Project 2: Slather

Group 1: Adam Chelminski, Kailash Meiyappan, Lu Yang

October 2016

Contents

1 Introduction

1.1 Premise

In Slather, players vie for control of a 10cm x 10 cm board using pre-programmed cells that naturally grow by up to 1% per turn, produce temporary trails of pheromones, and reproduce when their size reaches double their initial diameter of 1cm. Cells cannot collide with each other, and cells cannot collide with pheromones from another players cells. Each player can define a common algorithm for all their cells that has access to information about surrounding cells and pheromones, as well as a byte of persistent memory. The goal of Slather is to be the player that can reproduce the most and have the most cells on the board by the time no more reproduction can occur.

The game is made especially interesting because of the many different configurations that can be run. Adjustable variables include p, the total number of players, n, the number of starting cells for each player, d, the viewing distance for each cell, and t, the number of turns a pheromone lasts after being released by a cell. This means that players must be robust across multiple configurations that could range from anywhere between pairwise competitions with low view distance, short pheromone trails, and one starting cellto nine-player competitions with high view distance, long pheromone trails, and many starting cells.

1.2 Strategy Overview

Our strategy divides the game into the early game, the mid game, and the late game. The game changes from one state to another based on the number of enemy and friendly cells detected around the player cell. In the early game, our cell looks at all the cells around it and chooses the widest angle between two cells and moves in that direction. If there are no cells around, the cell simply stands still. As the cell reproduces, the daughter cells create distance between themselves based by moving into the largest angle. This leads to the creation of a cluster. Once the cells detect a certain number of cells around it, the cell infers that the map is starting to get packed and thus moves into the mid game.

In the mid game, we classify cells as border cells and interior cells, and independently as attackers and defenders. The first category is inferred from the cells surrounding the player, as border cells typically have a large number of enemy cells nearby and interior cells have few, if any enemy cells nearby. The second classification of attacker and defender is made at the time of reproduction. We make a cell an attacker with 66% probability and a defender with 33% probability. When there are no enemy cells around, they both behave the same way, going toward the largest gap between friendly cells. The difference is when there are enemy cells nearby. Attackers tend to move toward enemy cells. Their purpose is two-fold. On the border, they push outward, thus expanding the cluster. In the interior, they detect enemy cells, and surround them, preventing them from growing inside the cluster, much like a white blood cell attacks a disease. The defenders behave differently. They tend to move away from friendly cells. Defenders on the border keep enemies out. If an enemy comes near, if the defender gets pushed in, it sees the friendly interior cells and starts pushing outward. In the interior, the defenders do not really have a purpose except to divide.

The end game strategy is very similar to the mid game. The difference is that near the end, the gap between cells may not ever be large enough to move toward. So the cell randomly chooses different positions to move toward and if it succeeds in finding one, moves there.

The result is the creation of a cluster that keeps growing. The goal of this strategy is to

mark off space for our species and keep enemies out.

2 Strategies and Justification

2.1 Used Strategies

2.1.1 Largest Angle

When a friendly cell is being encroached by other cells or enemy pheromones, it must decide how best to move, because otherwise it will no longer be able to grow and reproduce. One of the first successful ideas we pursued in this project was the strategy of moving friendly cells toward the least inhibited path using a largest angle metric.

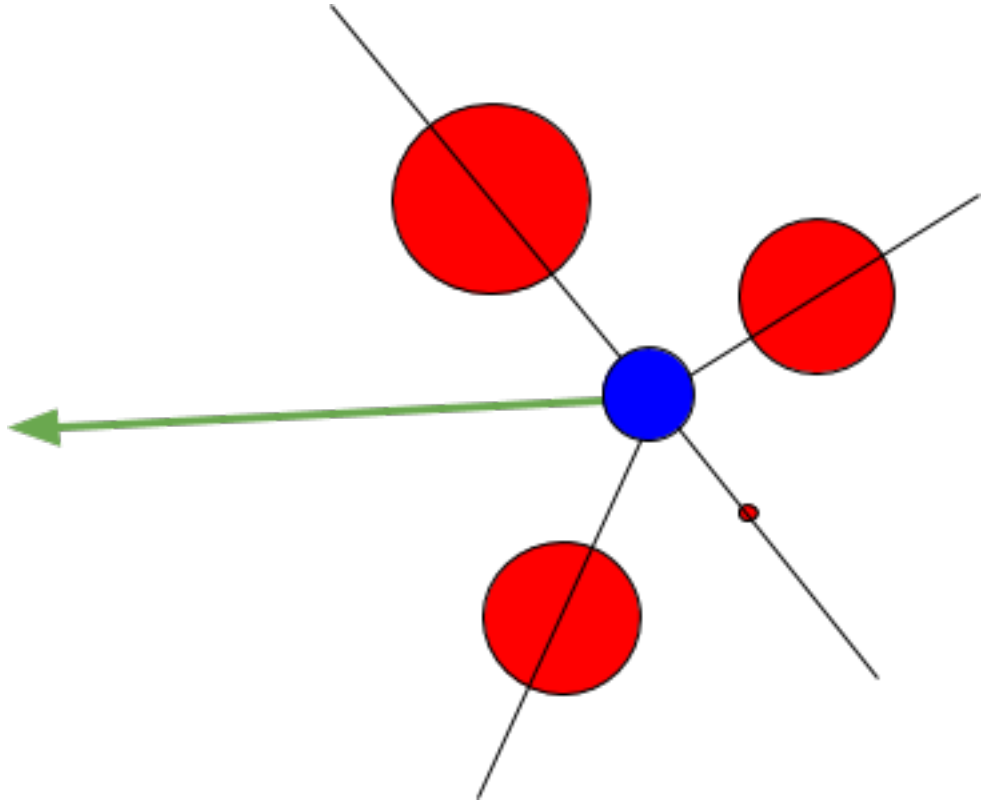

Figure 1

This strategy is demonstrated in the above figure, where the friendly cell is represented as a blue circle, and other cells or enemy pheromones are represented as red circles. The selected path (represented by a green arrow) is the path at the midpoint of the angle between the two cells or enemy pheromones with the largest angle.

