Assignment 2, Due October 21st, 2011
(Worth 10%)

Create a new ros package by following these instructions:

```
cd ~/ros_workspace
roscreate-pkg potential_field roscpp geometry_msgs sensor_msgs nav_msgs bullet
cd potential_field
echo 'rosbuild_add_executable(${PROJECT_NAME} src/potential_field.cpp)' >> CMakeLists.txt
```

Then place potential_field.cpp into ./src/

Instructions: search for the 2 occurrences of "TODO" in the source code, and then address them.

1. **30%**) Use a potential field to guide a robot to a user selected destination. Use the following functions:
   a. **Attractive force:** $\|F_a\| = \gamma \cdot \left\|X_{goal} - X_{robot}\right\|^2$
   b. **Repulsive force (per sensor reading):**

$$\|F_r^i\| = \begin{cases} \dfrac{\alpha}{(d_i - d_{safe})^2} & \text{if} \quad d_{safe} + \varepsilon < d_i < \beta \\ \dfrac{\alpha}{\varepsilon^2} & \text{if} \quad d_i < d_{safe} + \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

   c. Use vector summation for the forces of all sensor readings as well the attractive force: $F = \sum_i \vec{F}_r^{\,i} + \vec{F}_a$

where γ, α, β, and ε are constants defined by you and d_{safe} is a short safety distance. For example d_{safe} = 0.5 m, $\gamma = \alpha = 1$, $\beta = 4$ m, and ε = 0.05 m.

Local Minima: In order for the robot to move out of local minima, introduce a random force. Monitor the robots position and increase the magnitude of the random force the more the robot stays near the same place.

From forces to velocity commands:

- **Angular velocity**: $\omega = \kappa(\theta_F - \theta_r)$, where κ is a scaling constant, θ_F is the orientation of the cumulative force, and θ_F is the orientation of the robot. *Hint*, use the atan2 function for your calculations, during subtraction take into account the 0, 2π discontinuity.
- **Linear velocity**: project the cumulative force along the orientation of the robot and use the projected force. Remember to truncate accordingly, to account for

maximum allowed speed. If the vector points behind the robot, set the linear velocity to 0.

Use the source code "*potential_field.cc"* as a starting point command line parameters are: "id #robots destX destY": the robot's id; the team size; the X coordinate of the destination; and the Y coordinate of the destination. Experiment with "*cave_single.world*", and "*obstacle_single.world*" worlds.

2. **20%**) Using the potential field of question 1, implement a random walk algorithm, based on potential fields, by selecting a new destination randomly at short intervals. The only modification is at regular intervals, every 5 seconds for example, create a new goal:
$X_{goal} = \left[x_r + r\cos(\theta) \quad y_r + r\sin(\theta)\right]^T$ where $r = rand(1) + 0.5$, and $\theta = rand(0, 2\pi)$
Use the code from question 1, no need to enter a destination. Experiment with "*cave_single.world*", and "*obstacle_single.world*" worlds.

3. **50%**) Use a potential field to guide a team of robots (use four robots) to swarm and follow a leader robot. The leader (robot0) should implement the behavior from question 2. The rest of the robots (followers) should use repulsive forces for obstacles and also for the other followers. Robots are detected by the laser scanner. Filter out the LIDAR returns from the leader (do not be repulsed by robot 0). The attractive force for robot "j" should be:

$$\left\|F_a^j\right\| = \begin{cases} \dfrac{\alpha}{d_{j\to 0}^2}, & \text{if } d_{j\to 0} < d_{safe} \\ 0, & \text{if } d_{safe} < d_{j\to 0} < d_{far} \\ \gamma \cdot d_{j\to 0}^2, & \text{otherwise} \end{cases}$$

where $d_{j\to 0}$ is the distance between the centers of robot$_j$ and robot$_0$ minus twice the radius of the two robots. A safe distance could be 1 m, and a too far distance would be 4 m. The above are approximate numbers.
Extend the code from question 2, create switch statement that modifies the behaviour depending on the Boolean isleader variable. If the robot is the leader then it uses the random walk from question 2. Otherwise, implement the follower behaviour. Run four potential_field programs in different terminals (or tabs). Experiment with "*cave_multi.world*", and "*obstacle_multi.world*" worlds.

What to submit:

In class submit a written report presenting screenshots of your work, provide a brief overview on how the program works and also discuss the choices made. Email the report together with the source code to the TA.