate in Section IV-E, PSMAGE generates maps by using symmetries, so, we only need to determine one starting position; the starting positions for the other players will be generated automatically via symmetries.

PSMAGE determines which are the k top-left-most regions in the map ($k = 4$ in our experiments), and selects one of them at random as the starting position.

At this stage, PSMAGE needs to validate two conditions: First, the openness of the starting location must be bigger than a certain threshold. And secondly, a path between the starting location and at least one of the regions on the right border and one region on the bottom border must exist. This second condition ensures that there will be paths between the starting positions of all the players, once the final symmetry step is performed.

If either of the previous two conditions is not met, the process restarts from Step 1, and a new map is generated. Figure 2 is a valid configuration of a region with elevation and starting location already determined.

D. Base Locations

As we mentioned before, one of the important properties of regions are the resources they hold. In order to achieve balanced maps, one of our objectives is to distribute those resources equitably. Each RTS game have different types of resources and different densities; for instance a mining spot can be mined during a long period of time until exhaust the mineral, while a tree will disappear after a few chops. In StarCraft there are only two types of resources: *minerals* and *Vespene gas*. Both of them are dense (i.e. they take a significant amount of time to deplete). The approach used by PSMAGE is to place those resources following a pattern (typically used in all StarCraft maps) that is good for letting players create new *base locations*.

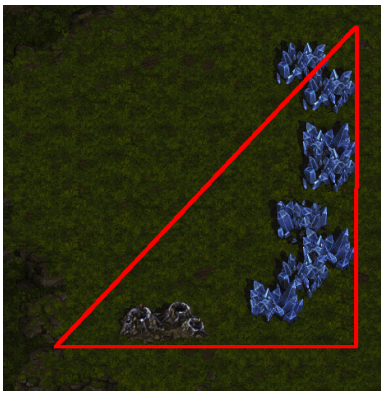

Fig. 3. StarCraft pattern in resource allocation.

A base location is a good place to locate the building to which resource gatherers need to bring the resources (e.g. a *Command Center* in StarCraft, if we are playing with the *Terran* race). A base location should be equidistant from all near resources. The pattern PSMAGE uses places a line of eight mineral spots, and optionally a gas geyser perpendicular to the mineral line (as shown in Figure 3). The minerals and the gas geyser form a right triangle, which right angle is typically located far from the region entrance.

As we explained before, the number of potential base

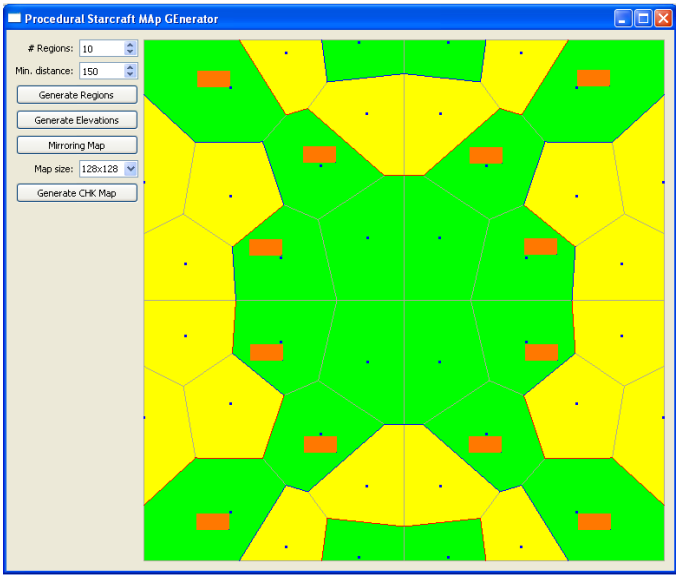

Fig. 4. Final abstract map with symmetries.

locations affects the gameplay of the map. Therefore for each player starting location we need at least two potential additional base locations to promote a delayed conflict to give time to develop any possible strategy of the game. Taking this into account we defined, PSMAGE stochastically selects a number of regions $r \geq 2$, but always smaller than βn, where $0 < \beta \leq 1$ is a user defined parameter (recall that n is the number of regions).

