Fields

Attractive Fields

My implementation uses attractive fields for two reasons: to attract my tanks to an enemy's flag and to attract them back to home base. Whenever I refresh the list of active potential fields, I test to see whether or not I have an enemy flag. If not, I create an attractive field around an enemy flag. The attraction covers the entire playing field. The magnitude of the resultant vector is proportional to the distance the tank is from the target, with a maximum magnitude of 2.0.

If I do have an enemy flag, I create an attractive field around my home base. To do this, I create four different attractive fields, one around each corner of my base. Once again the attraction covers the entire playing field, and the magnitude of the resultant vector is proportional to the distance the tank is from the target. My initial maximum magnitude for each field was 1.0, but then I realized that those vectors added up and were overpowering the obstacles' repulsive fields, so I lowered the maximum magnitude down to 0.25.

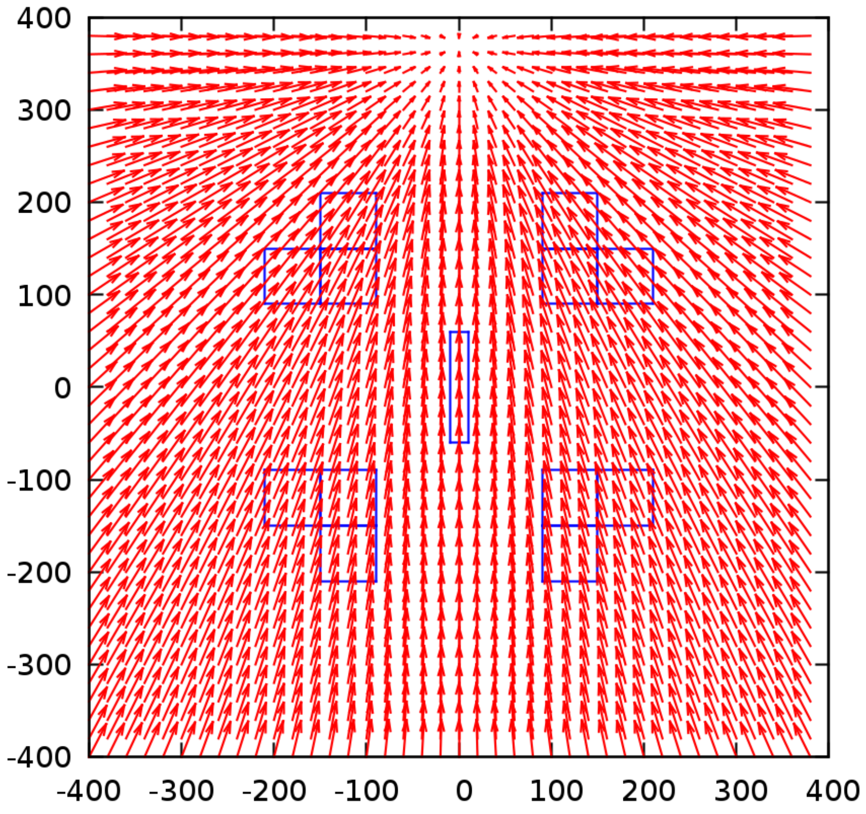

Figure 1 - Attractive field toward enemy flag

My initial idea to attract myself back to my home base was to set up an attractive field around my own flag. Of course that worked when I was playing with my really dumb agents, but I realized it would not work if an enemy tank were smart enough to take my flag.

Repulsive Fields

My implementation uses repulsive fields to avoid fixed obstacles. My implementation is naïve in that it only sets up repulsive fields around the corners of obstacles, so an obstacle with a long, flat wall might ruin the game for me. The repulsion covers a radius of 50 units around each corner. The magnitude of the resultant vector is inversely proportional to the distance the tank is from the obstacle, with a maximum magnitude of 0.5.

