24780 Engineering Computation: Problem Set 4

(*) In the following instructions (and in all course materials), substitute your Andrew ID wherever you see *yourAndrewId*.

You need to create a ZIP file (which may appear as a compressed folder in Windows) and submit the ZIP file via the 24-780 Canvas. The filename of the ZIP file must be:

```
PS04-YourAndrewID.zip
```

For example, if your Andrew account is hummingbird@andrew.cmu.edu, the filename must be:

```
PS04-hummingbird.zip
```

Failure to comply with this naming rule will result in an automatic 5% deduction from this assignment's credit. If we cannot identify the submitter of the file, an additional 5% credit will be lost. If we are ultimately unable to connect you with the submitted ZIP file, you will receive 0 points for this assignment. Therefore, ensure strict adherence to this naming rule before submitting a file.

The ZIP file must be submitted to the 24-780 Canvas. If you find a mistake in a previous submission, you can re-submit the ZIP file with no penalty as long as it's before the submission deadline.

Your Zip file should contain only two files, ps4-1.cpp and ps4-2.cpp. Do not include project files and intermediate files generated by the compiler. But, do not worry about some files or directories that are automatically added by the archiver (__MACOSX__ file for example).

Notice: The grade will be assigned to the final submission only. In the case of multiple file submissions, earlier versions will be discarded. Therefore, when resubmitting a ZIP file, it MUST include all the required files. Also, if your final version is submitted after the submission deadline, the late-submission policy will be applied, regardless of how early your earlier version was submitted.

Ensure that your program can be compiled without errors on one of the compiler servers. Do not wait until the last minute, as the compiler servers may become very busy just minutes before the submission deadline!

Submission Due: Please refer to Canvas.

START EARLY!

Unless you are a good programmer, there is no way to finish the assignment overnight.