The basic algorithm employed for this strategy is as follows:

1. If there is exactly one other cell or enemy pheromone within the view distance, have the friendly cell move in the direction directly opposite the other cell or pheromone.

2. Else if there are multiple cells or enemy pheromones (well call them enemy units) within the view distance of a friendly cell, there is an angle between every pair of units. Find all of these angles, and then find the largest of these angles. Have the friendly cell move in the direction of the midpoint of the two units that create this largest angle.

This strategy by itself provided very good results when competing with the default player, and was also the winning player in the class discussion where we first introduced it. Many other groups later adopted variations on this strategy in their players.

One legitimate concern with this strategy is that game configurations with large values for d may cause the player to evaluate too many angles and overlook paths that are actually not that inhibitive. To combat this, we opted to restrict the view to a variable. We start the variable at 1mm and increase it in steps of 0.5mm until a max of 5mm. We stop when we are able to see a cell and thus make a decision on which direction to travel.

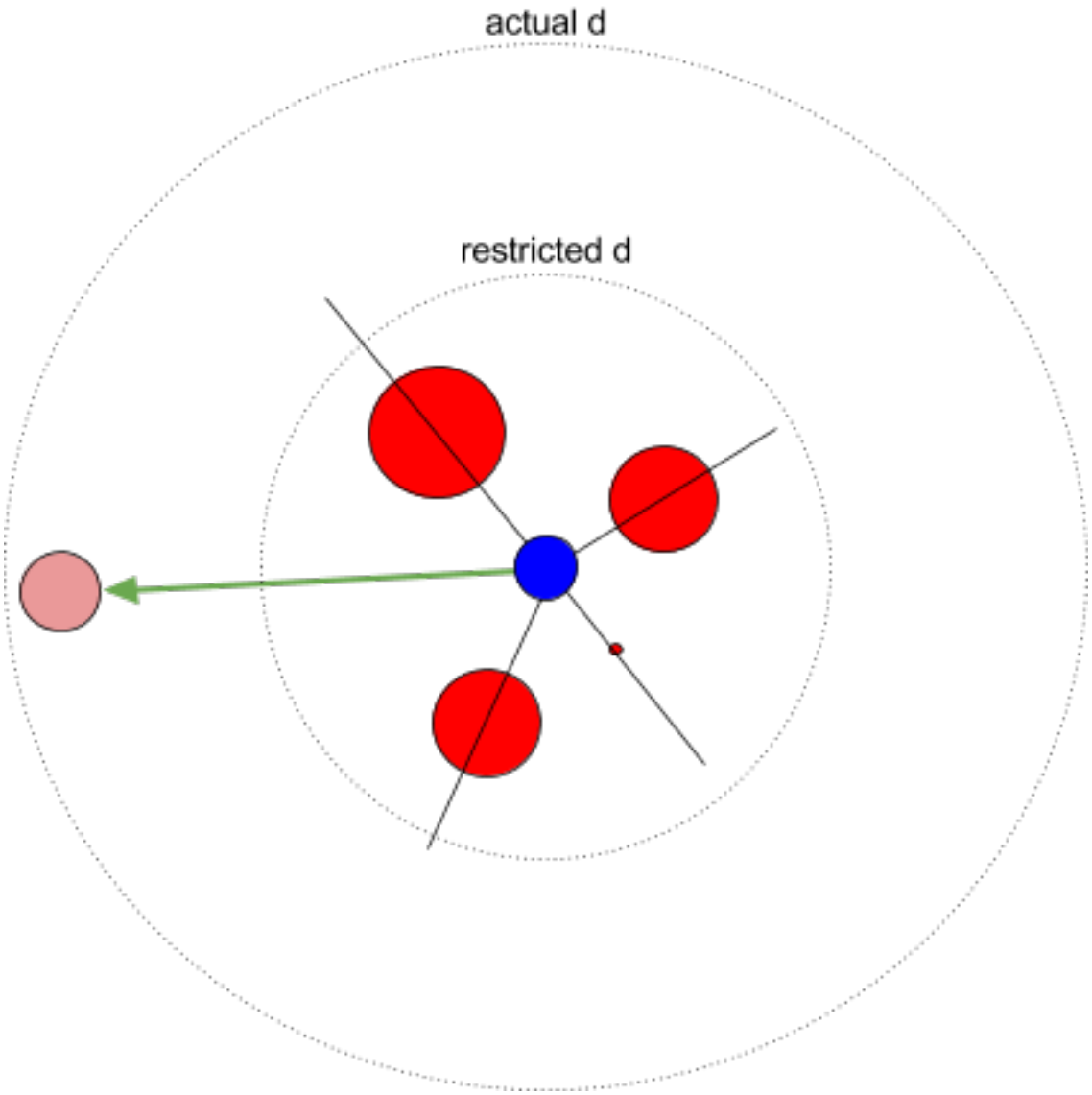

Figure 2

The above figure demonstrates the benefit of this view restriction. The pink cell is visible within the d of the game, but if the player considers it when making the determination of the largest angle, a less optimal direction may be chosen.
This largest angle strategy serves as the foundation for the early game strategy in our final

player. It is robust enough in to keep up with the rest of the groups for the first few generations of reproduction even when being encroached by other cells, and it works well as a means of creating distance between friendly cells to maximize reproductive potential.

