Übung 1 (Graphisch-Interaktive 2D-Systeme)

a) Laden Sie zunächst von Moodle die Datei „Plotter.zip“ herunter mit dem Codegerüst. Programmieren Sie auf dieser Basis unter Verwendung der 2D-API von Qt einen einfachen Funktionsplotter, welcher die beiden in der Vorlesung besprochenen Funktionen für Sinus und Kreis darstellen kann. Im wesentlichen brauchen Sie nur die Datei „canvas.cpp“ zu editieren, indem Sie hierfür u.a. die Methode *paintEvent()* entsprechend der TODOs ergänzen.

Beginnen Sie mit der Sinus-Funktion. Berechnen Sie dazu die Funktionswerte via $y = \sin(x)$ und transformieren Sie die für den gewählten Bereich berechneten (x, y)-Paare in Pixelkoordinaten. Wählen Sie zum Testen im User Interface geeignete Welt- bzw. Problemkoordinaten.

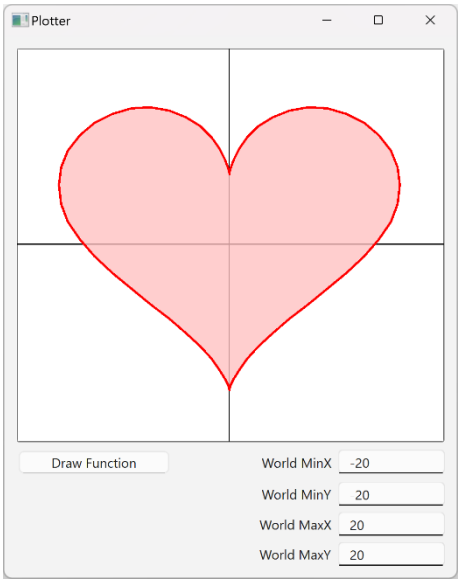
Bei Klick auf „Draw Function“ wird die Funktion für den eingegeben Wertebereich gezeichnet. Nutzen Sie zum Zeichnen die Methoden *moveTo()* und *lineTo()*, wobei Sie die resultierenden (x, y)-Werte, welche aus Effizienzgründen nicht allzu dicht nebeneinander liegen sollten, über Liniensegmente verbinden sollen.

Damit man zwischen unterschiedlichen Funktionen zur Laufzeit umschalten kann, sollten Sie allerdings noch eine `QComboBox` zur Funktionsauswahl vorsehen. Falls Sie bislang noch nie mit den GUI-Komponenten der Qt-Bibliothek gearbeitet haben sollten, dürfen Sie die GUI stattdessen notfalls auch um zwei weitere Buttons ergänzen.

b) Zum Valentinstag möchten Sie Ihrem bzw. Ihrer Liebsten stimmungsvolle Graphikgrüße in Herzform schicken. Dazu haben Sie im Internet folgende parametrische Beschreibung einer Herzkurve gefunden (für Parameter $t \in [0, 2\pi]$), die Sie hier als Grundlage nehmen:

$$\vec{s}(t) = \begin{pmatrix} 16\ sin^3(t) \\ 13\ \cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t) + 2 \end{pmatrix}$$

Zeichnen Sie diese Herzkurve (in der Abbildung rechts wurde bereits ein für die Funktionsdarstellung geeigneter Bereich ausgewählt – die Kurvendiskussion können Sie sich damit ersparen).



c) Erweitern die die Canvas-Klasse noch um die drei in den Vorlesungsfolien gezeigten Methoden zur Behandlung von Maus-Events. Fügen Sie weiterhin Ihrem User Interface ein zusätzliches `QLabel` (zur dynamischen Anzeige von Werten) hinzu.

Klickt ein User nun innerhalb der Zeichenfläche irgendwo hin (insbesondere z.B. auf die dargestellte Kurve), so soll der in Pixelkoordinaten gegebene Klickpunkt umgerechnet werden in Weltkoordinaten, welche im Label angezeigt werden sollen. Dazu benötigen Sie natürlich die inverse Window-Viewport-Transformation.