



## Bildgenerierung

Wintersemester 2023 / 2024

### Übungsblatt 1

#### Aufgabe 1 (*Test des Frameworks*)

Für die Bearbeitung der Übungsblätter existiert auf den Rechnern des CIP-Clusters in Raum G.14.11 ein Qt-Framework. Dieses befindet sich im Verzeichnis

```
/home/bildgen/cppqt
```

Um damit arbeiten zu können, gehen Sie wie folgt vor:

- a) Falls in Ihrem Home-Verzeichnis nicht bereits ein Unterverzeichnis mit dem Namen `bin` existiert, erstellen Sie es mittels

```
cd ~  
mkdir bin
```

Sie müssen sich danach neu einloggen, damit das Verzeichnis dem Suchpfad hinzugefügt wird.

- b) Erzeugen Sie einen symbolischen Link auf das Compiler-Script `g++drawqt` in Ihrem soeben erzeugten Verzeichnis `~/bin`:

```
ln -s /home/bildgen/cppqt/g++drawqt ~/bin
```

- c) Ein Beispielprogramm finden Sie unter

```
/home/bildgen/cppqt/bsp1.cc
```

Kopieren Sie das Beispielprogramm in Ihr Home-Verzeichnis und rufen Sie das Compiler-Script mit der Quelltext-Datei des Beispiels als Parameter auf, um es zu kompilieren:

```
cp /home/bildgen/cppqt/bsp1.cc .  
g++drawqt bsp1.cc
```

Erlaubte Dateierendungen sind `c`, `cc`, `cpp` und `c++`.

- d) Das erzeugte Programm trägt den Namen der Quelltextdatei ohne deren Endung. Testen Sie das Beispielprogramm:

```
./bsp1
```

Eine vollständige Auflistung der Funktionalität des Qt-Frameworks finden Sie unter:

```
/home/bildgen/cppqt/doc/index.html
```

Es existiert noch ein weiteres Compiler-Script namens `g++drawqt-g`, welches mit Debug-Symbolen kompiliert.

## Aufgabe 2 (Einfache Linien)

---

Schreiben Sie ein Programm, das

1. ein zunächst leeres Bild erzeugt,
2. zwei Punkte einliest,
3. eine Linie zeichnet, die beide Punkte verbindet,
4. und 2 bis 3 solange wiederholt, bis der Anwender negative Koordinaten eingibt.

Implementieren Sie zum Zeichnen der Linien den *naiven* Algorithmus aus dem Skript (oder ggf. einen eigenen *naiven* Algorithmus) aber noch nicht die optimierten (inkrementellen) Varianten. Kommentieren Sie in Ihrem Programm, wie der Algorithmus funktioniert.

Verwenden Sie hierzu nur die Funktion `Drawing::drawPoint()` und *nicht* `Drawing::drawLine()`.

## Aufgabe 3 (Einfache Kreise)

---

Schreiben Sie ein Programm, das

1. ein zunächst leeres Bild erzeugt,
2. einen Mittelpunkt  $m$  und einen Radius  $r$  einliest und
3. einen *gefüllten* Kreis um  $m$  mit Radius  $r$  malt.

Verwenden Sie nicht den optimierten (inkrementellen) Kreisalgorithmus, sondern lassen Sie ihren Algorithmus auf den *naiven* Varianten aus dem Skript (oder ggf. einer eigenen *naiven* Variante) beruhen. Kommentieren Sie in Ihrem Programm, wie der Algorithmus funktioniert.

Verwenden Sie nur die Funktion `Drawing::drawPoint()` und *nicht* `Drawing::drawCircle()`.

**Hinweis:** Für den Zugriff auf die Ausbildungsrechner müssen Sie sich immer eine VPN-Verbindung einrichten (<https://zim.uni-wuppertal.de/de/dienste/netzzugang/vpn.html>), für Linux per SSH und für Windows z.B. per *SmarTTY* (siehe <https://sysprogs.com/SmarTTY/>).

**Abgabe:** Mi., 25.10.2023, 08:00 Uhr

Senden Sie Ihre Lösungen der Theorie-Aufgaben und Ihre Programme per E-Mail an [bildgen@studs.math.uni-wuppertal.de](mailto:bildgen@studs.math.uni-wuppertal.de).