NORTHWESTERN
UNIVERSITY

Electrical Engineering & Computer Science Department

Technical Report
NU-EECS-12-02
March 30th, 2012

Qualitative Exploration in Freeciv

Christopher Blair

Master's Project Report
Christopher K. Blair

1.0 Abstract

Game artificial intelligence should be able to handle unknown environments intelligently and without omniscience because this leads to unrealistic play. Improving the ability of a game AI to explore an unknown environment is important because pre-made maps are not always available. The open source game Freeciv was used as a testbed for comparing two exploration algorithms. Using knowledge of the terrain and qualitative representations to guide the exploration led to markedly improved game play. Qualitative representations of the terrain created useful heuristics that increased the ability of the AI to play a game of Freeciv.

1.1 Keywords

FreeCiv, Exploration, Frontier Based Exploration, Game Artificial Intelligence, Qualitative Terrain Representation, Qualitative Terrain Reasoning

2.0 Introduction

Game artificial intelligence is best able to navigate known environments with tightly controlled parameters. Navigating an almost completely unknown environment is far more difficult. The computer must decide what direction to explore that will most likely lead to a payoff. In this paper I present two exploration algorithms in the context of a game of Freeciv. One algorithm is naive about the terrain and simply tries to explore as rapidly as possible. The other algorithm leverages information about the terrain to prioritize where to explore. I show that prioritizing exploration objectives improves the play of the computer at Freeciv.

3.0 Related Work

Exploration is an active topic in artificial intelligence, especially in robotics and game AI. Mohammad Al-Khawaldah et al. (2010) used a frontier-based exploration algorithm in the context of NetLogo. Frontier-based exploration is a technique where the computer tracks all parts of the map that are on the edge of the unknown, hence the name “frontier.” Al-Khawaldah used this Frontier algorithm with two explorers on a grid map. The algorithm calculated the desirability of exploring a tile by scoring how many unknown adjacent tiles there are, how far the tile was from the explorer, and how far the tile was from the second explorer. Al-Khawaldah used weights for each of the scoring parameters to optimize the time it took for the explorers to map out all of the unknown terrain.

Another approach to exploration was presented by Soule et al. (2009). Their problem involved two kinds of units: scouts and investigators. Investigators were slow moving, but required to mark special tiles as “investigated.” Scouts could move quickly and identify the special tiles for the investigators to go to. This kind of arrangement, of a fast knowledge gathering unit and a slower working unit, comes up in the real world too and the problem was chosen for this reason. The authors of this paper used genetic algorithms to tackle the problem, and each unit would have slightly different strategies based on the genetic algorithm.

My approach to exploration is superior to either of the above algorithms because my algorithm uses a qualitative representation of the map in order to prioritize where to explore. This gives it an advantage in finding important locations earlier in the game.

For this paper, I decided to directly compare my qualitative algorithm with the Frontier algorithm. This is because the NetLogo world of the Frontier algorithm is more compatible with Freeciv. The world of the scouts and investigators algorithm has some similar concepts to Freeciv, but unfortunately the role of Investigator does not correlate directly to anything in Freeciv. The closest unit to an Investigator in Freeciv is the Settler unit which is responsible for founding new cities. However, Settlers do not last the duration of the game because they are destroyed upon founding a city. This is not a trivial difference, and for this reason I decided to focus instead on the Frontier algorithm.

4.0 Exploration Problem

The most ideal way to allow a computer to navigate an unknown world would be to give the computer a complete map beforehand. This way everything is known, and the computer can simply path-find whenever necessary. Sometimes complete maps are available to give to the computer, but in games this might not be desirable. Part of the appeal of a game like FreeCiv is the novel experience of each game session, and playing on a random map facilitates this. Also, a computer that knows the complete map will play unrealistically because it will always attempt the most advantageous moves, according to its evaluation function. For autonomous robots, a detailed map may not even be available. One can imagine a robot operating at the scene of a disaster where debris and obstacles are unknown, or a scientific rover on another planet whose topography is only known generally. In these cases, the agent will have to make decisions based on limited information about where to explore to achieve its objectives. The agent will also have limited time, either a robot with limited battery power or a game AI competing with an opponent, to prioritize where to explore. Some kind of heuristic based on the world the agent is in will be needed to make intelligent choices.