E. Map Symmetry

Many strategies can be used in order to obtain procedurally generated balanced maps. PSMAGE uses an approach based on symmetries. A symmetrical map is made up of two or four identical (bar mirroring) parts, facing each other. And this strategy is also used in competitive maps for StarCraft as we can see in Figure 6. Some map designers masquerade this symmetry by placing different type of objects in symmetry to have the same functionality (for instance, blocking a path). This is called *functional symmetry*.

The reason to use symmetries is because we want to generate tournament maps. The StarCraft community has been mastering the creation of this kind of maps since the releasing of the game in 1998. The high competitiveness in these tournaments makes map makers to extremely care about balance, and the best way that they find it is creating symmetric maps to give fair opportunities to win to each player. Our first step then is to try to reproduce these professional maps, analyze the balance properties. As part of our future work, we want to explore how to keep the balance properties while breaking some of the symmetries.

In our algorithm we used a mirror symmetry in the center of the map. PSMAGE mirrors the map generated so far, obtaining four big areas each one symmetric to the others. The final look is showed in Figure 4.

F. Realization

The last step is to transform this 2D abstract map into a StarCraft map. Although a StarCraft map is also a 2D map, it is tile-based, and the tiles follow a diamond pattern because the pixel art simulates an isometric perspective. Therefore, we need to transform our square grid into a diamond grid. Since the resulting diamond grid is an approximation of the original one, some inaccuracies from the original abstract map can arise. The fact that the StarCrat map format (.CHK) is a closed source file format generated significant technical difficulties for generating the final map. We used some reverse engineering studies[3] to generate the binary file in order to be able to actually use our generated maps in StarCraft.

G. Summary

Using the taxonomy proposed by Togelius et al. [11], our map generation method can be categorized as an *offline* method, since it takes place during game design. The content generated is *necessary* to play tournament games. It uses a combination of *random seeds* and *control vector* where a game designer can decide some features (such as map size, number of regions and the percentage of elevated regions). It is *stochastic* because given the same features the output is unpredictable. And it is mainly a *constructive* approach although in section IV-C we follow a *generate and test* schema with a fitness function.

In our experiments we used the following user inputs: 128×128 map size in *tiles* (4352 in pixels), $n = 10$ seed points, $d_{min} = 150$ as a minimum distance between seeds, $\alpha = 50$ as the percentage of elevated regions, and $k = 4$ top-left-most regions to consider to place a starting location.

V. Map Balance Analysis

Once a map has been generated, we are interested in automatically analyzing how balanced and how strategically interesting the map is, in order to check if it is suitable for a tournament game. To this end, this section presents a collection of evaluation metrics to measure different aspects of a map. Notice that since PSMAGE uses symmetries to generate maps, some of these functions, aimed at assessing how balanced maps are, will always return very high scores. However, they are useful to compare the maps generated by PSMAGE with maps generated by other approaches.

Togelius et al. defined a collection of fitness functions for the quality of StarCraft maps [15], and Mahlmann et al. presented similar functions in their approach [22]. Below, we present a collection of functions that contain modified versions of those by Togelius et al. and also a series of new functions we defined in order to ensure balanced maps for tournament competitions.

- f_{1a}, f_{1b}: *Starting Location Space*. Each starting location must provide a similar terrain space for placing buildings. That means that we want to minimize the standard deviation of the *openness* and *area* of all the starting location regions. More formally, $f_{1a}(m) = \sigma(A_{sl})$, i.e. the standard deviation of the area of

[3] http://quantam.devklog.net/CHKFormat.htm

the starting location polygons in the map m. And $f_{1b}(m) = \sigma(O_{sl})$, i.e. the standard deviation of the openness of the starting locations in the map m. The openness is calculated taking the maximum value of the *distance transform* of the map in a given region.

- f_{2a}, f_{2b}: *Starting Location Spread*. In order to ensure a fairness distance between each starting location, we want to minimize the standard deviation of the shortest ground distance (using A^*) and the shortest air distance (using Euclidean distance) between each pair of starting locations. Then, $f_{2a} = \sigma(A^*_{sl})$, where $\sigma(A^*_{sl})$ is the standard deviation of all the ground distances between each starting location; and $f_{2b} = \sigma(E_{sl})$, where $\sigma(E_{sl})$ is the standard deviation of all the air distances between each starting location.