2.1.2 Motionless by Default

For much of our development process, we adopted the strategy that most groups did whereby our cells constantly moved across the board. This worked moderately well as a way to explore empty space, but we noticed that this strategy fell apart in the later stages of the game. Our blind exploration strategy was overshadowed by other groups and their clustering strategies which were able to claim more space without needing to explore all of the empty space available on the board. In the last two weeks of the project, we moved to a strategy where our cells were motionless by default. This strategy was first adopted by Group 1 as a means of naturally clustering without developing a sophisticated clustering algorithm.

We took Group 1s basic principle and expanded upon it by creating a heuristic known as the safe range, which is the distance that friendly cells should ideally maintain between each other in this cluster. To make our initial cluster harder to penetrate, we wanted to limit this gap, but to make our cluster larger, we wanted to maximize this gap. This tradeoff is explained in more detail in the Parameter Tuning section where we justify the final chosen heuristic.

To execute this safe range heuristic, we simply employed the Largest Angle strategy discussed in the previous subsection, and set the restricted view distance to our safe range heuristic. By setting the restricted view distance to our safe range heuristic, the Largest Angle strategy in conjunction with keeping cells motionless by default naturally created a cluster of cells that were separated by no more than this safe range.

2.1.3 Game State Detector

The game has three states, the early game, the mid game, and the late game. The state is stored in memory and the transition between states depends on the number of cells near the player cell. The cell starts at the early game state, where it is motionless by default and avoids other cells by traveling toward the largest angle. The cell moves to the mid game state when there are at least three other cells within a distance of two units from it. Finally, the cell goes to the late game state when there are eight cells within a distance of two units from it. The values used to determine the state are discussed further in the parameter tuning section. The function that controls the change of state is called for each cell during each move.

2.1.4 Border Strategy

The goal of our player is to form a cluster and block off as much space as possible to grow. A key part of this is how our border cells behave in the mid game. There are a few important factors governing how our border cells behave.

- Enemy cells should not enter our cluster.
- Our border cells should expand as the cluster grows, creating more space to grow.
- The few stray enemy cells that enter into our cluster should be dealt with.

We divide our cells into two types: Attackers and Defenders. We decide which role they will be at the time of their creation, with attackers having a probability of $\frac{2}{3}$ and defenders having a probability of $\frac{1}{3}$. The values used here are further explained in the parameter tuning section. These values are stored in memory using one bit of information. The cells also have another parameter called the expansion ratio, which is randomly chosen from 1.0 to 2.05, in gaps of 0.15. The value (expansion ratio - 1.0)/0.15 is stored in memory, using three bits of information as this range requires numbers 0-7. These numbers are discussed in detail in the parameter tuning section.
A cell is determined to be on the border if:

$$n * (expansionratio) > m$$

where n is the number of friendly cells and m is the number of enemy cells. A larger expansion ratio would mean that the a cell is on the border for fewer enemy cells near it, while a smaller expansion ratio will prevent the cell from expanding too fast.
The difference between attackers and defenders is that attackers move in a way that takes them in a direction opposite to the largest angle between enemy cells, thus moving toward enemy cells. This makes them push the borders outward, increasing the space for our cells. The problem with attackers is that they often push too far, creating gaps in the borders that allow enemy cells to enter. To solve this problem, we use defenders instead. Defenders move toward the largest angle made by friendly cells. The difference between the two is subtle, but when an enemy cell moves toward our cluster, the attackers will avoid it and keep expanding, while a defender will attempt to push it out of the cluster. If the enemy cell manages to get past the defenders however, our attackers have another purpose. On the interior of the cluster, if an attacker comes into contact with an enemy cell, it moves toward it. This causes the enemy cell to be surrounded by attackers, preventing its movement. While this might lower the number of cells that reproduce within the cluster, this does not really hurt us in the grand scheme, because eventually the cells will reproduce and fill the space anyway. We credit group 9 for the idea of defenders, as their strategy involves making all their cells behave the same way as our defenders do. However, we found that attackers allow us to have more flexibility with expansion, which is one of the reasons we performed well in the two-player games.

2.1.5 Aggressive Interior Cells

Since we are using aggressive strategies not only restricted to the border, enemy cell inside our inner territory (which is the quite rare) actually help us a lot in packing our cell. Newly born attacker near to an enemy cell will move towards it very quickly which will later never move if no other enemy cell appear. This behavior helps us stop these cells from growing and save space later.
One very interesting fact we observed is that no matter in the 1-on-1 game or 9 players game, we always the last player to finish filling our space, which implies that we are the most space efficient team. This is because attacker role do not care about growing at all and often the case they are not growing. Therefore, using aggressive strategy **not only help us protect territory but also help us achieve higher packing rate.**

2.1.6 Random Player

As the game nears the late game, the best way to maximize the score is to pack cells efficiently. What our algorithm does is to try finding a direction vector using the mid game strategy. If the direction found is impossible to travel in, the cell instead looks at 10 random directions with distance sizes varying from 0.05 to 1.0 in steps of size 0.05. If it finds a direction to move that

does not collide with any cell, the cell takes it.
We did not expect much change from adding this random fallback strategy, but surprisingly the packing greatly improved. The figure 4 shows clearly how much better the random player packs cells than without it in figure 3.