5.0 Architecture of the FAP & Companions

The Freeciv AI Player (FAP) is a program, originally developed by Phil Houk (2004), which interacts with a Freeciv server to send game commands. The program has a minimal amount of the rules of Freeciv built into it, and relies on the server to enforce all rules. The program can make all legal moves that a player could and has no special access to information beyond what a human player would have. The FAP uses the Qualitative Reasoning Group's reasoning system FIRE to store knowledge and strategies. FIRE is a reasoning engine which supports general purpose reasoning over large knowledge bases (Forbus et al., 2010a). Companions is a cognitive architecture which supports analogical processing and has a distributed architecture (Forbus et al., 2010b). It functions as a layer above the FAP which can make multiple instances of the FAP agent in order to play batches of Freeciv games.

5.1 Freeciv

Freeciv is an open source game based on Sid Meier's Civilization. Freeciv is a turn-based strategy game where the player develops a civilization over the course of many thousands of years, researching technology and founding new cities along the way. The player encounters other civilizations which are competing for the same resources on a limited map. The player can trade, negotiate, or even go to war with any of the other civilizations. Balancing limited resources thus becomes an important part of the game strategy.

6.0 Frontier Algorithm

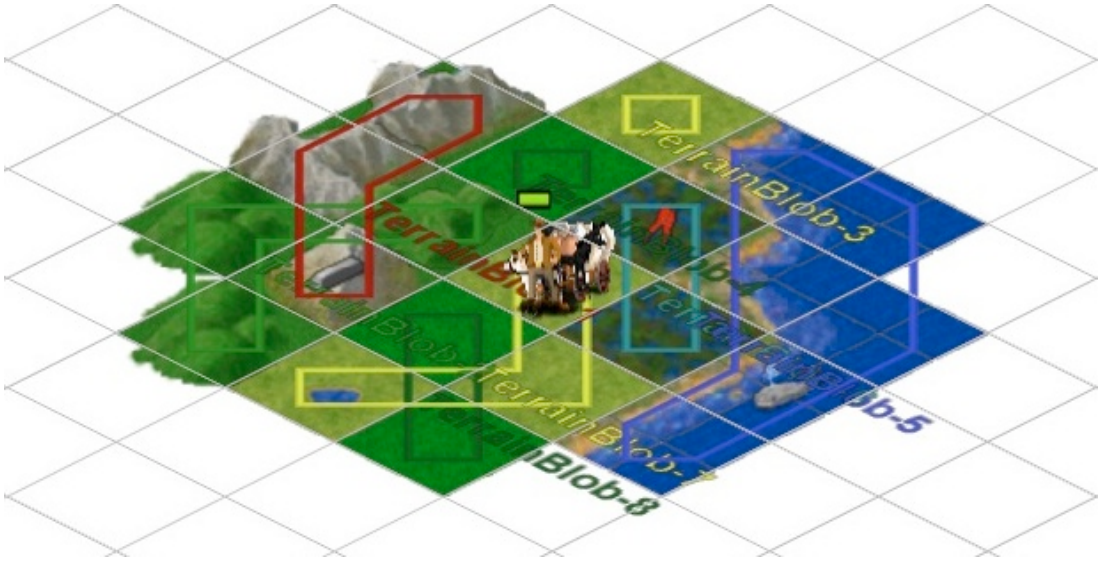
The Frontier algorithm works by scoring known tiles and choosing the tile with the highest score to explore. The algorithm is calculated differently for each explorer, depending on the explorer's position and the other nearest explorer's position. In this way explorers can coordinate between themselves. B_i is the score of a tile. It is the sum of three measurements:

$$B_i = W_n * N_u + W_p * D_p - W_c * D_r$$ (Al-Khawaldah et al., 2010)