- f_3: *Base Expansion Accessibility*. As we showed before, in order to promote all kind of strategies in a map, we have to ensure that each player can expand (establish in a new base location) two times before the conflict becomes inevitable. We want to minimize the standard deviation of the ground distance from each starting point to the closest base ($\sigma(B1_{sl})$), and the ground distance from each starting point to the second closest base ($\sigma(B2_{sl})$). Then $f_3 = \sigma(B1_{sl}) + \sigma(B2_{sl})$.

- f_4: *Base Location Distribution*. Here we want to evaluate the fairness of the base location distribution in the map. This distribution must be equitable for all starting positions. Thus, we need to minimize the standard deviation of the standard deviation of the shortest ground distance from one starting point to all the base locations in the map. More formally, $f_4(m) = \sigma(\{\sigma(Bn_{sl})\}_{n=1...4})$, where $\sigma(Bn_{sl})$ is the standard deviation of the ground distances between starting location n and all the base locations in the map m.

- f_{5a}, f_{5b}: *Choke Points Symmetry*. Choke points are one of the most important strategic locations in RTS game maps. Most of the combats happen in these locations. So we have to be sure that there is not a starting location that has "better" choke points than others. The concept of "better choke point" can differ from game or from race, so a good metric to ensure this is to look the symmetry of this strategic positions. To quantify this, first we build the sorted list of choke points present in the shortest ground path between two start locations S_1 and S_2, $CPL_{1,2} = \{CP_0, CP_2, ..., CP_{q-1}, CP_q\}$. Then for each symmetric choke point we compare the ground distance to the closest start locations and the properties of the choke points (width and ramp). To compute this we used the following formula:

$$
\begin{aligned}
j &= \lfloor CPL.size/2 \rfloor \\
q &= CPL.size \\
X_d(CPL) &= \{dist(S_1, CP_1), ..., dist(S_1, CP_j)\} \\
Y_d(CPL) &= \{dist(S_2, CP_{q-j}), ..., dist(S_2, CP_q)\} \\
\sigma_{dist}(CPL) &= \sqrt{\mathrm{E}[(X_d(CPL) - Y_d(CPL))^2]} \\
X_w(CPL) &= \{CP_1.width, ..., CP_j.width)\} \\
Y_w(CPL) &= \{CP_{q-j}.width), ..., CP_q.width)\} \\
\sigma_{width}(CPL) &= \sqrt{\mathrm{E}[(X_w(CPL) - Y_w(CPL))^2]}
\end{aligned}
$$

Then $f_{5a} = \sigma(\{\sigma_{dist}(CPL_{x,y})\}_{x=1...4,y=1...4})$, where $\sigma_{dist}(CPL_{x,y})$ is the standard deviation of the distance for the pair of starting points x and y. And $f_{5b} = \sigma(\{\sigma_{width}(CPL_{x,y})\}_{x=1...4,y=1...4})$, where $\sigma_{width}(CPL_{x,y})$ is the standard deviation of the width for the pair of starting points x and y.

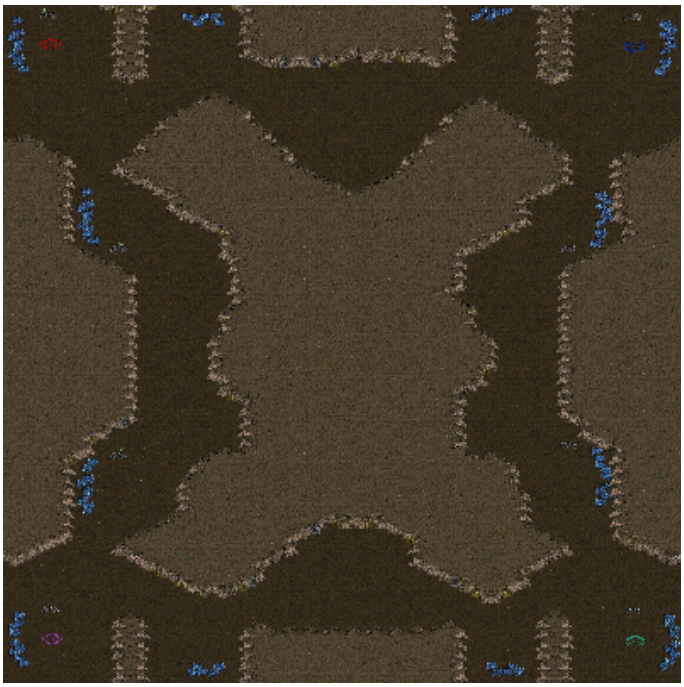

Fig. 5. Procedural map generated by PSMAGE.