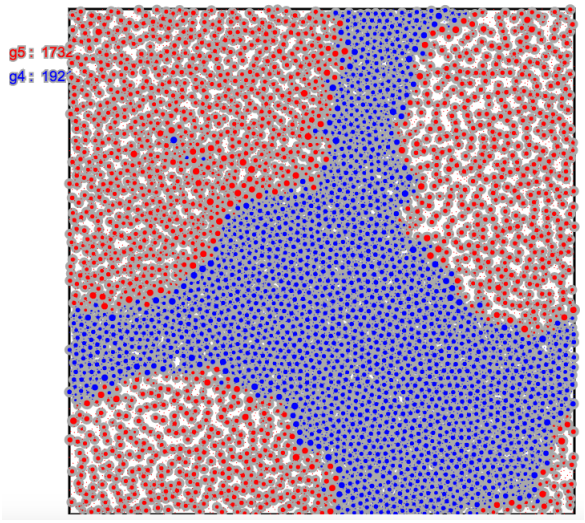

Figure 3

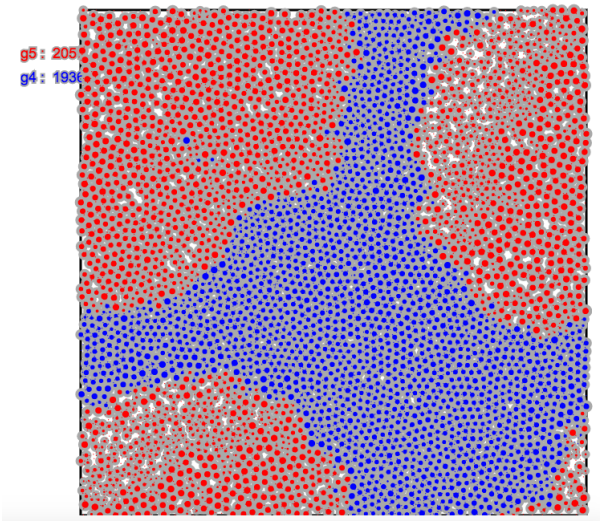

Figure 4

2.1.7 Largest Traversable Distance

We implemented a function that on input the direction would give the largest vector in the same direction that the cell could move without causing a collision. This is implemented by looking at all vectors in the direction of the input, of scale from 1.0 to 0.05, in steps of 0.05.

2.1.8 Modified Collision Checking

We realised that we should always allow the cell to grow after the current move is made. We modified our collision checking function to check if the cell would collide if it were 1% larger. This gives the cell enough room to grow on its next turn.

2.1.9 Increasing vision radius

Our cell initially restricts its vision to a low value of 1 unit. If there are no cells within this distance, we increase the distance to 1.5 and try again. We keep repeating this until a distance of a maximum of 5 units is reached. This improved our player greatly since we were more effectively able to use larger distance values, as opposed to our earlier implementation which only looked at distances of 2 units.

2.2 Failed Strategies

2.2.1 Circling Strategy

One of the first strategies we experimented with was to form a circle of consecutive cells. The idea behind this approach was to block off a large area for our species by having cells follow each others pheromone trail in a circle, thus creating an impenetrable wall. We planned to have this wall expand by increasing the radius of the circle so that we would eventually have a huge area that we could later populate.
This strategy failed however, for two reasons. The first was that it was not obvious how to deal with reproduction of cells. Once the cell reproduced, there were two cells in its place. This would break down the pheromone circle, because two cells would compete to be in the same position in the circle.
The second reason this failed and consequently led us to believe that pheromone trails were not useful, is the way the cells move. The cells move from a location (x,y) to a location (x + i, y + j) by moving instantly there. What this meant was that if we tried to block space with pheromones, a cell that knew our strategy could go close to the pheromone trail and then instantly move to the other side of it. We calculated that the distance a cell had to move was only $\sqrt{r^2 - \frac{1}{2}^2}$, where r is the radius of the cell. For a cell of diameter 1, since the distance between the pheromones is 1 unit, moving between them was very easy. In fact, any cell with a diameter less than $\sqrt{2}$ could move across pheromones, breaking this strategy. The picture below shows how a cell could travel over pheromones even though the gap is smaller than its diameter.

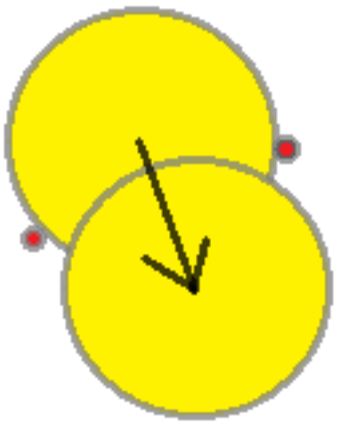

Figure 5

2.2.2 Purely Aggressive Strategy

One early strategy we attempted was to be aggressive toward enemy cells when they were visible by friendly cells. The theory behind this was that we could inhibit the growth of other players by intentionally causing collisions. The inherent flaw in this strategy is that inhibiting the growth of another player will also inhibit the growth of friendly cells. The result is that this strategy works okay in pairwise competitionespecially against random or naive players that dont attempt to evade attacks, though it totally falls apart when more than two players are involved. Being aggressive against one player simply lets the third player grow unfettered. We thus learned early that having a primarily aggressive strategy. Our final strategy contained some aggressive elements as described in the Border Strategy section, but they were balanced against the competing interest of allowing friendly cells to grow.

2.2.3 Tangents

The largest angle strategy uses vectors from the center of the player cell to the center of the nearby cell. In reality, the player cell can only move along the inner tangents, as shown in the picture below.