The algorithm starts by scanning the map looking for tiles that are adjacent to unknown tiles, called "Frontier tiles." The first measurement N_u is obtained by counting how many adjacent unknown tiles there are. Because the map consists of square tiles, a Frontier tile can have between 1 and 8 adjacent tiles that are unknown. The second measurement D_p is the tile distance between the nearest other explorer and the Frontier tile. Diagonal moves are allowed and count as a tile distance of 1. The third measurement D_r is the tile distance between the explorer and the tile. D_r has a negative weight to reduce the score of frontier tiles that are far from the explorer.

6.1 Blob Algorithm

The Blob algorithm makes use of the Freeciv terrain map in order to choose where to explore. The known portions of the map are divided into continuous sections of the same tile type. These portions are called "blobs." Every turn the blobs are updated to reflect newly acquired information about the map. A value is associated with every blob, which is determined by what kind of tile it contains. This value is calculated by counting the number of tiles in the blob and multiplying by the point value of the tile. There are eleven tile types in FreeCiv, and each blob can be only one type. Tiles have three different resources each with their own uses: food, production and trade. I gave each tile type a score based on the sum of food and production. Trade is only on one tile, ocean, and it does not directly benefit the growth of the civilization. The tiles and their scores are included in Appendix B.



This is an example of a first turn setup in a game of Freeciv from map "I" of the Freeciv map corpus. This screenshot is taken while running the Qualitative Reasoning Group's Cogsketch software. The Blob algorithm has drawn shapes around the continuous sections of terrain. Here, TerrainBlob-1 (displayed in red) corresponds to a mountain range in known area. The white squares are tiles that have yet to be explored. The Blob algorithm goes through each known tile and adds up the blob value for adjacent tiles. It then subtracts the distance of the explorer to that tile, to discourage the explorer from wandering too far. In this example, the highest scoring tile is the bottom-most tile of TerrainBlob-1. It is surrounded on three sides by TerrainBlob-2 (which is made up of Hills, a valuable tile) and on one side by TerrainBlob-8 (made up of Grassland, also valuable). TerrainBlob-7 and TerrainBlob-1 also contribute to the score, but less substantially. The explorer will then move to this tile, and hopefully reveal terrain that is also valuable. Blobs will be redrawn to take into account new information as it is made available.

7.0 Experiments

I set up the experiments for testing the Blob algorithm and the Frontier algorithm in the following way. I created ten random maps in Freeciv and labeled them "A" through "J." For each map I had the computer play 50 turns with the Blob algorithm, and then play a new game for 50 turns with the Frontier algorithm. At the end of each game I recorded the number of known tiles, the sum of the point values for the known tiles, and the final population. The strategies for founding and growing cities was fixed between algorithms. Both algorithms had access to the same three explorers on turn 1, and did not build any more: one Explorer (which can move 3 spaces per turn) and two Workers (which can move 1 space per turn). The complete data from these experiments is contained in Appendix A, and the averages are displayed below.

Average Values	# of Tiles	Points	Pop. (K)	Points / Tile
Blob	262	392	130	1.50
Frontier	370	549	70	1.48

7.1 Analysis

Points/Tile Pairs:

Difference between Blob and Frontier tile values is D

-Assume there is no difference between algorithms: D = **0**

Hypothesis:

-Null Hypothesis: $U_d = D = \mathbf{0}$

-Alternate Hypothesis: $U_d \neq D = \mathbf{0}$

Significance Level = **0.05**

Analysis:

-Average difference between population pairs is: $\underline{d} = \mathbf{0.016013}$

-Number of pairs is: n = **10**

-Standard Deviation is s:

$s = \sqrt{(\sum(d_i - \underline{d})^2/(n-1))} = \mathbf{0.065950}$

-Standard Error is SE:

$SE = s/\sqrt{(n)} = \mathbf{0.020855}$

-Degrees of Freedom is: DF = 10-1 = **9**

-t-score = $(\underline{d} - D)/SE = \mathbf{0.76780}$

-P(t < -0.76780) = **0.2311**

-P(t > 0.76780) = **0.2311**

-P-value = 0.2311 + 0.2311 = **0.4622**

Result:

Because the final P-value of 0.4622 is greater than the desired Significance Level of 0.05, we cannot reject the null hypothesis.

Population Pairs:

Difference between Blob and Frontier population pairs is D

-Assume there is no difference between algorithms: D = **0**

Hypothesis:

-Null Hypothesis: $U_d = D = \mathbf{0}$

-Alternate Hypothesis: $U_d \neq D = \mathbf{0}$

Significance Level = **0.05**
Will use **Matched-Pairs t-Test** for analysis

Analysis:

-Average difference between population pairs is: $\underline{d} = \mathbf{60}$
-Number of pairs is: $n = \mathbf{10}$
-Standard Deviation is s:

$s = \sqrt{(\sum(d_i - \underline{d})^2/(n-1))} = \sqrt{(32000/9)} = \mathbf{59.628}$

-Standard Error is SE:

$SE = s/\sqrt{(n)} = 59.628/\sqrt{(10)} = 59.628/3.1623 = \mathbf{18.856}$

-Degrees of Freedom is: $DF = 10\text{-}1 = \mathbf{9}$
-t-score = $(\underline{d} - D)/SE = (60\text{-}0)/18.856 = \mathbf{3.1820}$
-$P(t < -3.1820) = \mathbf{0.0056}$
-$P(t > 3.1820) = \mathbf{0.0056}$
-P-value = 0.0056 + 0.0056 = **0.0112**

Result:

Since the final P-value of 0.0112 is less than the desired Significance Level of 0.05, we must reject the null hypothesis. The alternate hypothesis of $U_d \neq 0$ is more likely to be true.

7.2 Observations

When comparing the Points/Tile Pairs, the Blob algorithm performed marginally better than the Frontier algorithm. A higher tile value means that an algorithm succeeded in finding the more valuable tiles. Unfortunately the the small difference between pairs of points/tile means that the slight improvement by the Blob algorithm is not statistically significant. The desired Significance level was 0.05, and the P-Value here was a much higher value of 0.4622. Further refinements of the algorithm might lead to better results, but the data here is not encouraging.

Comparing the Population Pairs shows that the Blob algorithm performed significantly better than the Frontier algorithm. A large population number indicates that the civilization was able to found cities which grew quickly. A lower number indicates that the civilization struggled. After performing a paired t-test, the improvement of population on the Blob algorithm is indeed statistically significant. The final p-value of 0.0112 is much smaller than the the desired Significance Level of 0.05, so we can say with confidence that the Blob algorithm had some kind of effect.

8.0 Conclusion

By far, the most interesting result of these experiments is the dramatic increase of the final population when running the Blob algorithm versus the Frontier algorithm. Civilizations in Freeciv tend to grow exponentially because cities build settlers, and settlers found new cities and so on. So the

more cities there are the faster the civilization grows. Because of this exponential growth, a large population indicates that fast growing cities were founded early. It seems that the Blob algorithm was able to discover high quality city sites near the beginning of the game, and this contributed to the higher final population.

It is also worth noting that, overall, the Blob algorithm did not discover tiles with a significantly higher average value. I think that this is because the average tile value on a map in Freeciv is fairly consistent over large areas. This means that at the end of the 50 turns of a game, each algorithm had explored so many tiles that their average values could not differ by very much. Keeping tile values consistent is probably a gameplay and balance issue, because creating maps with wildly different tile values could give some players an unfair advantage.