VI. Experimental Evaluation

In this section we give a brief description of how the algorithm was implemented. And we show our experiments comparing the maps generated by PSMAGE and some of the maps used in the International Cyber Cup (iCCup)[4], which organizes different international StarCraft tournaments.

A. Implementation

PSMAGE has been implemented in C++ and it uses the Qt framework to provide a graphical user interface where designers can interact with the variables of the algorithm.

[4]http://www.iccup.com/starcraft/

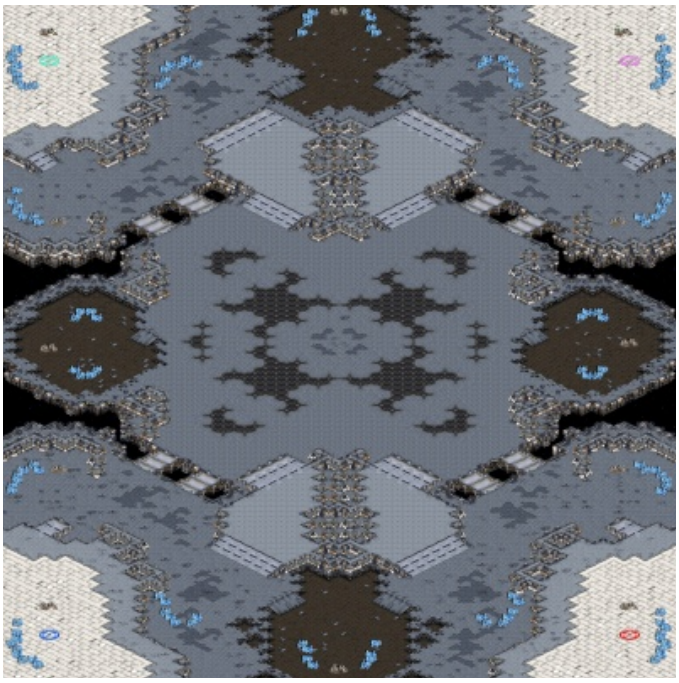

Fig. 6. The *Circuit Breaker* tournament map.

TABLE I. RACE STATISTICS IN *Circuit Breaker* TOURNAMENT MAP.

Game	*Won*	*Lost*	*% Won*
Terran vs Zerg	63	57	52.5%
Zerg vs Protoss	59	55	51.8%
Protoss vs Terran	71	63	53.0%

We are planning to release PSMAGE as an open source code to the research community. Additionally, we extended the functionalities of the BWTA library, implementing our proposed balance analysis.

B. Experimental Results

In this section we present a balance comparison, using our proposed evaluation metrics, between a map generated by PSMAGE (Figure 5), a set of well balanced tournament maps created by the StarCraft community, and a map bundled with the original StarCraft (*Nightfall*), which is balanced, but not as well balanced as the tournament ones. Figure 6 shows an example of a tournament map (*Circuit Breaker*). We would like to emphasize that this tournament map is one of the most well balanced StarCraft maps ever designed. This is reflected in Table I, showing win ratios for different races in this map in tournament games. Analyzing these numbers we can observe that the standard deviation between the win ratios and a perfect balance (50% win ratio) is only 2.48. These statistics are gathered by the website TeamLiquid[5].