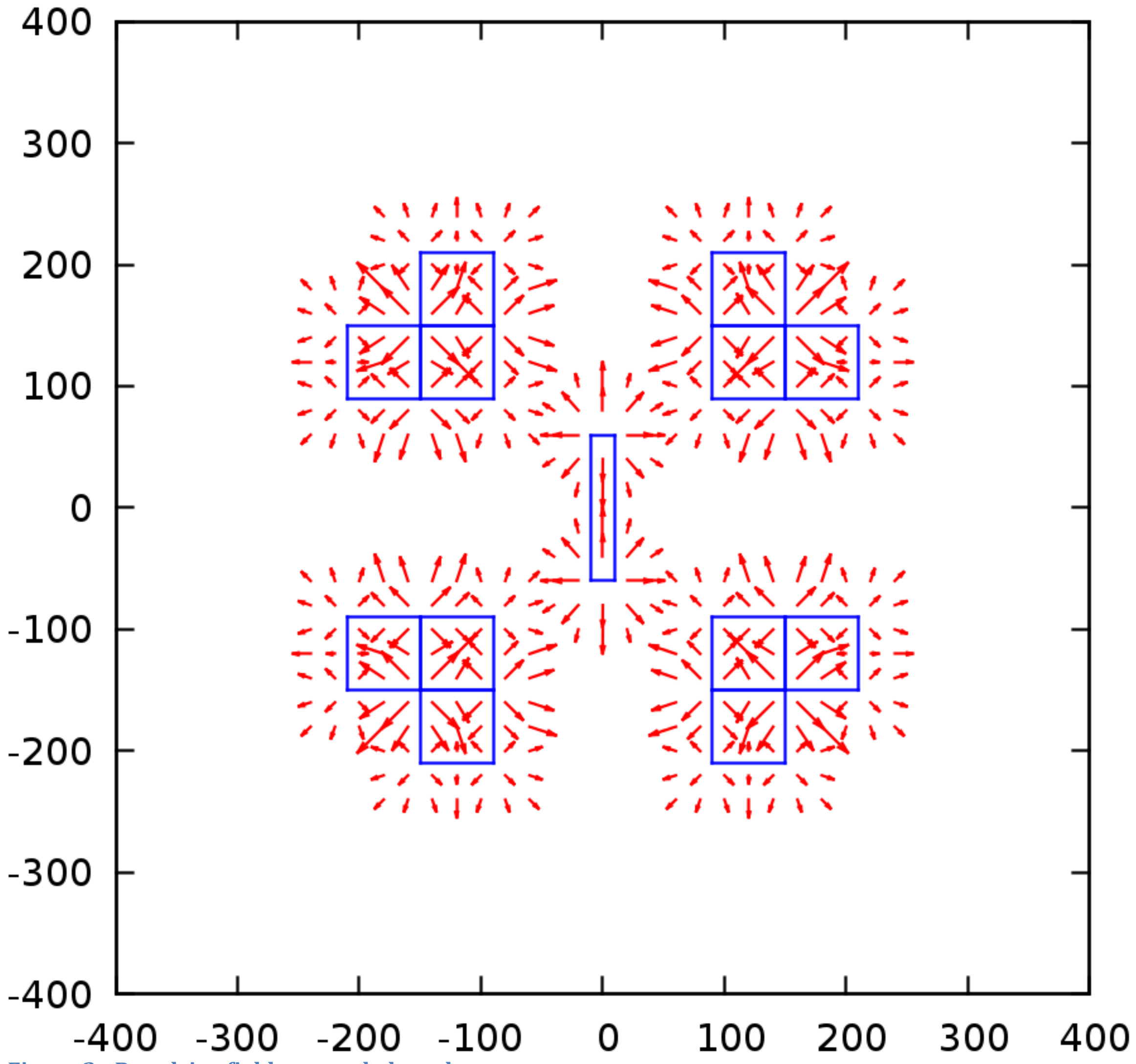

Figure 2 - Repulsive fields around obstacles

Tangential Fields

My implementation uses tangential fields for two reasons: to go around obstacles and to slide past other tanks. Obstacles also have repulsive fields, so I made my tangential fields a little larger (with a radius of 60 units) so that a tank would start turning around an obstacle before being repelled by it.

Tanks have small tangential fields, only twice the size of a tank's radius with a maximum magnitude of 0.1. I initially gave tanks a repulsive field and a larger tangential field, but when enemy tanks huddled around their flag my tanks couldn't get to the flag. The small tangential fields help me so that I can slide past a tank if I run into it, but they won't stop me from getting the flag.

