Assignment 2: Rasterization

CS 4600 Computer Graphics
Fall 2019
Ladislav Kavan

In this assignment, we will explore basic image rasterization algorithms. This is an individual assignment, i.e., you have to work independently. All information needed to complete this homework is covered in the lectures and discussed at our Canvas Discussion Boards. You shouldn't have to use any textbooks or online resources, but if you choose to do so, you must reference these resources in your final submission. It is strictly prohibited to reuse code or fragments of code from textbooks, online resources or other students -- in this course, this is considered academic misconduct (https://www.cs.utah.edu/academic-misconduct/). Do not share your homework solution with anyone -- this is also treated as academic misconduct in this course, even if nobody ends up copying your code.

The framework code is written in C++ with the following dependencies:

- OpenGL 1.0
- GL Utilities (GLU)
- C++ STL
- OpenGL Extension Wrangler Library (GLEW)
- GLFW3

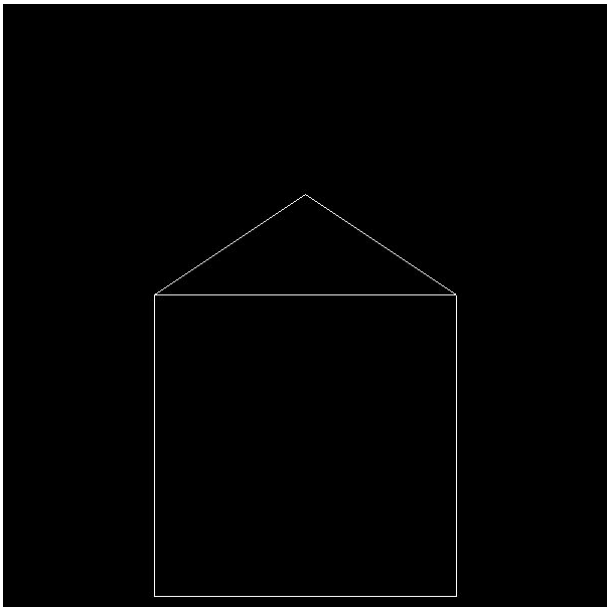
The recommended IDE is Visual Studio 2017 Community Edition, which is available free of charge for educational purposes. The framework code provides precompiled dependencies for Visual Studio 2017. If you choose to use a different platform or IDE version it is your responsibility to build the dependencies and get the project to work.

The assignment should be implemented inside the provided *main.cpp* file using the specified subroutines. No other source code/dependencies/libraries are needed or allowed for this assignment. The provided source code, after being successfully compiled, linked, and executed, should display a black image. Your task will be to draw something into this image!

1. Line rasterization (35 points)

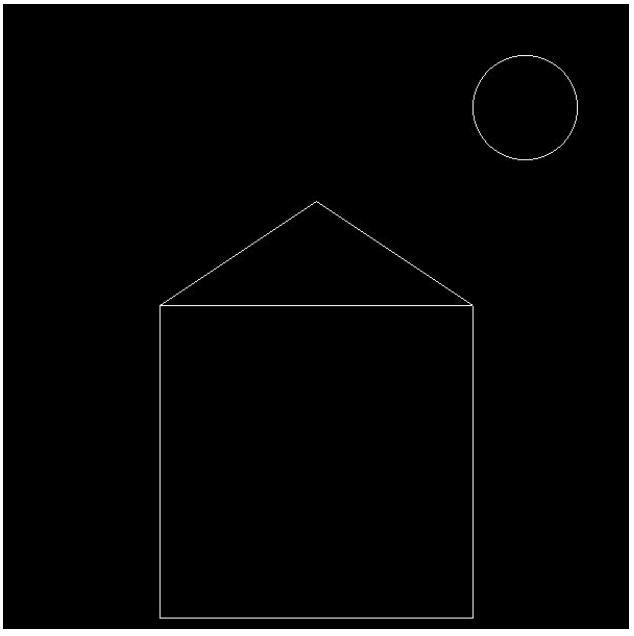
Your first task is to complete the function *drawLine(int x1, int y1, int x2, int y2)*. Using the Bresenham algorithm for line rasterization, this function should draw a line from pixel (x1, y1) to pixel (x2, y2). Warning: typical formulations of the Bresenham algorithm support only certain combinations of points (x1, y1) and (x2, y2). Your function is supposed to be general and support all possible inputs (x1, y1) and (x2, y2). The individual pixels of the line should be written into the buffer *g_image*. You may use the provided function *putPixel(int x, int y)* which writes a white pixel into *g_image* at point (x, y). Your line drawing algorithm needs to be able to handle lines of all directions, but you can assume that both of the points (x1, y1) and (x2, y2) are inside the dimensions of *g_image* (600 x 600). After you've completed the

