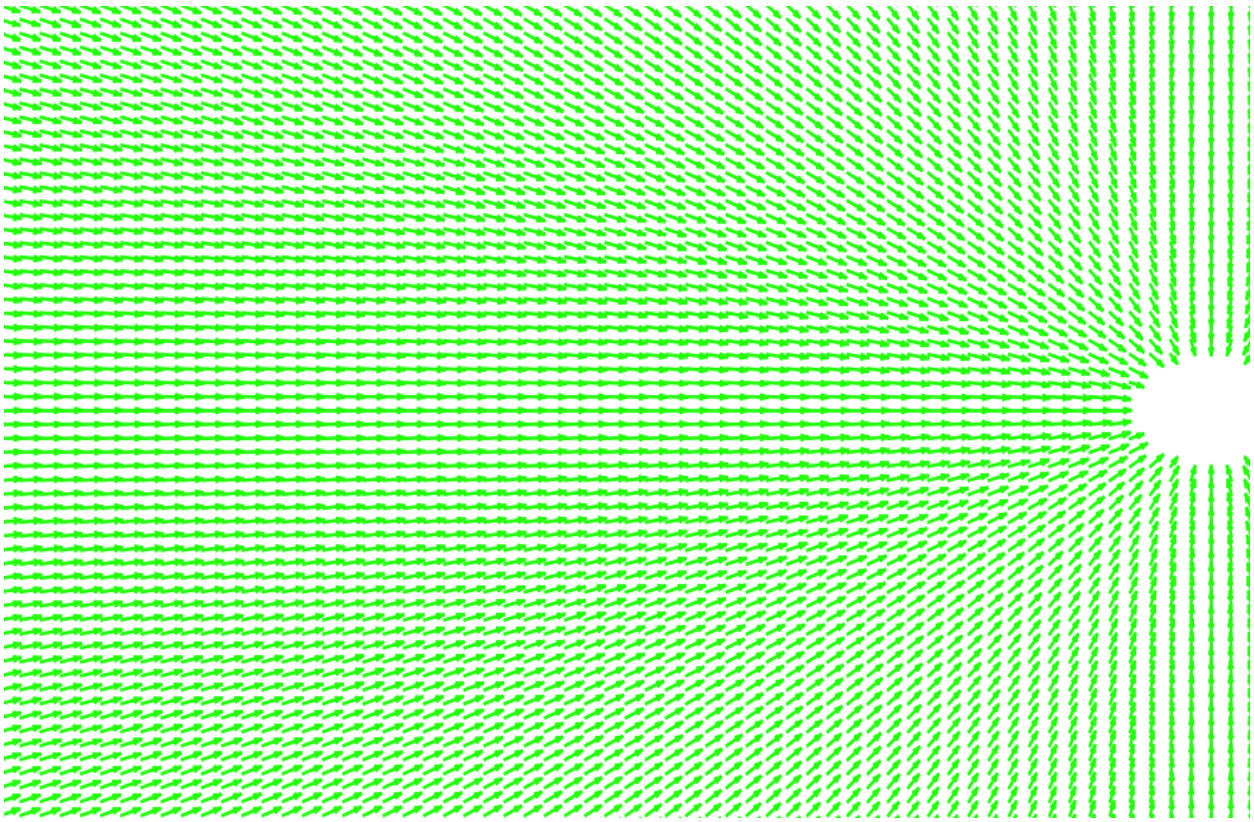
Potential Fields

Todd Greener & Jeff Hoffman

Following is a description of our potential fields project. First is a discussion about the use and implementation of the fields with our agents, and then a description of how they performed. For this lab we used three potential fields: attractive fields, repulsive fields, and tangential fields. We performed several tests again dummy agents and agents implemented by another group in the class.

We implemented the field classes such that they each inherited from a common interface which only has one method: fieldAtPoint(x, y). Each subclass then has their own special version of this method which allows for different fields. We chose this approach to make adding all of the fields together conceptually simple. One must merely create a collection of fields, iterate over it, and sum the field vectors together.

The first and most fundamental field used by our agents is an attractive field which guides them toward a goal. An image of this field is shown below. If an agent does not have a flag, the center of this field will be on one of the opposing teams' flags, but if instead the agent is carrying a flag, then this field will be centered on the agent's own base.