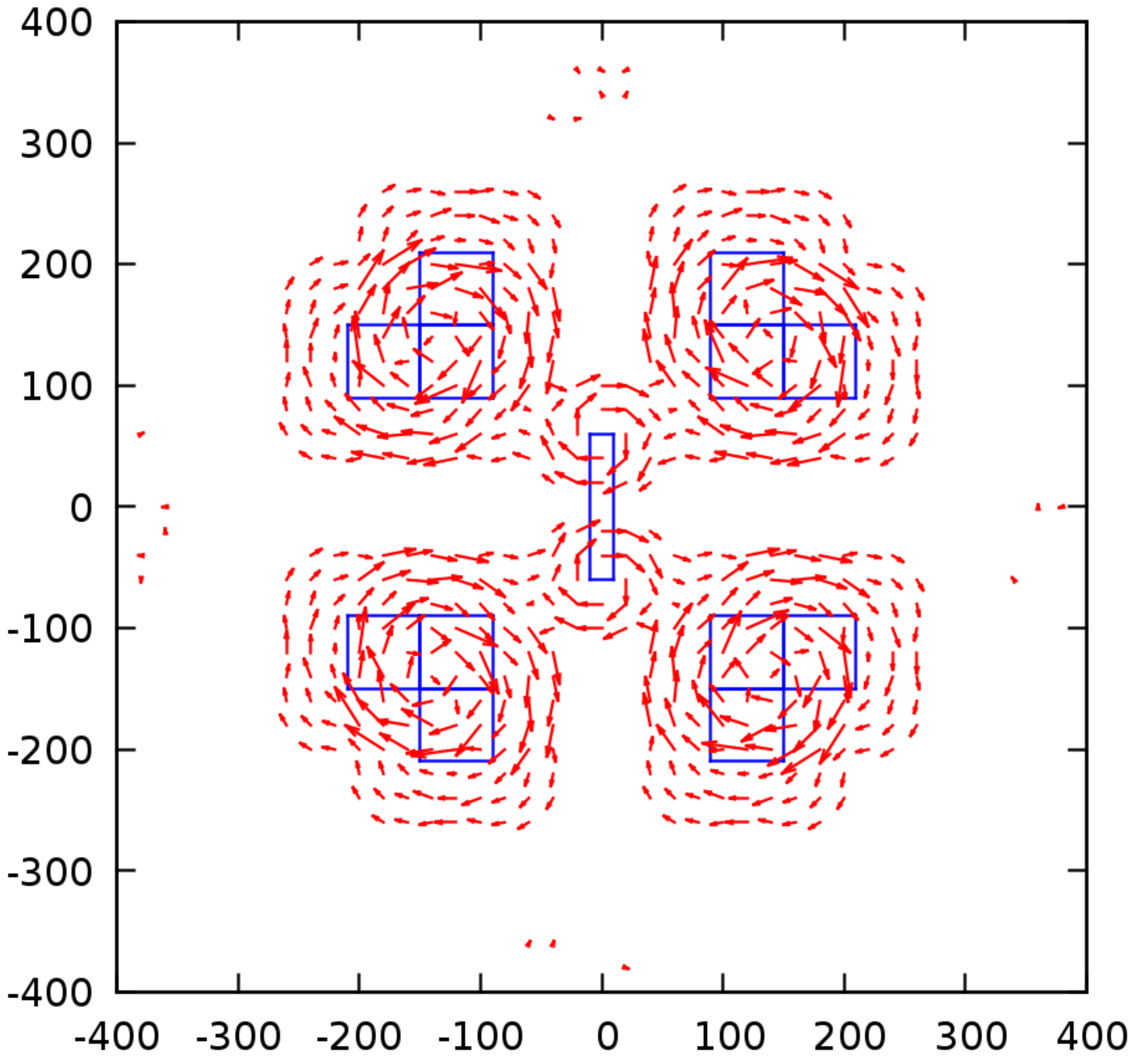

Figure 3 - Tangential fields around obstacles and tanks

Combined

These two figures show the results of all of these vectors combined.

