Michael Zhang
netID: mbz27

NOTE: For the implementation of the agent forces, I tried to choose constants that made it so the agents do not get pushed through walls and have them stay a fixed distance away from each other. This resulted in some peculiar forces that sometimes had a chain reaction where an agent pushes itself away from another agent (or wall) and the neighboring agents do the same and so on until one or two agents get flung out of the crowd. I tweaked it so that this scenario will happen less frequently.

Youtube Links to Demos:
Part 1 - https://www.youtube.com/watch?v=qGMh-pw6fxs
Part 2 Crowd Following - https://www.youtube.com/watch?v=-wTpZvMqKCY
Part 2 Leader Following - https://www.youtube.com/watch?v=CN5EU0igCdo
Part 2 Purse and Evade - https://www.youtube.com/watch?v=f1v8KZJw-cQ
Part 2 Spiral - https://www.youtube.com/watch?v=AdphPtRMmjA

Part 1:

Goal Force
I used this formula:

$$m_i \frac{v_i^0(t)\mathbf{e}_i^0(t) - \mathbf{v}_i(t)}{\tau_i}$$

Where m_i is the mass of the agent, v_i0 is the desired velocity, e_i0 is the normalized vector pointing away from the agent to the next point towards the goal, v_i is the current velocity of the agent, and t_i is a constant.

Proximity Force
I used this formula:

$$A_i \exp[(r_{ij} - d_{ij})/B_i]$$

Where r_ij is the sum of agents i's and j's radii (which in this case is just 2*r since all of the agents have the same radii), d_ij is the distance between agent i and j, and Ai and Bi are constants. This is the same formula used for the proximity for walls, simply replace agent j with the wall.

Repulsion Force
I used this formula:

$$kg(r_{ij} - d_{ij})$$

Michael Zhang
netID: mbz27

Where g is a function that is zero if the agents do not touch otherwise it will equal r_ij - d_ij, and K is a constant. This is the same formula used for the repulsion force for walls, simply replace agent j with the wall.

Sliding Force
I used this formula:

$$\kappa g(r_{ij} - d_{ij})\Delta v_{ji}^{t}\mathbf{t}_{ij}$$

Where g is a function that is zero if the agents do not touch otherwise it will equal r_ij - d_ij, delta v_jit is the tangential velocity difference where delta v_ji is the difference between the agents velocity vectors which we take the dot product with the tangential direction (v_j - v_i)·t_ij, t_ij is the tangential direction, and kappa is a constant.

Wall Forces
I used this formula:

$$\mathbf{f}_{iW} = \{A_i \exp[(r_i - d_{iW})/B_i] + kg(r_i - d_{iW})\}\mathbf{n}_{iW} - \kappa g(r_i - d_{iW})(\mathbf{v}_i \cdot \mathbf{t}_{iW})\mathbf{t}_{iW}$$

This formula uses the same components that were described before for the agents except agent j is the wall.

<u>Part 2:</u>

Crowd Following
I used the same formulas as I did in part 1 except in the goal forces formula I used a different formula for e_i0:

$$\mathbf{e}_i^0(t) = \mathrm{Norm}[(1 - p_i)\mathbf{e}_i + p_i \langle \mathbf{e}_j^0(t)\rangle_i]$$

This takes a constant p which dictates the weight between following the crowd or following the path to the goal. Where e_i is the vector that points towards the goal (for agent i), and <e_j0(t)>_i is the average vector of all of the neighboring agents' vectors that point toward the goal. This produces a very interesting pattern where if the constant p is a large number (closest to 1) the agents will favor following each other than actually following their own path to the goal.

Michael Zhang
netID: mbz27

Leader Following

This was a fairly simple implementation. I had the leader agent follow wherever the specified destination is and all of the other agents' destinations were set to the position of the leader agent minus a constant times the vector velocity of the leader agent. This makes it so that the leader agent will always be in front of all of the agents.

Growing Spiral

This was also a fairly simple implementation. I simply got the individual agents' position and rotated their positions a certain amount until the agents started to spiral outward.

Pursue and Evade

This implementation was very crude and admitted did not have the best outcome. The pursuer agents had the simple implementation where it picks the closest evader agent to pursue. I used the leader-following behavior to have the pursuers follow the evaders. The evaders' implementation was that if a pursuer was close enough it will choose a random spot on the edge of the box that is furthest away from the closest pursuer and will set its destination to that spot. But this was fairly rudimentary and there were no forces that acted on the evader agents that actively avoided the pursuer agents while trying to get to the destination. This resulted in scenarios where the evader bumps into the pursuer agent while trying to get to the destination. If I had a bit more time I would have implemented a better mechanism to push the evader agents away from the pursuer agents.