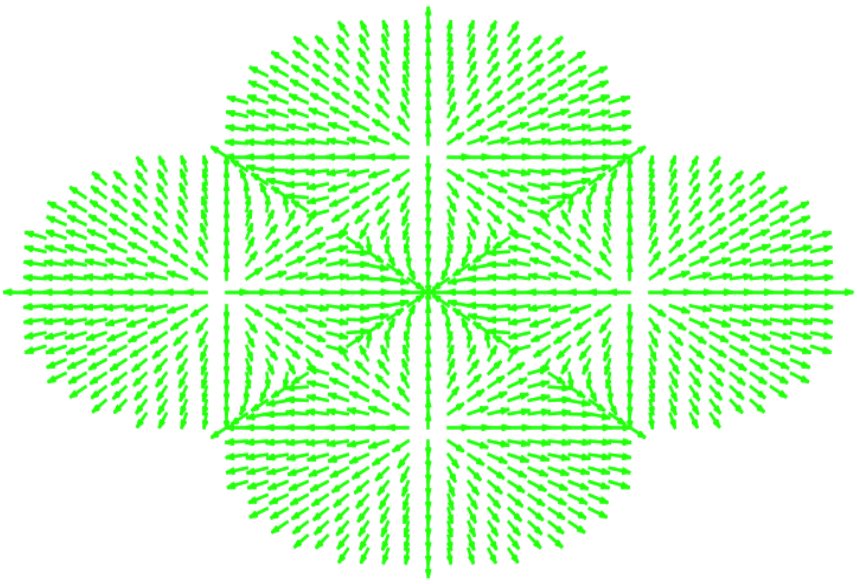


The unique code used to generate the field vector for attractive fields looks like this:

```
double dx, dy = 0;
if(radius <= d && d <= (spread + radius)) {
      dx = alpha * (d - radius) * cosTheta;
      dy = alpha * (d - radius) * sinTheta;
}
else if(d > (spread + radius)) {
      dx = alpha * spread * cosTheta;
      dy = alpha * spread * sinTheta;
}
```

Basically this is saying that if the distance is between the radius of the goal and the spread of the field, then the vector is proportional to how close you are to the goal. If the distance to the goal is greater than the spread of the field, then the maximum possible vector is returned instead. If neither of those conditions are true, the only remaining place you could be is within the goal, so the zero vector is returned.

The next field we implemented was the repulsive field. We used this field to prevent the agents from running into obstacles. Below are shown the repulsive fields used in the rotated box world. Each of the boxes had a repulsive field at its center, designed to keep agents away. If you look closely at the picture you may notice that there is a dead zone in the middle of the four boxes where there is no way for the agents to leave. This would be a problem if any agents ever managed to wander into that area, but if you look at the outer parts of the field, you'll see that the agents never actually want to go into the center at all.