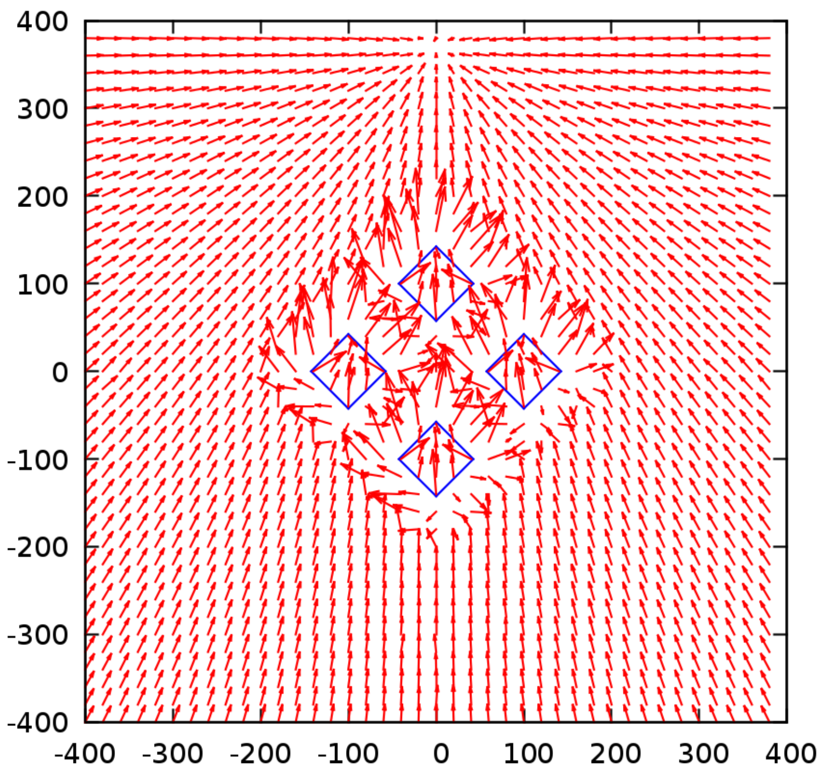

Figure 4 - A graph of the potential fields when a blue tank is returning an enemy flag to its base in the rotated box world