function *drawLine*, the provided function *drawImage* should draw a simple image on the screen. The same image will be also saved into file “data/out.ppm”. Your window should look like this.



2. Circle rasterization (35 points)

The next task is to implement Bresenham’s algorithm for circle rasterization inside function d*rawCircle(int x0, int y0, int R)*, where (x0, y0) is the center of the circle and R is the radius. Just like in Task 1, you can assume that none of the pixels of the circle will lie outside of the g_image window. After completing Task 2, your window should look like this.



3. Interactive Drawing (30 points)

The next task is to add mouse and keyboard controls to make your program interactive. This feature will be helpful to test if your *drawLine* and *drawCircle* functions work correctly for arbitrary lines and circles.

Your program should be able to:

- Color the pixel white after a mouse click
- Use the L key to switch on the line mode
- Draw a line after 2 mouse clicks in the line mode
- Use the C key to switch on the circle mode
- Draw a circle after 2 mouse clicks in the circle mode. The first clicked point should be the center of the circle and the second clicked point should be a point on the circle.

You should complete the functions *glfwMouseButtonCallback* and *glfwKeyCallback* and the member functions of struct *line* and *circle* to achieve this.

4. Extra Credit (up to 20 bonus points at instructor's discretion)

If you are looking for additional challenges, implement an interactive drawing tool for Bezier curves and use it to rasterize a new image that will use lines, circles, and curves in a creative way. The points for the Bezier curve control polygon should be selected using the mouse clicks. Render the control polygon as well as the Bezier curve.

5. Submission

When you're finished with Tasks 1, 2 and 3, you'll need to submit the following:

- Source code (you should only modify the main.cpp file). The main.cpp file is all we need -- please do not submit any other files, especially NOT .exe files and other files created by Visual Studio.
- PDF document describing what you did, screenshots are welcome. If you used any textbooks or online resources that may have inspired your way of thinking about the assignment, you must reference these resources in this PDF document
- Your resulting image "out.ppm" showcasing your line and circle rasterization routines.
- [optional] If you have succeeded with Task 4, submit the code "main-extra.cpp" and an image "out-extra.ppm" showing your piece with ellipses. This way your solution of Tasks 1, 2 and 3 are clearly separated from the extra credit Task 4 -- this helps to prevent confusion when grading.

Please pack all of your files to a single ZIP file named Lastname_Firstname_HW2.zip

Please submit the ZIP file(s) via Canvas and in the comments specify how many late days you're using and have remaining (e.g. "Used 0 late days, 5 remaining.") Failure to comply

with submission instructions may result in loss of points (up to zero in cases of severe negligence).

Late policy

Everyone has 5 late days to cover circumstances such as minor illness, competitions, family matters, etc. You can distribute your late days arbitrarily among assignments (e.g. you can use 0 late days on HW assignment 1 and all 5 late days on HW assignment 2, leaving no late days left for subsequent assignments). After using up all of your late days, a 0.1*X*number_of_late_days will be deducted from your points, where X is the pre-penalty number of points. Each late day is defined as 24 hours after the submission deadline on Canvas, there is no distinction between weekdays and weekends or holidays. Requests for exceptions to this policy (e.g., due to major medical conditions) must be made in writing, with a date and a signature of the student or their legal representative, and include documentation pertaining to the case (e.g., a note from the University Center for Disability & Access).