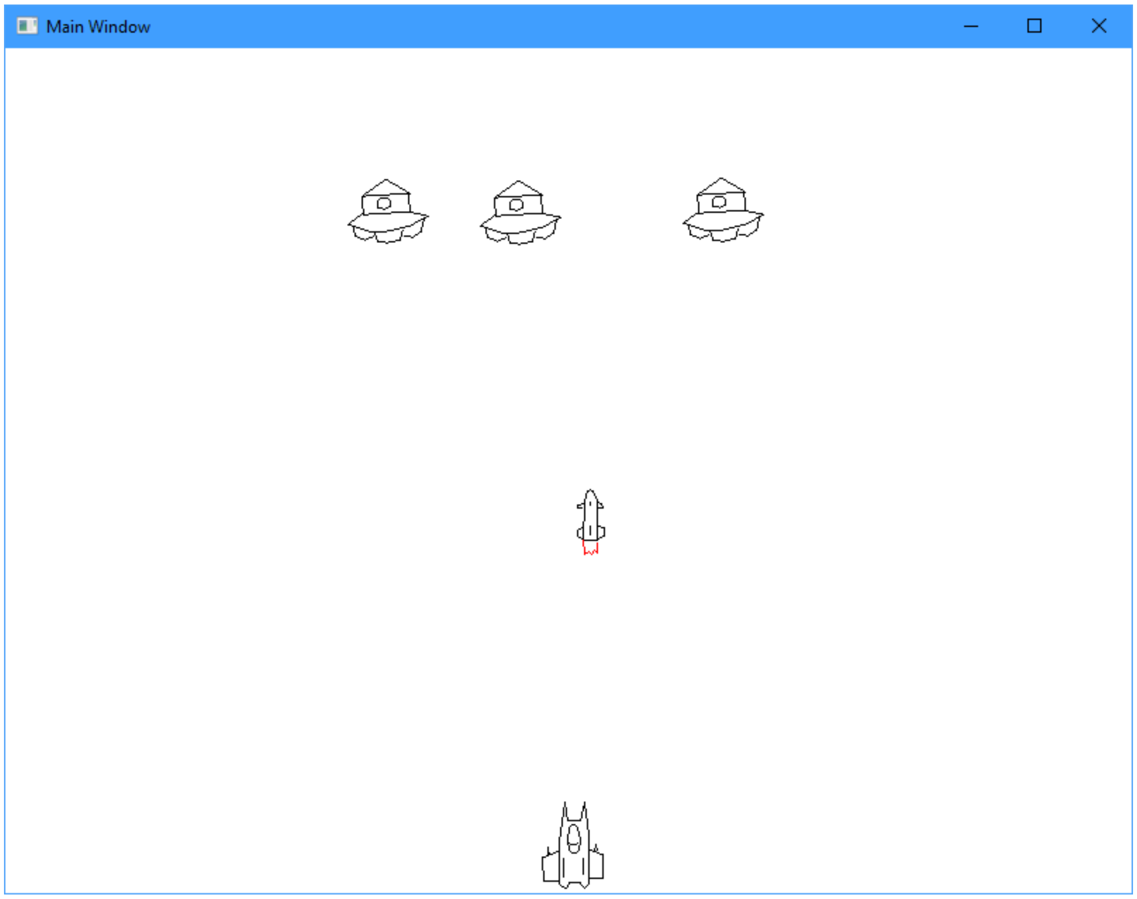

Fig. 1: Shooting game with wire-frame graphics

PS4-1 Wireframe Graphics for the Shooting Game [ps4-1.cpp] (30 pts)

Download the base code called ps4-1.cpp. It is a slightly modified version of the shooting game we did in class.

Also find spaceship.cpp, ufo.cpp, and missile.cpp. Each includes a function to draw a spaceship, UFO (or target), and misile respectively.

Cut & Paste those functions into ps4-1.cpp, and use them to render (slightly) better-looking shooting game.

Save your source file as ps4-1.cpp and include in the zip file you submit to Canvas.

Fig. 1 shows a sample rendering.

PS4-2 Cannonball Game [ps4-2.cpp] (70 pts)

When you start the program, it opens a graphics window of the size 800x600 pixels. This window represents an 80x60-meter rectangular region on a vertical plane.

In the window are the following objects drawn with different colors:

- a cannon (blue square and a line segment),
- a cannonball (a circle),
- a target (a red square), and
- five obstacles (five green (initially) or yellow (after taking a cannonball hit) rectangles).

The goal of the game is to hit the target with the fewest cannonballs. If there is no clear path to hit the target, you may shoot an obstacle to destroy it, but you waste cannonballs. The game ends when you hit the target. A cannonball is subject to the gravitational force, but there is no other force applied to the cannonball - this makes the trajectory a perfect parabola.

An obstacle can be destroyed by two cannonballs. The color of the obstacle should be green initially. When one cannonball hits an obstacle, the obstacle changes color to yellow. An yellow obstacle disappears when another cannonball hits it.

Although the trajectory can be analytically computed with little difficulty, since one of the purposes of this assignment is to let you practice numerical integration, calculate the trajectory using a numerical integration scheme such as the Euler's method. If you want to study and try a higher-order scheme such as Runge-Kutta method, it is also ok.

The cannon is placed near the lower left corner and initially points to the upper right corner. The direction of the cannon is adjusted by pressing up and down allow keys (FSKEY_UP and FSKEY_DOWN). One key stroke of an up/down arrow key must rotate the cannon by 3 degree. Maximum angle should be 90 degrees (straight up), and the minimum 0 degree (horizontal).

You shoot a cannon by pressing the space bar (FSKEY_SPACE). The initial speed of the cannonball is 40 m/sec. The trajectory of the cannonball should be animated.

When the game is started, the program places five rectangular obstacles at random locations. The width and height of each obstacle should range between 8m and 15m, randomly selected. Note that the effective size of the obstacle (size visible on the window) must be between 8m and 15m. For example, if the left edge of the obstacle is located 4m from the right edge of the window, effective width is only 4m, and therefore it does not satisfy the requirement.

The color of the cannonball needs to change based on how many cannonballs have been shot. The first cannonball should be blue (R,G,B)=(0,0,1), the second cyan (0,1,1), the third yellow (1,1,0), the fourth magenta (1,0,1), and the fifth and on red (1,0,0).

The target should be moving along the right vertical wall, and its size is 10m x 10m. Initially the top edge of the target is 60m high from the bottom edge of the window, and it should be moving down at 10m/sec. When the bottom edge of the target touches the bottom edge of the window, reverse the velocity and go up until the top edge of the target hits the top edge of the window. Then reverse the velocity and repeat.

None of the obstacles and the target should overlap the cannon, but they can overlap each other. If an obstacle and the target overlaps, the target should be visible (in other words, the target can hide a part of an obstacle, but no obstacle hide a part of the target).