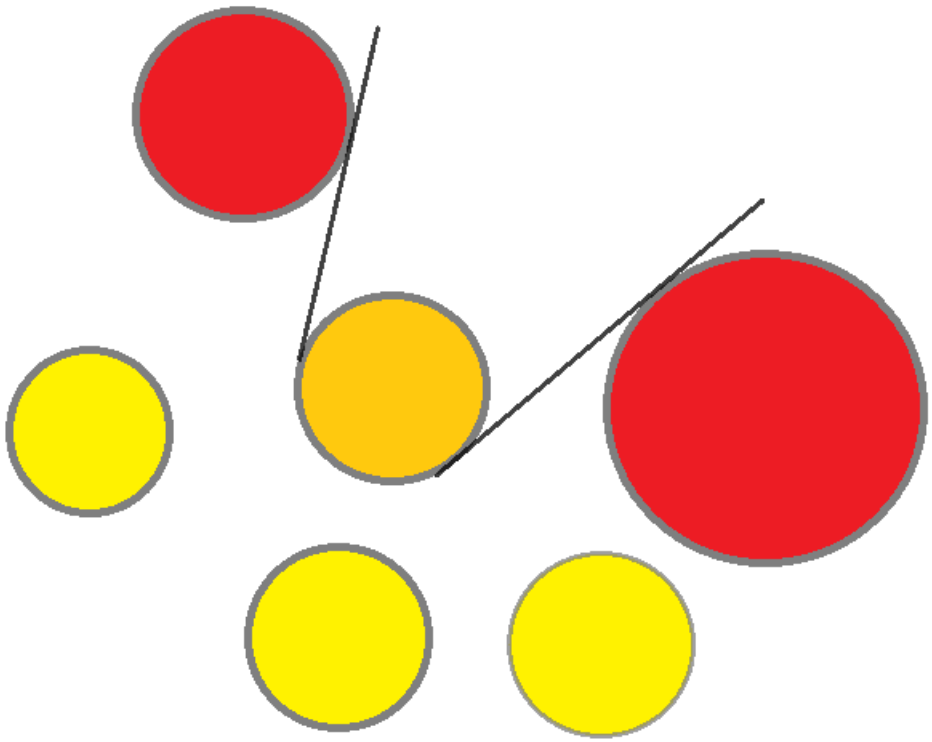

Figure 6

Our implementation for the angle strategy was changed to use the tangents instead of the centre - centre vector. We expected an improvement from this strategy.
However, this performed poorly and was often beaten by the simpler centre - centre vector strategy. We believe the reason for this is the implementation of movement. In the simulator, a move such as figure 7 is considered valid.

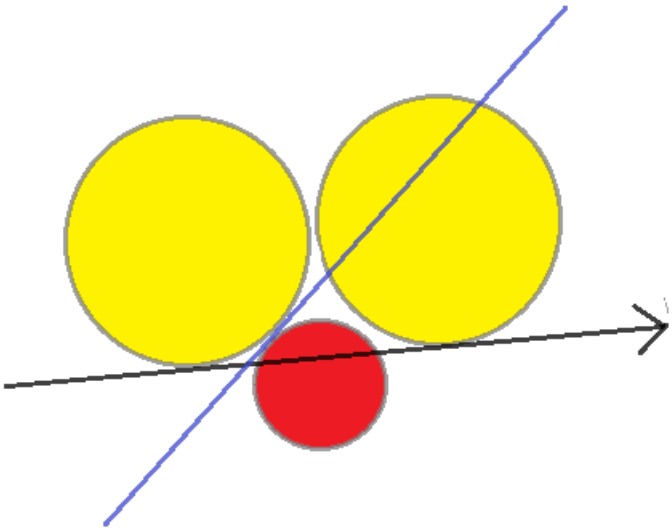

Figure 7

The line in blue is the tangent and the line in black, which is clearly not allowed in normal movement is still a valid move because the cell travels instantly. We abandoned this implementation because of this failure.

2.2.4 Generation Tracker

Before we used the heuristic of nearby cells to determine the current stage of the game for cell role assignment, we explored the strategy of tracking how many generations of reproduction have passed. This idea fit well with the constraints of the game since we were able to use four of the eight allowed bits in memory to keep a count of the current generation, we could increment this count on reproduction, and we could use the current generation as a heuristic to determine the role of the current cell.

The problem with this idea was not with the implementation, but with the fact that a generation counter is wholly insufficient when it comes detecting the current stage of the game. The number of allowable generations of reproduction within the 10mm x 10mm board depends entirely on variables unknown to the player: n, the number of starting cells for each player, and p, the total number of players on the board. In a pairwise (p=1) competition with n=1, cells could often reproduce over nine generations before a late-game strategy was viable. In a competition with p=9 and n=20, late-game conditions could appear after only three or four generations of reproduction. For this reason, we chose to abandon this strategy and use those memory bytes for the more useful purpose of maintaining a role and the randomness for expansion ratio.

2.2.5 Escape Closest Cell

Our first attempt at a late game strategy was to look at all nearby cells, pick the closest one and move in the opposite direction of it. This did not work as well as the random direction

lategame strategy, so we abandoned it.

2.2.6 Largest Traversable Distance using Binary Search

The largest traversable method looks like it could use binary search in its implementaion to get more precise values than the linear check we ended up using. This turned out to not work well in practice. The reason for this is that since cells move instantly, the collision function is no longer monotonous as a function of the length of the vector.

3 Parameter Tuning and Memory

This section discusses the parameters we used and why we selected certain values:

3.0.1 Midgame Cell Threshold

This is the number of cells that the player should see for it to transition from the early game state to the mid game state. The value we use is 2. This corresponds to a cluster of around 10 cells, which is large enough for our mid game algorithm to work correctly. The reason we do not use larger values is that if we stay in the early game state for too long, other cells might enter our cluster early on and harm our mid game strategy, which is more important.

3.0.2 Expansion ratio

This is the rate at which the cluster expands. A larger value of around 2.0 of expansion ratio leads to larger expansion at the cost of creating gaps in the border that enemy cells can enter. A smaller value close to 1.0 on the other hand does not expand as fast and may cost our species a higher score. To balance this, we decided to use different values of expansion ratio for different cells. Each cell is randomly assigned a random number from 0 - 7 and this determines the expansion ratio. The random values are stored using 3 bits of memory.

