IOS
Fakultät für Informatik
Labor für Computergrafik
Prof. Dr. G. Umlauf
HTWG
Hochschule Konstanz
University of Applied Sciences

Konstanz, 23.09.2025

Assignment 1

Computer graphics

Deadline 05.11.2025, F033

Programing frame-work:
Download the programing frame-work from the web-page of the lecture. The zip-file contains the framework for the exercises and a recent version of **`freeglut`** (plus some **`x64`**-binaries).

- If you are using Microsoft Visual Studio: Open the project file with Visual Studio and compile/start the program. A window similar to Figure 1 should pop up.
- If you are using Qt: Open the .pro-file, chose your favorite configuration (switch off the shadow build), and run qmake.
- Otherwise, install the appropriate **`freeglut`**-version on your machine, set up a new console application in your C++-IDE, add the **`cpp`**-file of the framework to the project, and compile/start the program. A window similar to Figure 1 should pop up. You might have to adapt the include paths in the **`cpp`**-file depending on your **`freeglut`** installation.

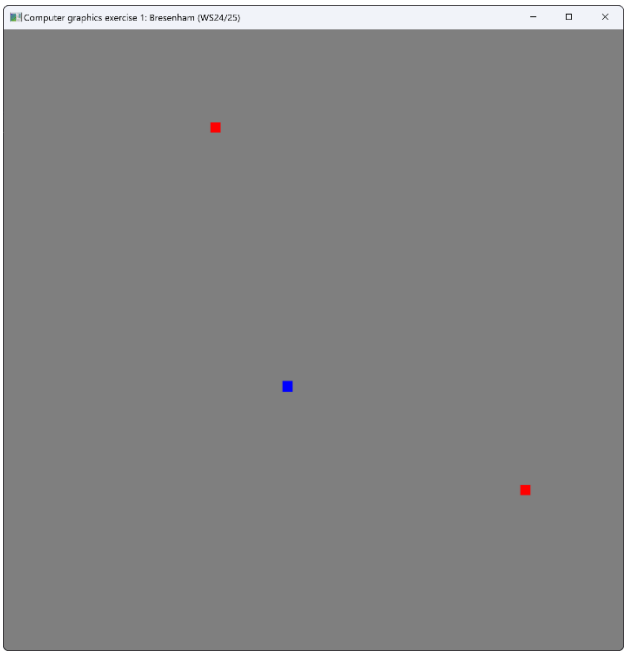

Figure 1: Programing frame-work.

If the framework compiles, the program should open a window that look like Figure 1.

The provided files contain some program code necessary for the display of your results. In particular there is some code to enlarge the displayed results. This is necessary since modern screens have such large resolution such that programming at pixel resolution would yields hardly recognizable results.

Important for the exercises are the following functions

- `clearImage(Color c)`: Clears the display and fills it with color `c`.
- `setPoint(Point p,Color c)`: Use this function to draw a "pixel".
- `display()`: Triggers a re-draw of the viewport/display.
- `bhamLine(Point p,Point q,Color c)`: Draws a line from `p` to `q` with color `c`.
- `bhamCircle(Point p,int r,Color c)`: Draws a circle at `p` with radius `r` with color `c`.

The display-area is bounded by the coordinates $[-50;\ 49] \times [-50;\ 49]$. If you draw a pixel outside this area, an error message is printed on the console window.

Exercise 1 (Bresenham algorithm for lines) — 5 points

Part 1:
Implement the Bresenham algorithm for lines in the first octant by completing the function

```
BesenhamLine(Point p, Point q, Color c).
```

This function is invoked by the function `display()`.

Part 2:
Generalize the above function to arbitrary octants by reflections of the start- and end-point of the line at the x- and y-axis.

Exercise 2 (Bresenham algorithm for circles) — 5 points

Part 1:
Implement the Bresenham algorithm for circles in the second octant around the origin only by completing the function

```
BesenhamCircle(Point p, int r, Color c)
```

This function is invoked by the function `display()`.

Part 2:
Generalize the above function to draw a complete circle by reflection of the output pixels and around arbitrary center points and radii.