Table II shows the results for the proposed balance metrics with 11 tournaments maps. The tournaments maps selected are those where the win ratios between the different races have a standard deviation smaller than 6, i.e. the most balanced maps between races. As we can observe, most of the metrics, when evaluated in PSMAGE's map are less than on standard deviation away from the mean obtained in the tournament maps (marked in bold in the table). Moreover, in most of the cases the results of PSMAGE's map are less than the average of tournament maps. Only the metric f_{2a} (A* distance between starting points) has worst results. Now if we compare the results of tournament maps and the first multiplayer maps bundled with StarCraft like *Nightfall*, we can see how *Nightfall* is worst than the average in most of the metrics. Now, comparing PSMAGE with *Nightfall* we obtained better scores in all metrics but f_{2a}, f_{2b} and f_3. This indicates that the maps generated by PSMAGE are very well balanced according to the metrics being used. We believe that the errors introduced in the step of converting the abstract square grid into a diamond grid introduce some deviations in PSMAGE's maps, worsening the values obtained in some of the metrics (specially for f_{2a}). But in spite of this, our generated maps are comparable with the tournament ones in these metrics.

We did not observe any correlation between the game statistics and the proposed balance metrics. The main reason is due the noise in the game statistics and the metrics. In order to evaluate empirically if a map is well balanced we need game statistics between players with the same skill level. And this is really difficult to have as it is really difficult to measure precisely the skill level of a human. In the other hand some of our proposed metrics are sensitive to BWTA errors. It has been shown that the algorithm that BWTA uses to decompose the maps in regions and choke points has some false positive and false negative issues in some maps. Given the issues discussed previously, we cannot conclude any significant correlation between game statistics and balance metrics.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented PSMAGE, an algorithm to generate tournament StarCraft maps procedurally. We defined some constraints in the process and a control vector to give some control to map level designers. We also presented some metrics to evaluate the balance of a given tournament map. Using those metrics we showed that the PSMAGE's maps are comparable with those made by humans for a tournament game propose, therefore PSMAGE could be used in the future for completely automated tournament map generation and/or to assist human map level designers.

Moreover, there is still a significant room for improvement. For example, the inclusion of non-walkable regions has been left for future work. Additionally, we would like to include the possibility to connect regions of different levels using *ramps*. Also, in its current state, PSMAGE does not generate decoration of the generated map. We are currently experimenting with noise in the tilesets and with statistical models to generate *clutter* based on previous existing maps.

Finally, although we would also like to experiment with additional types of symmetries, a big challenge is to skip the symmetry step altogether, while still keeping the map balanced. The map balance analysis presented in this paper constitutes one step towards this direction. As part of our future work, we would like to incorporate these metrics more tightly into the map generation process (for example, as fitness functions in some of PSMAGE's steps), and use, generate-and-test approaches to generate balanced maps that do not look symmetric to the naked eye.

[5]http://www.teamliquid.net/tlpd/korean/maps/404_Circuit_Breaker

TABLE II. EVALUATION METRIC COMPARISON BETWEEN PROCEDURAL MAP AND TOURNAMENT MAP, LOWER IS BETTER. MARKED IN BOLD THE RESULTS THAT ARE LESS THAN ONE STANDARD DEVIATION AWAY FROM THE MEAN OBTAINED IN THE TOURNAMENT MAPS.

Map	f_{1a}	f_{1b}	f_{2a}	f_{2b}	f_3	f_4	f_{5a}	f_{5b}
Circuit Breaker	12,995.60	1.00	530.86	684.35	50.65	6.17	40.28	3.21
Python	21,287.10	0.83	458.33	1,013.16	220.35	51.74	110.24	16.23
Fighting Spirit	24,182.50	2.28	458.20	687.44	146.39	10.12	216.22	2.31
Icarus	42,302.60	0.71	460.77	511.10	148.26	13.94	251.86	92.86
Colosseum II	26,959.30	4.90	339.01	661.16	160.49	23.56	190.83	68.16
Fantasy II	45,958.70	3.45	612.45	765.91	389.93	174.63	436.60	39.04
Nostalgia	41,331.80	2.05	349.56	690.19	184.26	19.05	145.31	15.45
Luna	141,193.00	4.87	551.02	594.24	527.09	102.91	231.08	138.18
Rivalry	36,199.60	2.18	475.35	673.02	130.48	33.80	162.32	49.72
Lost Temple	96,894.50	2.38	758.35	1,101.78	435.76	188.52	227.14	54.36
Desert Lost Temple	60,454.20	2.59	769.01	1,101.38	593.61	199.99	186.25	27.33
Average	49,978.08	2.47	523.90	771.25	271.57	74.95	199.83	46.08
Standard deviation	37,890.48	1.44	142.55	204.62	182.25	77.40	99.44	41.50
PSMAGE's map	**60,5589.70**	0.00	1,313.33	**695.10**	**130.99**	**8.95**	**283.49**	**30.49**
Nightfall	107,232.00	**1.79**	1,141.23	**574.00**	82.89	**21.24**	357.83	98.09

REFERENCES

[1] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 9, no. 1, pp. 1:1–1:22, Feb. 2013.