3.0.3 Role Probability

This is the probability that a newly created cell is assigned defender or attacker. A large number of attackers performs very well in 2 - player games, but it performs terribly in multi player games because of the gaps that attackers create in the border. When enemy cells enter these gaps, attackers on the interior surround them. Since many enemy cells would permeate our cluster, the entire interior of our cluster gets ruined by having all attackers.
All defenders on the other hand performs well in multiplayer games. In fact many of the top performing groups like g9 and g2 use a strategy with all defenders. This creates tight borders that other cells cannot penetrate but they do not push against other clusters to expand as much as attackers do. We found the best proportion of the two to be $\frac{2}{3}$ attackers and $\frac{1}{3}$ defenders. The reason for this is that this was the lowest proportion of attackers we could choose to outperform other players in 2-player games, which we made our goal as a group to win. One bit of memory stores the role of the cell.

4 Tournament Analysis

4.1 Pairwise Games

For both configurations chosen by Kevin for pairwise tournaments, we were the dominant player and **won every single matchup**, even when ran again in independent simulations. We believe that this is because we were the only group to adopt a strategy with outwardly aggressive elements, as described in the Border Strategy section. In pairwise competition, aggressive behavior doesnt benefit a third player, which can make it a viable strategy if it sufficiently disrupts the enemy player. The following screenshots, taken in simulations running the pairwise n=2, d=2, t=10 configuration show how an aggressive border strategy was able to disrupt enemy players g1, g3, g6, g7, and g8 (not pictured), in the early to mid stages of the game to allow our player to claim significantly more space before the game entered later stages:

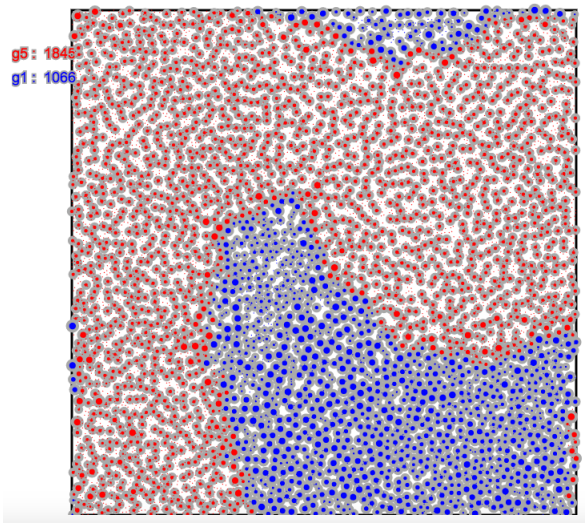

Figure 8

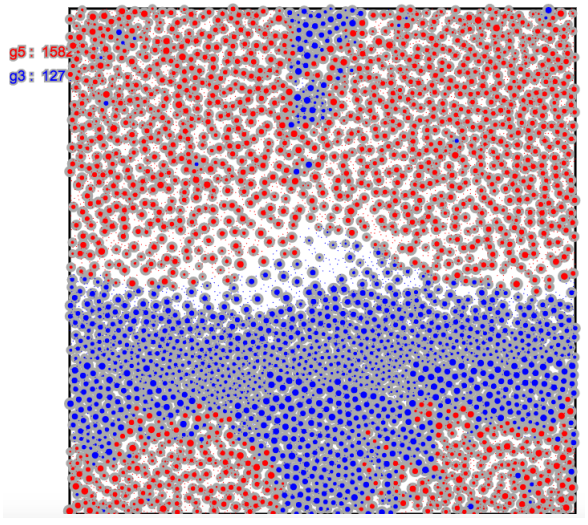

Figure 9

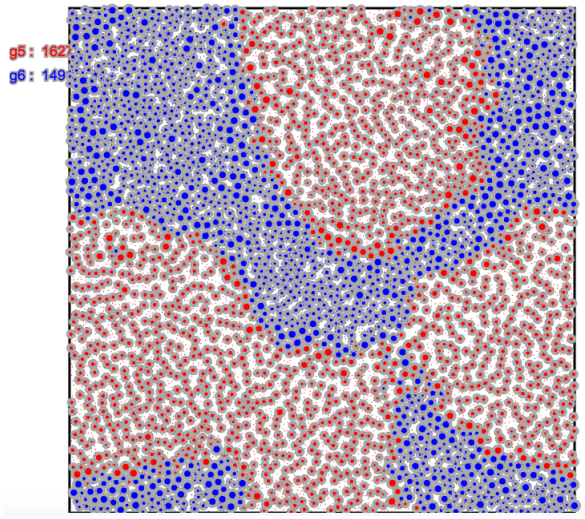

Figure 10

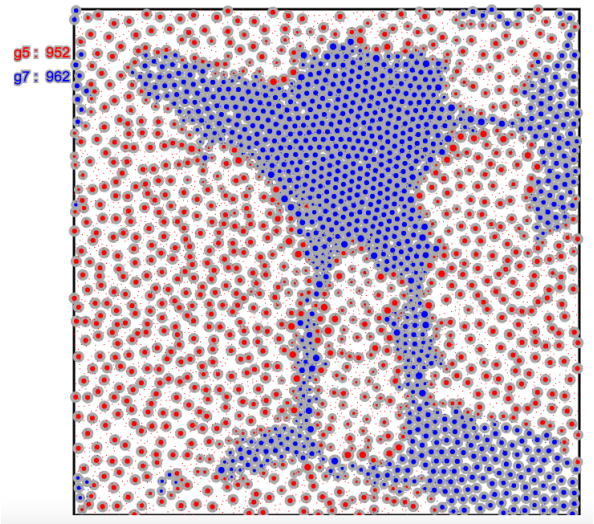

Figure 11

The aggressive strategy was also useful when trying to quell enemy infiltrations into our clusters, as in one simulation against g4:

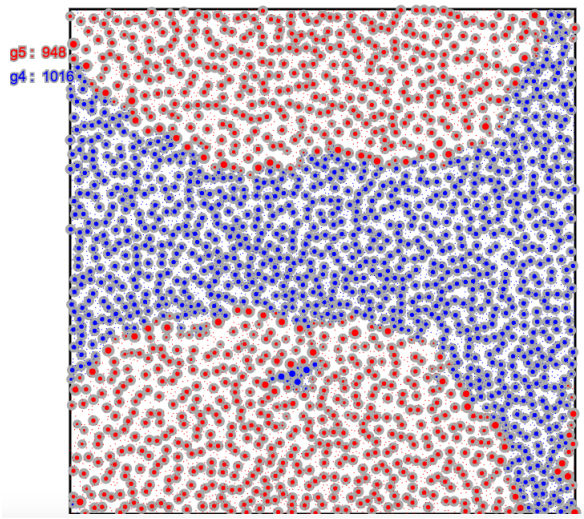

Figure 12

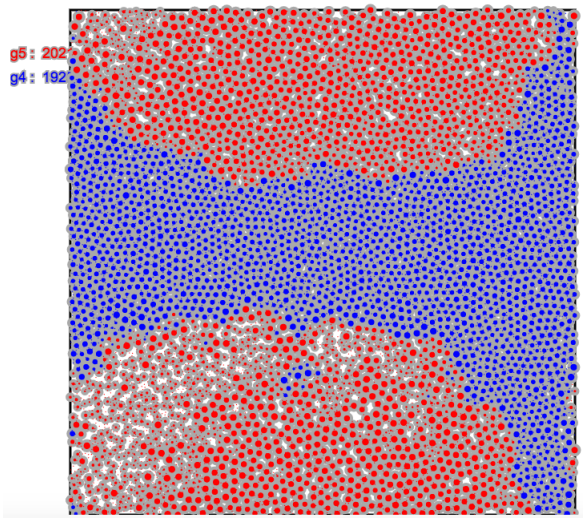

Figure 13

The grouping of blue g4 cells in the friendly cluster pictured above was inhibited from growing and disrupting the reproduction within our cluster because several cells were designated as attackers to prevent it from growing.
There were also cases in the pairwise competitionsuch as with g2where we were losing in the mid-game, but eventually overtook the other player because of our late-game random player strategy which efficiently packed cells into space that we claimed in the early game:

Similar phenomena were observed for the n=1, d=5, t=20 configuration, which produced wins with even larger margins!

All in all, the mean margin of victory for the n=2, d=2, t=10 configuration was 1147.5, and for the n=1, d=5, t=20 configuration it was 1254. These means were skewed by outlier results, so we also calculated the median margins of victory which were 872 and 1245.5 respectively.

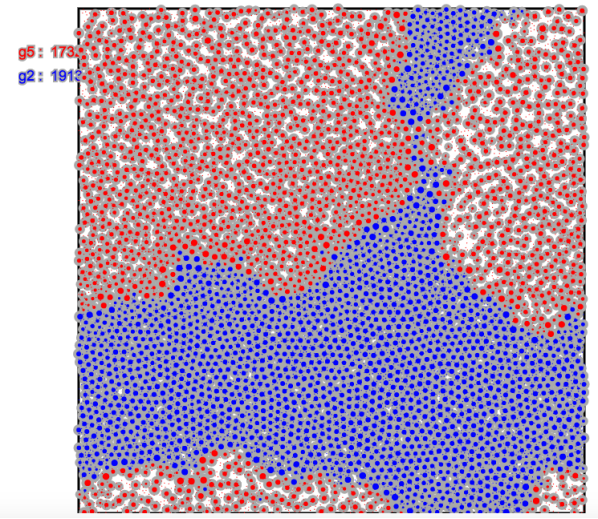

Figure 14

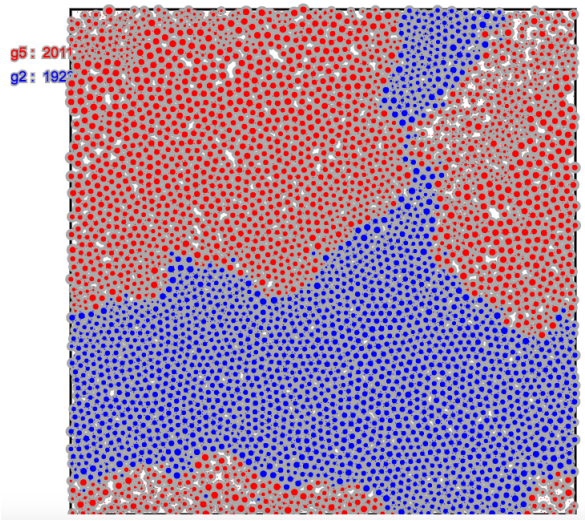

Figure 15

4.2 All Player Games

Our expectation for Slather was that since our group strategy relies on creating and expanding a large cluster, we would perform poorly when we could not create as large a cluster. This would happen either if there were too many players or too many starting cells, which would lead to smaller clusters. We came in fifth for most of the n=63, d=2, t=10 games. Unexpectedly, we performed better than we expected for n = 20, coming in the top four places for most of the games. We performed very well for the games with n = 1, d=1 and t =2 coming in first three times and in the top 3 in six out of ten games. We also performed well in n=1,d=4,t=4 coming in the top three for about half of the games. This was expected since low n values meant that we would form a single large cluster. We did not perform as well as we hoped for the case n=1,d=10,t=1, coming in fifth for a majority of the games. We suspect that other players used the increased vision better than we did. We infer from these runs that our player outperforms the others for the case that n is small, and there is low vision, irrespective of pheromones, but does poorly in games with large vision range or large number of

starting cells. During our tests, we concluded that the choice of parameters is very important and determines the performance for different cases. We were able to determine a good choice of parameters for 2-player games and for multi player games separately, but we were unable to unify the two. Below is the result of tuning our parameters to play well against multiple players.