9.0 Future Work

I think that the Blob algorithm would benefit from a higher degree of cooperation between the explorers. That way, different explorers could prioritize different sections of the map to speed up exploration. Right now the explorers try to explore locally. However, if two explorers happen to end up close to each other, then going to nearby tiles does not do a good job of separating the units. This is obviously inefficient, and multiple explorers should not be trying to cover the same tiles at the same time.

Another issue with the Blob algorithm is that the explorers tend to backtrack over known tiles frequently. This is because when the unit explores one area, the new resulting blob calculations can make tiles on the other side of the map more desirable. So the unit spends time traveling to the more desirable tiles, only to have the same thing happen again which results in more traveling. I was able to improve this issue slightly by introducing a score parameter which makes distant tiles score poorly. However the units still continued this behavior when distant tiles scored very highly. Addressing this issue would make the algorithm perform more efficiently.

10.0 Acknowledgements

I want to thank the many people who have supported me during this project. Ken Forbus was immensely helpful in guiding my research and encouraging alternate ways to think about problems. Tom Hinrichs always swiftly explained technical concepts to me, even when I had a multitude of questions. When I needed additional help, Matt McLure would graciously provide extensive feedback which helped me understand not just my particular problem but larger issues too. I would also like to thank Caitlin Killmer for her loving support and encouragement during very difficult times. I also owe a thanks to Goce Tracjevski for his help and his confidence in me. Finally I would like to thank my mother for her abundant support and encouragement, and my father for introducing me to computer programming and fostering my interest throughout the years.

11.0 Works Cited

Al-Khawaldah, M., S. Livatino, and D. C. Lee, “Weight Parameters Tuning for Frontier-based

Cooperating Robots Exploration," *Recent Advances in Signal Processing, Robotics, and Automation*, 2010
Houk, P. A., "A Strategic Game Playing Agent for FreeCiv," *Northwestern University, Computer Science Department, Technical Report*, 2004
Soule T., and R. B. Heckendorn, "Environmental Robustness in Multi-agent Teams," *Genetic and Evolutionary Computation Conference*, 2009
Forbus, Kenneth D., Tom Hinrichs, Johan de Kleer, et al., "The FIRE Manual," Qualitative Reasoning Group, Northwestern University, 2010a
Forbus, Kenneth D., Matthew Klenk, Jeff Usher, et al. , "Companions User's Guide," Qualitative Reasoning Group, Northwestern University, 2010b

12.0 Bibliography

Hartglass, D. A., "Master's Project Report," *Northwestern University, Computer Science Department, Technical Report*, 2010
Hinrichs, T. R. and K. D. Forbus, "Analogical Learning in a Turn-Based Strategy Game," *International Joint Conferences on Artificial Intelligence*, 2007
Yamaguchi, B., "Frontier-Based Exploration Using Multiple Robots," *Proceedings of the Second International Conference on Autonomous Agents*, 1998
Wilensky, U., "NetLogo," http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.

Appendix A: Experimental Results

	Blob	**Blob**	**Blob**	**Blob**	**Frontier**	**Frontier**	**Frontier**	**Frontier**
Map	# of Tiles	Points	Points / Tile	Pop. (K)	# of Tiles	Points	Points / Tile	Pop. (K)
A	233	360	1.55	90	408	605	1.48	100
B	243	380	1.56	150	386	567	1.47	90
C	259	379	1.46	180	336	507	1.51	90
D	283	426	1.51	170	382	594	1.55	50
E	230	352	1.53	120	407	596	1.46	140
F	236	328	1.39	230	346	484	1.40	50
G	296	439	1.48	120	341	524	1.54	70

	Blob	Blob	Blob	Blob	Frontier	Frontier	Frontier	Frontier
H	293	448	1.53	90	272	385	1.42	30
I	267	393	1.47	70	491	749	1.53	50
J	279	418	1.50	90	328	480	1.46	40
Average	**262**	**392**	**1.50**	**130**	**370**	**549**	**1.48**	**70**