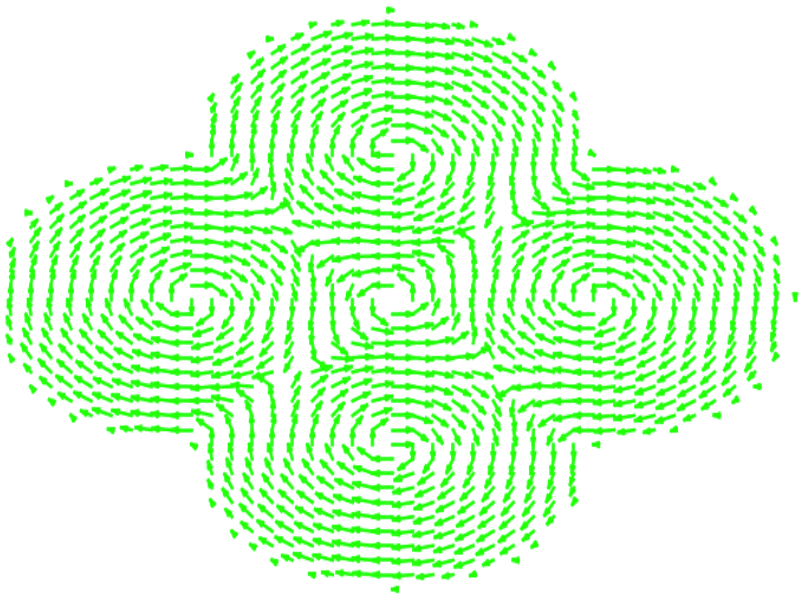


The unique code used to generate the field vector for repulsive fields looks like this:

```
double dx, dy = 0;
if(radius <= d && d <= (spread + radius)) {
      dx = -alpha * (spread + radius - d) * cosTheta;
      dy = -alpha * (spread + radius - d) * sinTheta;
}
else if(d < radius) {
      dx = -getMaxMagnitude() * cosTheta;
      dy = -getMaxMagnitude() * sinTheta;
}
```

This is very similar to the attractive field, as far as the different regions to consider go; however notice how the region between the spread and the inner radius gives a vector away from the center of the field (by flipping the sign), and the proportion is the inverse of the attractive field. That is, the closer to the center of the field, the higher the magnitude of the vector. For the repulsive field the regions inside the inner radius and outside the spread are opposite of the attractive field. When inside the inner radius the strongest vector is used, and the zero vector is returned when outside the spread.

The last field we made was the tangential field. We used this help agents circumnavigate the obstacles in the center of the rotated box world. Notice how the field keeps the agents moving around the boxes, avoiding getting stuck. There is also an interesting vortex pattern in the center that appeared when putting all the fields together though it would not cause an agent to get stuck.