[2] X. Pi, J. Song, L. Zeng, and S. Li, "Procedural terrain detail based on patch-lod algorithm," in *Technologies for E-Learning and Digital Entertainment*, 2006, pp. 913–920.

[3] G. S. P. Miller, "The definition and rendering of terrain maps," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '86, 1986, pp. 39–48.

[4] J. Olsen, "Realtime procedural terrain generation," *Department of Mathematics And Computer Science IMADA University of Southern Denmark*, p. 20, 2004.

[5] S. T. Teoh, "Riverland: An efficient procedural modeling system for creating realistic-looking terrains," in *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*, 2009, pp. 468–479.

[6] P. Krištof, B. Beneš, J. Křivánek, and O. Šťava, "Hydraulic erosion using smoothed particle hydrodynamics," *Computer Graphics Forum (Proceedings of Eurographics 2009)*, 2009.

[7] R. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra, "Integrating procedural generation and manual editing of virtual worlds," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010, pp. 2:1–2:8.

[8] G. J. P. de Carpentier and R. Bidarra, "Interactive gpu-based procedural heightfield brushes," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, ser. FDG '09, 2009, pp. 55–62.

[9] J. Doran and I. Parberry, "Controlled procedural terrain generation using software agents," *IEEE Transactions on Computational Intelligence and AI in Games*, pp. 111–119, 2010.

[10] S. Shoemaker, "Random map generation for srategy games," in *AI Game Programming Wisdom, Vol. 2*, S. Rabin, Ed. Rockland, MA, USA: Charles River Media, Inc., 2004, pp. 405–412.

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

[12] M. Frade, F. F. de Vega, and C. Cotta, "Evolution of artificial terrains for video games based on accessibility," in *Proceedings of the 2010 international conference on Applications of Evolutionary Computation - Volume Part I*, 2010, pp. 90–99.

[13] N. Sorenson and P. Pasquier, "Towards a generic framework for automated video game level creation," in *Proceedings of the 2010 international conference on Applications of Evolutionary Computation - Volume Part I*, 2010, pp. 131–140.

[14] D. Ashlock, T. Manikas, and K. Ashenayi, "Evolving a diverse collection of robot path planning problems," in *Proceedings of the Congress On Evolutionary Computation*, 2006, pp. 6728–6735.

[15] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis, "Multiobjective exploration of the starcraft map space." in *CIG*. IEEE, 2010, pp. 265–272. [Online]. Available: http://dblp.uni-trier.de/db/conf/cig/cig2010.html#TogeliusPBWHY10

[16] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva, "A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars," in *EvoApplications*, 2013, pp. 274–283.

[17] L. Perkins, "Terrain analysis in real-time strategy games: An integrated approach to choke point detection and region decomposition." in *AIIDE*, G. M. Youngblood and V. Bulitko, Eds. The AAAI Press, 2010. [Online]. Available: http://dblp.uni-trier.de/db/conf/aiide/aiide2010.html#Perkins10

[18] T. Reddad and C. Verbrugge, "Geometric analysis of maps in real-time strategy games: Measuring map quality in a competitive setting," GR@M: Games Research At McGill, School of Computer Science, McGill University, Tech. Rep. GR@M-TR-2012-3, sep 2012.

[19] F. Aurenhammer, "Voronoi diagrams – a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[20] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions," in *SIGGRAPH 2007*, 2007.

[21] S. Fortune, "A sweepline algorithm for voronoi diagrams," in *Proceedings of the second annual symposium on Computational geometry*, ser. SCG '86. New York, NY, USA: ACM, 1986, pp. 313–322. [Online]. Available: http://doi.acm.org/10.1145/10515.10549

[22] T. Mahlmann, J. Togelius, and G. N. Yannakakis, "Spicing up map generation," in *EvoApplications*, 2012, pp. 224–233.