Appendix B: Tile Types and Scores

Tile Type	Food	Production	Trade	F + P = Points
Forest	1	2	0	3
Grassland	2	0	0	2
Plains	1	1	0	2
Desert	0	1	0	1
Hills	1	0	0	1
Jungle	1	0	0	1
Mountains	0	1	0	1
Swamp	1	0	0	1
Tundra	1	0	0	1
Ocean	1	0	2	1
Glacier	0	0	0	0

Appendix C: Population Measurements By Turn - Maps C & F

Turn	Map F, Frontier Pop.	Map C, Frontier Pop.	Map F, Blob Pop.	Map C, Blob Pop.
0	0	0	0	0
5	10	10	20	20

Turn	Map F, Frontier Pop.	Map C, Frontier Pop.	Map F, Blob Pop.	Map C, Blob Pop.
10	10	20	60	20
15	10	20	60	50
20	20	60	70	30
25	20	40	80	80
30	30	50	80	110
35	30	90	100	80
40	40	110	120	150
45	70	130	160	140
50	70	100	230	180

Appendix D: Question & Answer With Thomas Hinrichs on 3/7/2012

1) *In the Frontier algorithm, what are the coefficients? How were they selected?*

   The coefficients are taken from Al-Khawaldah's original paper. They are derived from his experiments as having consistently fast results. From the source paper, the values are Wn = 2, Wp = 2, and Wc = 1. Technically my algorithm uses a weight of Wc = -1 because all of the terms are added together. In the original paper this term is subtracted, but the two expressions are mathematically equivalent.

2) *Wouldn't the analysis have been more straightforward if you had used three explorers instead of one explorer and two workers? Given that the Frontier algorithm uses the tile distance, the factor Dp is asymmetric now, since explorers and workers travel different distances per turn.*

   I could have modified the maps to include 3 explorers, but I used the workers because I wanted the initial game-state to be as original as possible. The factor Dp is asymmetric because of the way Freeciv calculates movement points. In a given game, the algorithm updates every time a unit moves so an Explorer might recalculate its destination 3 times in a turn (because it can move 3 times). However, if the destination is far away, the Explorer might use a "go to" command which will spend all of its movement points immediately without a chance to recalculate. I did not think this was a big issue because units in the Frontier algorithm tended to explore locally and did not use the "go to" command very much.

3) *Does Dr itself actually have a negative weight, or are you referring to the fact that it's subtracted in the equation?*

Yes, the term Wc * Dr is subtracted in the equation. The negative value in the weight Wc serves to keep the sign correct when adding multiple terms together. Dr is always positive because it is a measure of distance.

4) *In the Blob algorithm, it's fine that you don't take trade points into account in the tile score, but you might clarify that land tiles can have trade points if they have the right resources. In the long term, trade is important - that's where gold and science points come from, and gold lets you buy production items early, etc.*

It is true that trade is an important resource for playing the later stages of the game. However most trade (with the exception of Ocean tiles) comes from improving the land or special resources. For the purpose of growing cities and building settlers, I decided to focus the Blob algorithm on food and production and not worry about improving the land.

5) *Is there any role for resources in the algorithms? I assume not, since you can't predict the presence of resources on unexplored tiles.*

I did not include any recognition for special resources. Since special resources are almost always non-adjacent, they cannot make blobs on their own.

6) *I'm actually really confused by the Blob algorithm. It sounds like you take the visible map, compute the terrain blobs, compute the highest scoring tile and go there. Does it even take into account adjacency to unknown tiles? What's to prevent units from scrambling to the iron mine and just staying there?*

The scores are only calculated for tiles that are adjacent to unknown tiles. In this way the Blob algorithm is similar to the Frontier algorithm. So in your example, the first unit to move to the iron mine will reveal all adjacent tiles, and subsequently no unit will consider the iron mine as a possible destination.

