Autonomous Flight Engineer -- Technical Test

Asteroids!

You are a gunner on a spaceship in the game asteroids (www.kevs3d.co.uk/dev/asteroids/), required to determine the firing directions to destroy the asteroids around you. Unfortunately, your connection to the navigation team has been lost so you do not know the motion of your own ship. You must rely only on your sensors to determine where to shoot (e.g. you will only be able to estimate the relative state of the asteroids).

Dynamics

You know that your spaceship's pilot has three possible commands: „Thrust", „Rotate Left", and „Rotate Right". This means your spaceship follows the following discrete dynamics in a world fixed coordinate system, where α is the spaceship's heading:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ vx_{k+1} \\ vy_{k+1} \\ \alpha_{k+1} \end{bmatrix} = \vec{\boldsymbol{x}}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \vec{\boldsymbol{x}}_k + \begin{bmatrix} 0.5\, T_k\, \Delta t^2 \cos \alpha_k \\ 0.5\, T_k\, \Delta t^2 \sin \alpha_k \\ T_k \Delta t \cos \alpha_k \\ T_k \Delta t \sin \alpha_k \\ R_k \Delta t \end{bmatrix}$$

Where $T_k = \{0, 5\}\ \frac{m}{s^2}$, $R_k = \{-1\ , 0\ , 1\ \}\frac{rad}{s}$, and $\Delta t = 0.2s$. Whether an action is taken at each timestep is independent of the previous timesteps.

Asteroids move with constant velocity, meaning they have the following discrete dynamics in a world fixed coordinate system:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ vx_{k+1} \\ vy_{k+1} \end{bmatrix} = \vec{\boldsymbol{x}}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \vec{\boldsymbol{x}}_k$$

In „Asteroids" there are no disturbances, so both spaceship and asteroids follow the above dynamics exactly!

Sensors

Fortunately every asteroid has a unique color, so all sensor measurements include a unique ID for the asteroid it measured. All sensors have measurement noise on them, we leave it to you to determine a reasonable error model, and reasonable error parameters. You have the following sensors available.

Camera: Your spaceship has a 2D camera with 360 pixels which returns which pixel the asteroid is seen in. The camera also sees the sun, giving a measurement of heading (see Figure 1).

Radar: Your spaceship has a radar, which returns the distance of all the asteroids (see Figure 1)

Goal

Your goal is to determine the relative positions and velocities of the asteroids around you, in the coordinate frame of your ship (Frame XY in Figure 1). Based on this infromation, your ship will always fire at the asteroid with the highest remaining ID. Your bullets will move at a relative speed of 30 m/s with respect to your spaceship, at the time of firing, giving them the following velocity:

$$v_{bullet} = v_{ship} + 30\ m/s \begin{bmatrix} \cos(\theta + \alpha_{ship}) \\ \sin(\theta + \alpha_{ship}) \end{bmatrix}$$

The asteroids are 3 m in diameter, a shot anywhere on the asteroid destroys it, but it will only destroy the asteroid you aimed at. You may assume instantaneous knowledge of whether your shot destroyed the intended asteroid.

Evaluation

It is assumed that your solution will have two components, an *initialization* phase where a first estimate of the asteroids is made, and a *tracking* phase where the asteroid estimates are updated. Please provide your solution for the initialization phase in C++, and please provide a description of how you would complete the tracking phase (in any combination of text, equations, psuedocode, or C++). If you come up with a solution that does not have a clear initialization phase, please provide enough of your solution in C++ to demonstrate your skills.

You will be evaluated both on the quality of your approach and your implementation. You will be evaluated on:

1. Algorithm selection and implementation
2. Software organization, architecture, and reusability
3. Coding style, readability, and proper C++ usage
4. Code maintainability and efficiency

Please be ready to discuss your solution, both key decisions and overall approach as well as specific technical details of your implementation. We don't expect a perfect solution as your time is valuable, but demonstrate your skills in the solution and be ready to discuss what you would have done with more time.

Make sure to include complete instructions for compiling and running your solution.
To test your solution we have provided 6 sets of sensor data and correct firing directions which may be run via *main.cpp*. For three of the scenarios the true state of the ship and asteroids is also provided to aid you in debugging. Plotting functions are provided along with main.cpp to make debugging easier. Feel free to change (or improve) *main.cpp* as you need. *main.cpp* provides the framework for implementing the full solution, but only implement the initialization step unless you wish to do the full problem in C++.