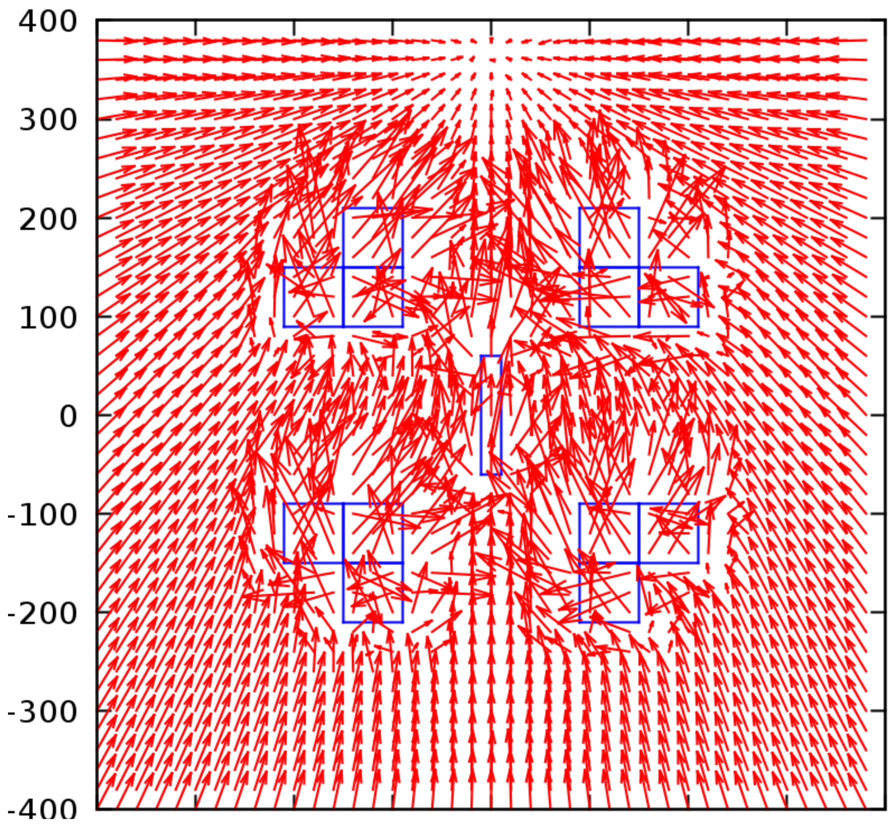

Figure 5 - A graph of all the existing potential fields when a green tank is pursuing the blue flag in the four_ls world

Testing

Running Against My Really Dumb Agents

My PF agent always won against my really dumb agents. The dumb agents, of course, have no real goal and just aimlessly shoot. Though I did not program my PF tanks to avoid shots, my tanks did not get hit and were able to grab an enemy flag and make it back to home base.

Running Against My PF Agent

When I ran two reactive agents against each other, I thought that the agent that I ran first would win, having the slight time advantage. However, this wasn't the case the first time I ran it. I first ran blue, then purple. The blue one took a little longer to get to the purple flag, and once it did get there it got stuck between two enemy tanks. The two tanks' tangential fields seemed to cancel each other out. As fate would have it, the returning purple tank (programmed to shoot at random) shot one of the two tanks in the blue tank's way, and the blue tank was able to start heading back to its base. However, by that time the purple tank had made it back with the blue flag.

Running Against Other Really Dumb Agents

I tested my implementation against Christopher Morgan's. Unfortunately for me, Chris's really dumb agents were much smarter than my really dumb agents, as he commanded the entire team of tanks in parallel, while my really dumb agents just switched control back and forth between two tanks. My PF agent controlled just a single tank, so that single tank did get shot several times by Chris's really dumb agents. I always won eventually, but it took some time. I'm sure that if I had all tanks working together in parallel that I would have been able to fare better.

Chris Against My Really Dumb Agents

As my really dumb agents were dumber than Chris's really dumb agents, he was able to do really well. My really dumb agents shot his tanks several times, but as he commanded the entire team in parallel it didn't slow him down much. His team did have problems with friendly fire, though. His tanks kept shooting each other and passing my flag back and forth between them. A couple of times they overshot my flag completely, but always made it back around to grab it eventually.

Running Against Chris's PF Agent

To make it fair, Chris modified his code so that he would only command a single tank, like I was doing. If he hadn't done that then he definitely would have won both times. The first time we played against each other, we both got the other's flag, but Chris made it back to his base first and won. The second time we played against each other, his tank was actually closer to a different team's flag, so his tank went after that flag. He was able to quickly get that flag and return to his base. It took me more time, but I was able to get his flag and return to my base. His team lost points when I stole his flag, so I technically won.

Chris's tanks didn't seem to be as precise as mine, but they definitely moved a lot faster. In talking with him about his implementation, I found he used some interesting ideas. For example, both of us made it so that a tank's speed is always non-negative. To deal with repulsive fields, we would stop the tank and change the angle instead of move the tank backwards. Chris dealt with this by making his velocity purely a function of his angular velocity. If his tank was far from its desired angle, its angular velocity would be high and it would slow down to a stop. The closer his tank's angle was to the target angle, the faster his tank would go. Thus his potential fields had no direct effect on his speed. His potential fields only determined his desired angle, and then the error of the angle was used to determine his speed.

I dealt with this problem in a completely different way. I programmed each of my potential fields with a magnitude that would affect my speed as well as my angle. My tank would set the angle, set the speed, go forward for a few seconds, slow down, then reset the angle and reset the speed according to the potential fields affecting it at its new location.

The problem with Chris's implementation is that the potential fields have no direct effect on speed, and the problem with my implementation is that my tank just goes forward blindly for a few seconds and then has to slow down to readjust. I think it would be interesting to combine our ideas by programming a tank so that its speed is a function of both the magnitude of the vector defined by the potential fields and the error of the angle.