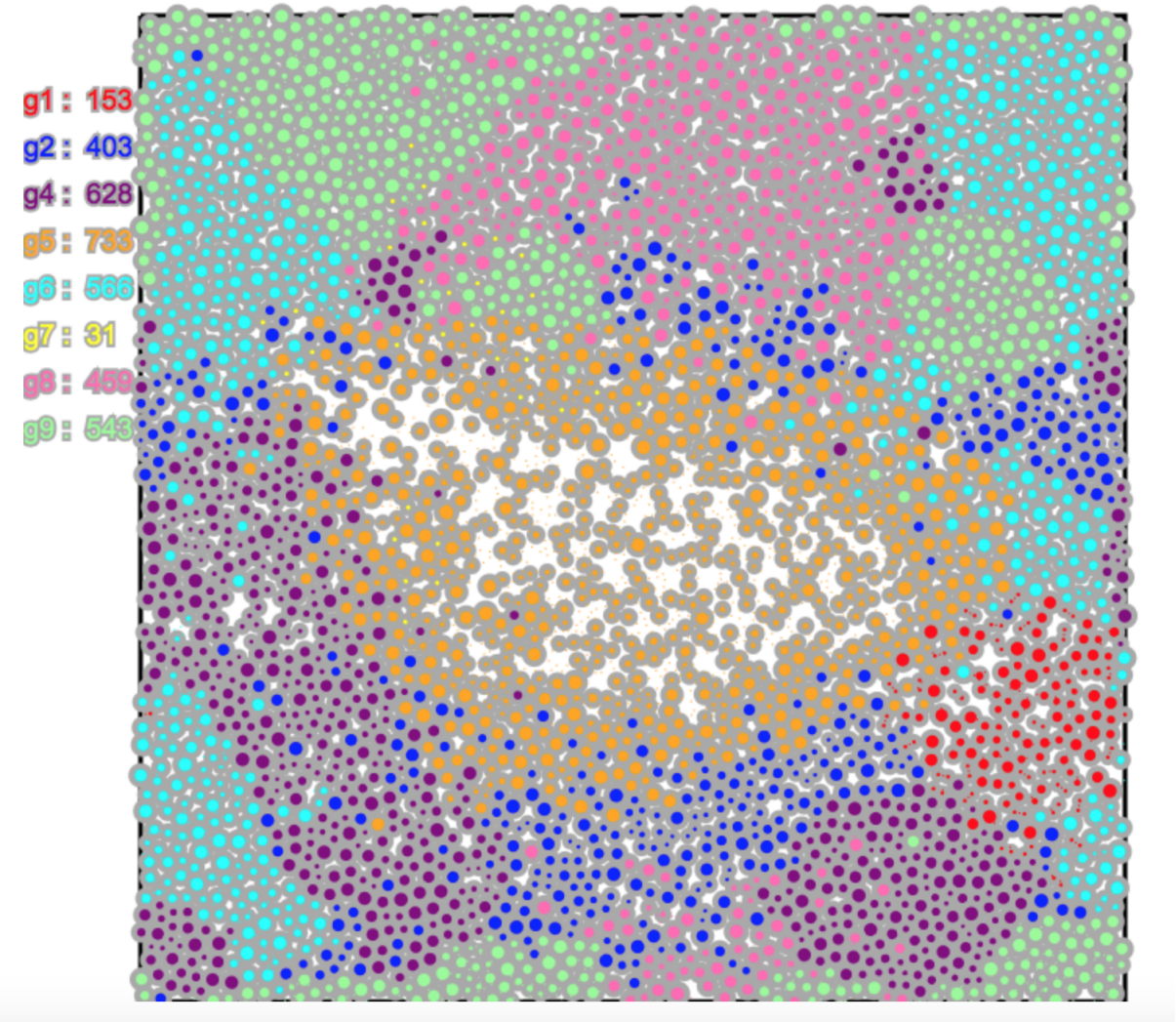

Figure 16

5 Future Improvements

Based on our experience, we found it to be very clear that the effectiveness of a Slather strategy depends on whether the strategy is pitted against a single other player or against several other players. We were able to independently develop two strategies that always won in one of these two cases, but we were forced to make a tradeoff for our final player. If we were to have more time on the project, wed spend time investigating the feasibility of having cells detect the number of players in the game and communicate this knowledge to all friendly cells. This would allow us to select the strategy that works best for the number of players in a particular game.

6 Conclusions

The key finding we made regarding Slather is that its very difficult to create a player that is robust across all configurations. No single strategy is ideal for all configurations, so creating robustness was a matter of understanding the tradeoffs of each strategy, and making an informed decision based on the configurations in the tournament. As the tournament results illustrate, we went with a strategy that performs excellently in pairwise competition, but is average in nine-player competition. We made this tradeoff because we knew that a significant number of the tournament runs would be pairwise. If the tournament was structured differently, or if our player was supposed to be as general purpose as possible, running on configurations unknown to us before the tournament started, wed likely reevaluate and readjust our strategy further.

7 Acknowledgement

We would like to thank Professor Kenneth Ross for this fun course and the discussion classes. We would also like to thank Kevin Shi for making the Simulator.