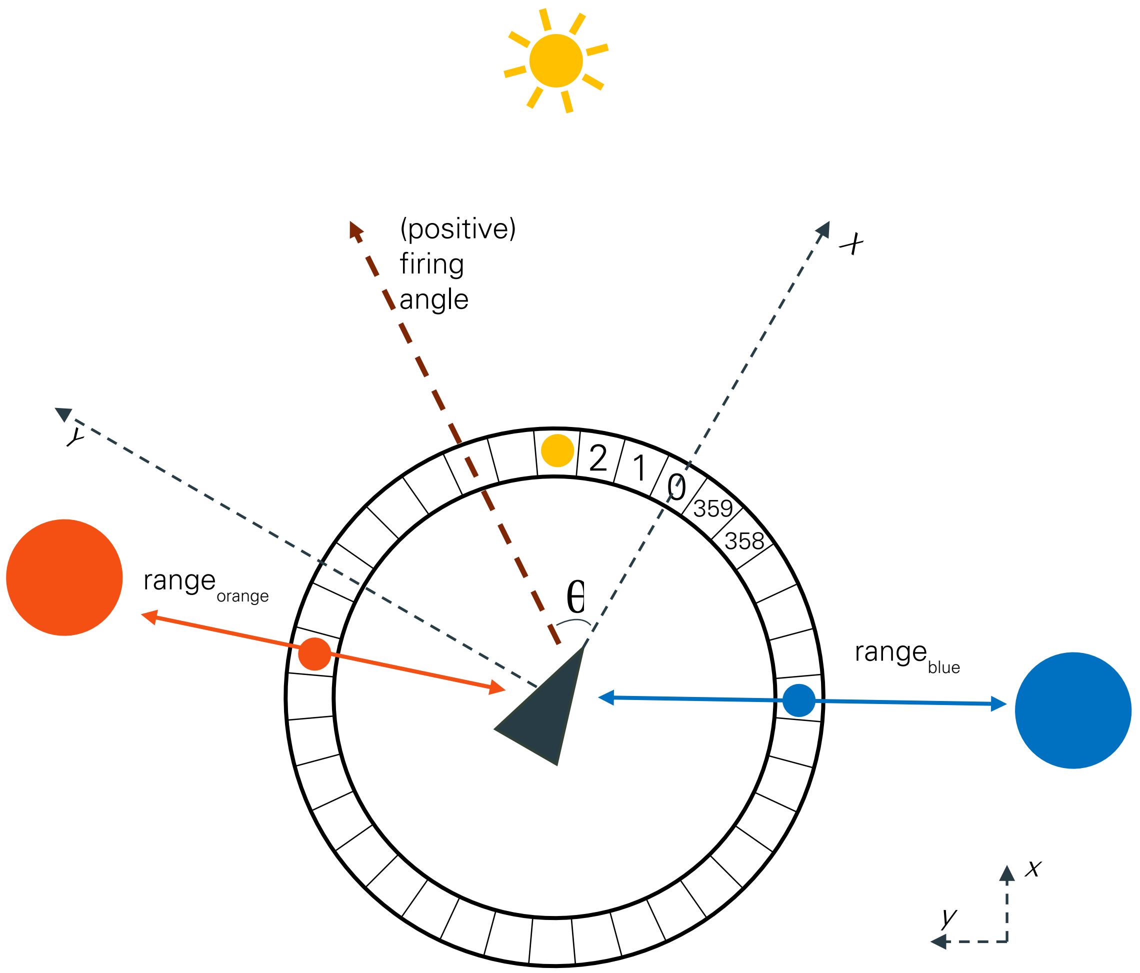

Figure 1: Diagram of asteroids game scenario. The ring marks a camera sensor with 360 pixels, indexed starting from 0 for the pixel at the front of the spaceship, and increasing counter clockwise. The camera measures which pixel it sees each asteroid in, as well as which pixel it sees the sun with. The radar returns the range to each of the asteroids. All measurements include the unique id of the asteroid it measured. The goal of the task is to determine the relative position and velocity of the asteroids in the ship's coordinate frame (frame XY)