7) *Your evaluation suggests that the Frontier algorithm did a better job of revealing terrain faster, but the Blob algorithm revealed marginally more valuable terrain on average. You emphasize the final population as the better criteria for exploration, but it's not clear what's going on there. Can you tell if the difference is due to founding cities earlier, or placing them where they will grow faster? In fact, because you didn't say what the city founding strategy really is, it's entirely possible that a worse exploration strategy would lead to a bigger population over 50 turns. That's because the city siting algorithm may ignore distance entirely, so a good exploration strategy could reveal high value sites that are farther away. Early in the game, that could lead to a lot of wasted travel time.*

The city siting algorithm works by searching for locations that meet certain conditions within a distance of 8 from the settler. Two of the conditions are that the tile must be on land, and the tile must not be too close to an existing or planned city. The tile is then scored based on how much food it can produce. This is calculated by summing the tile's food with the best other 2 tiles within a city radius. Of course, a city location that can produce more food is preferred. The city siting algorithm is reevaluated for each settler every turn and this will turn out to be important.

To get a better understanding on how the algorithms differed, I reran two of the experiments and measured the population once every 5 turns. I chose map F and C because in both maps, the Blob algorithm outpaced the Frontier algorithm. Map F highlights the slow start that the Frontier algorithm got in the first 10 turns, and then never catches up. On Map C, the Frontier algorithm gets a similar start but stays closer in population throughout the game. One thing that I noticed happening on both maps was the Frontier algorithm struggled to found its second city. On Map F the second city does not get founded until between turns 15 and 20, and on Map C it is not until between turns 5 and 10. From watching it play, I think that part of the Frontier algorithm's problem was that a settler would reevaluate its city siting every turn. So what was happening is that the 2nd settler would start walking towards a good city site, but before the settler arrived there a better city site would be revealed. The settler would then start moving there, but then the same thing would happen. The result of this is that the settler spent many turns moving and moving but never actually founding a city until many turns into the game. Having only 1 city at the beginning of the game instead of 2 is a significant handicap. The Blob algorithm did not suffer from this problem as much because tiles were revealed at a slower pace. Therefore it was less likely for the settler to see a better option before it arrived at its original site.

8) *You didn't say anything about Settlers in your experiment description, but you must have had at least one settler in the beginning.*

There are two settlers present at the beginning of each game. The starting unit roster is: two settlers, two workers, and one explorer.

9) *Assuming the Blob algorithm actually does work well, why do you suppose that is? As I admitted earlier, I don't fully understand that algorithm, but it sounds like you're banking on extending high-value terrain blobs. Given that the maps are randomly generated, why should we expect that a high-value blob (say, lush prairies) should continue into the unknown tiles? In the real world, we wouldn't expect abrupt terrain changes, but in Freeciv, who's to say?*

Yes, I was banking on extending high-value terrain blobs. I thought that the Freeciv map generator already tries to make maps that are "clumpy." I mean clumpy in the sense that similar terrain is put next to each other. Land tiles are often put next to each other, as opposed to randomly scattered in the ocean. Of course islands are a kind of scattering but since they contain many tiles, I considered them to be clumps too. Mountains would often appear as mountain ranges, and Forests would appear together. I think part of the reasoning for this is to match the expectations of the user.

A map where every tile had some independent probability would make a very diffuse world. Land, mountains, and forests would be scattered. Island would unlikely be recognizable. To a user, this would not look like our world because our world is clumpy. Part of the appeal of this game is feeling like you're recreating human history, and that a familiar world is important to that appeal.

10) *I'd really like to see some spot-checking of the scenarios to get a better understanding of why the end populations were bigger for the Blob strategy. If it turns out that it's just a better impedance match to the city siting strategy but not actually a better exploration strategy, that's ok. It's more important to understand exactly what's going on than to achieve better performance right now.*

I have included supporting data for spot checking two of the maps. See #7 for my analysis of what's going on.