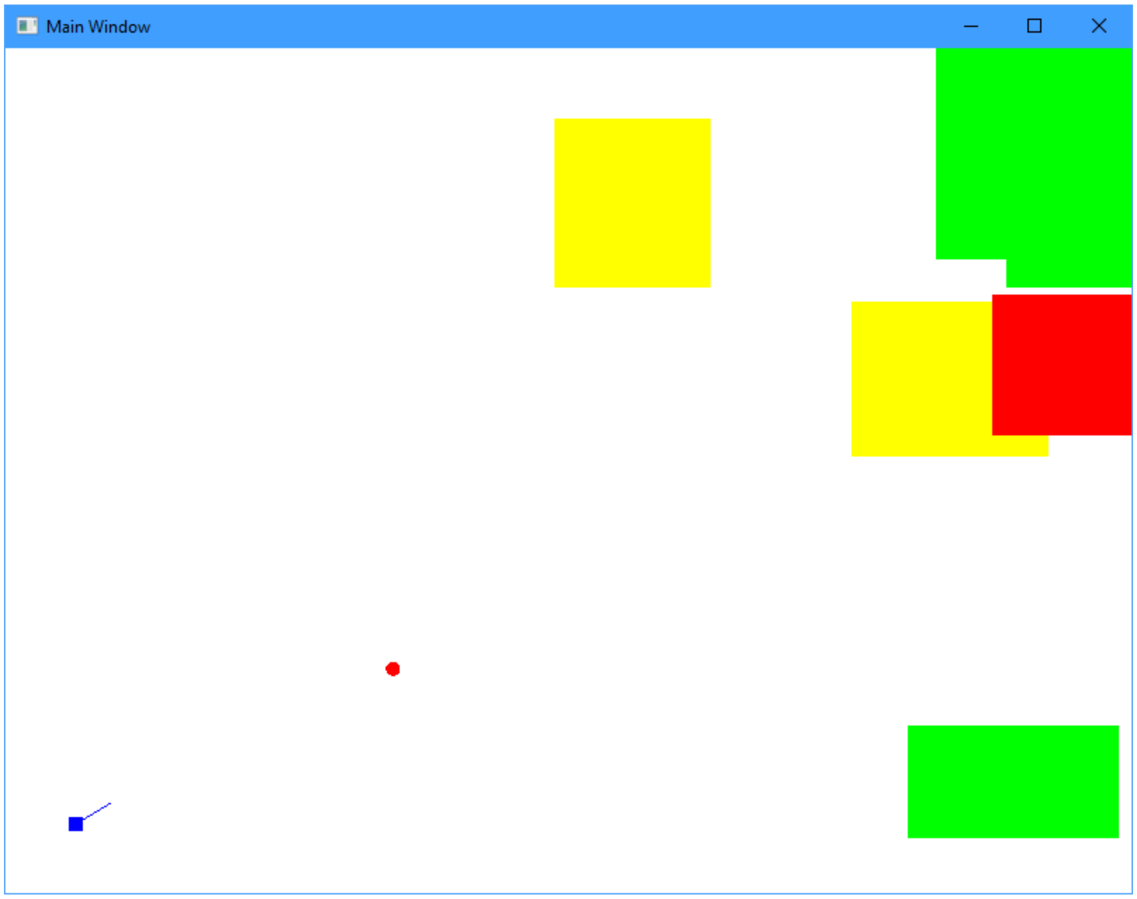

Fig. 2: Cannonball Game

Write the C++ program named ps4-2.cpp. The .cpp file must be included in the Zip file submitted to Canvas.

Extra 5 points: Remember positions of the cannonball of the last 10 frames, and draw a GL_LINE_STRIP, so that a cannonball appears to have a tail. The color of the line strip should be same as the cannonball.

Note that if you end up making an error in an attempt to get this 5 extra points, you may rather lose points.

Fig. 2 shows a sample rendering.

Test Your Program with One of the Compiler Servers

Test your program with one of the following compiler servers:

```
http://freefood1.lan.local.cmu.edu
http://freefood2.lan.local.cmu.edu
http://freefood3.lan.local.cmu.edu
http://freefood4.lan.local.cmu.edu
```

You need to make sure you are not getting any errors (red lines) from the compiler server.

It is a good practice to remove warnings as well. However, we will not take points off for warnings as long as your program satisfies requirements of the assignment.

You can only access these servers from CMU network. If you need to access from your home, use CMU VPN. Please visit the CMU computing services web site how to install the VPN.