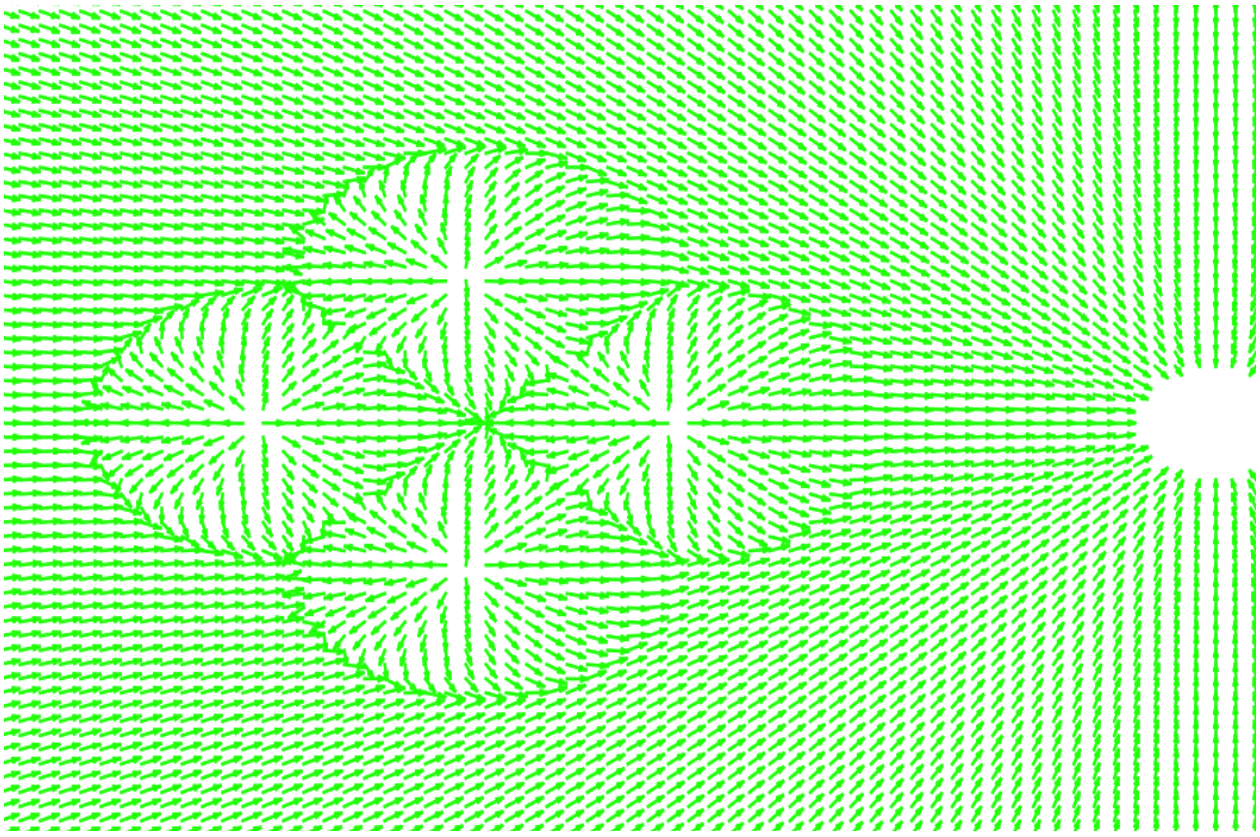


The unique code used to generate the field vector for repulsive fields looks like this:

```
double dx, dy = 0;
theta = rotateTheta(theta);
if(radius <= d && d <= (spread + radius)) {
	dx = -alpha * (spread + radius - d) * cosTheta;
	dy = -alpha * (spread + radius - d) * sinTheta;
}
else if(d < radius) {
	dx = -getMaxMagnitude() * cosTheta;
	dy = -getMaxMagnitude() * sinTheta;
}
```

This is pretty much exactly the same as the repulsive field, except that we rotated the angle which we are repelling the agent by 90 degrees (clockwise or counter clockwise can be changed using a setter).

For each of these agents we were able to tweak them by changing the values of the inner radius, the spread, and the strength (alpha) of the field. In this way, we were able to create an attractive field that covers the entire field and has a stronger pull than the other fields. This makes sure that the robot is always moving generally toward the goal, and is less likely to get distracted by other motivations. We also combined the repulsive and tangential fields to both keep the agent away from obstacles and cause the agent to move around them.



Agent Tests

1. Our agent against our really dumb agent was an easy win for the agent. Both the dumb agent and the agent scored a somewhat similar number of points by killing tanks, but the agent spent much less time stuck in corners or facing the walls and more time moving towards goals and capturing flags, so the agent was able to achieve a much higher score.
2. Against two dumb agents our one intelligent agent fared about the same as it did against one dumb agent. The intelligent agent avoided obstacles and moved towards goals, while the dumb agent was often stuck against walls and obstacles.
3. Our pf agent against itself was a more interesting match up. Both agents were set to specifically capture each other's flag, and it was impressive how consistent the scores remained during the match. Both agents seemed to capture flags within almost a minute of each other.

Against a classmate opponent

We tested our robots against the robots of Daniel Gunnell and Nick Wilson. I'm going to refer to them as our opponents.

4. Our pf agent fared similarly well against the dumb agent of our opponents as it did against our own dumb agent. It took slightly longer for our agent to capture flags, which we mostly attributed to lag between our client and the server and our agent failing to update as often as it needed to. Agents in this matchup still had a somewhat similar number of kills, but only the pf agent was able to capture the flag.

5. Against two dumb agents our robot was once again superior, both dumb agents spent a significant amount of time trapped in corners and failing to attack opposing tanks or bases. Our agent did not greatly excel in kills, but it did capture the flag enough to come out ahead. In the end we were ahead in kills because our robot wasn't as likely to be facing a wall or obstacle, but that was not a significant part of our robot's strategy.

6. Fighting against our opponent's pf agent was much more of a challenge for our robot. We did spend most of the early part of the match ahead of our opponent, but in the end we fell behind in points for a few reasons. Our agent only sought one flag at a time, and it was more successful at quickly capturing that flag than our opponent was at capturing a flag. Our opponent's agent, however, targeted multiple flags at once. Because of issues with server response time both agents had a difficult time updating quickly, so compared to the performance we observed watching our agent try to capture a flag with no opponent it was more difficult for our agent to perform. Still, it was a close match between our two agents, but our strategy could use some improvement. It was interesting to see how often both teams had a couple tanks that looked like they were "lost". We mostly blamed odd behavior like that on tanks failing to update quickly enough.

Other Team's results

Our opponent's agent was similarly able to destroy our dumb agents with great success. Their agent did win against ours, and it did behave in an intelligent manner.

Time Spent

Todd and I spent about twenty hours each on this project.