Projektprüfung: Einführung in die Softwareentwicklung

Wintersemester 2020/21 · 22. – 25. Februar 2021

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Anleitung zur Konsolenbibliothek „ConsoleWindow.zip“

Veröffentlicht: 16. Februar 2021

Worum geht es?

In der Projektprüfung zur Einführung in die Softwareentwicklung im Wintersemester 2020/21 wird eine Aufgabe gestellt, bei der eine interaktive Anwendung entwickelt werden soll, die Ausgaben auf einer Konsole machen soll (ähnlich zu der Aufgabe des Verkehrssimulators aus den Übungen) und dabei auf Tastaturbefehle reagieren soll (während die Anwendung weiter läuft). Dies ist mit den Boardmitteln von C++ (`std::cin` / `std::cout`) kaum möglich, und es gibt kaum *einfache* Bibliotheken, die auf allen relevanten Personalcomputer-Plattformen gleich funktionieren.

Aus diesem Grunde stellen wir eine Bibliothek (`ConsoleWindow.h`/`.cpp`) zur Verfügung, die wir für diese Prüfung selbst entwickelt haben. Sie erlaubt es, Zeichen auf dem (virtuellen) Konsolenbildschirm (der in einem Fenster dargestellt wird) einzeln zu setzen und zu lesen. Außerdem kann man jederzeit abfragen, welche Tasten gedrückt sind. Um alles so einfach und verständlich wie möglich zu halten, wird lediglich eine Ausgabe von ASCII-Zeichen in schwarz-weiß unterstützt. Gleiches gilt für die Tastatureingabe, wobei zusätzlich die Abfrage der Cursortasten möglich ist.

Die Bibliothek im beiliegenden ZIP-Archiv enthält ein einfaches Demo-Programm (`ConsoleDemo.h`/`.cpp`, `main.cpp`), welches zufällige Zeichen ausgibt und ein (rudimentäres) Scrollen des Bildschirms erlaubt (der Rahmen des ebenfalls angezeigten Anleitungstextes des Demos wird dabei mit auf dem Bild „verschmiert“; das ist kein Bug sondern so gewollt).

Die Bibliothek ist in C++ mit Qt 5 geschrieben (benötigt wird min. 5.2.x, wir empfehlen 5.12.x).

Die Benutzung in der Projektprüfung ist freiwillig – Sie können eine ähnliche Funktionalität auch gerne selbst mit Qt implementieren (die gesamte Qt-Bibliothek in einer Version 5.x.x darf in der Projektprüfung benutzt werden). Zu beachten ist aber, dass eine etwaige eigene Lösung plattformübergreifend auf Mac/Windows/Linux-Systemen funktionieren muss, damit wir bei der Korrektur nicht auf ein spezielles System festgelegt sind, was wir nicht garantieren können (daher die Nutzung von Qt). Wir erwarten, dass die meisten Lösungen die vorgegebene Bibliothek aus Gründen der Einfachheit nutzen werden. Daher stellen wir die Bibliothek vorab zur Verfügung, damit Sie sich schonmal damit vertraut machen können, und die Übersetzung mit Qt 5.x.x einrichten können.

Der gesamte Code darf in der Prüfung auch ohne Quellenangabe frei verwendet werden (aufgerufen, kopiert, abgewandelt), und kann so auch als Ausgangspunkt für eine eigene Lösung dienen.

Bugs?

Rückfragen (oder Bug-reports) können im MS-Teams Kanal „Abschlussprüfung“ des EIS-WS20/21-Teams gestellt (übermittelt) werden.

Dokumentation der API

Die Konsole ist in der Datei `ConsoleWindow.h` definiert. Dort wird die Qt-Klasse `ConsoleWindow` definiert, die von `QtMainWindow` abstammt. Man kann sie also als Hauptfenster einer Anwendung benutzen (der Demo-Code zeigt, in main.cpp, wie dies geht). Denken Sie beim Übersetzen daran, dass der „`moc`"-Preprozessor benutzt werden muss. Die beigelegten `qmake` bzw. `cmake`-Dateien sind bereits so konfiguriert, dass dies automatisch geschieht.

Der Code ist grob dokumentiert – die Kommentare sind zwar rudimentär, aber die API ist wirklich sehr, sehr einfach. Es sollte daher kein Problem sein, sie zu verstehen.

Hier nochmal das wichtigste zusammengefasst:

```
class ConsoleWindow : public QMainWindow {
   ...
 private:
   ...
```

(interne Details, nicht wichtig für die Benutzung)

```
 protected:
    virtual void onRefresh() = 0;
```

Diese Funktion ist wahrscheinlich als einzige etwas ungewöhnlich: Sie wird 20x pro Sekunde automatisch aufgerufen (mit einem `QTimer`). Eine konkrete Instanz von `ConsoleWindow` muss diese überschreiben (Sie müssen also eine Unterklasse erzeugen – siehe Demo-Code).

Jedes Mal hat die Anwendung die Aufgabe, den Bildschirm aktuell zu halten und auf Tastarturbefehle zu reagieren. Ihre Anwendung läuft also nicht in einer kontinuierlichen Schleife, sondern Ereignisorientiert: Wenn Sie unsere Bibliothek nutzen, müssen Sie die Berechnungen in kleine „Häppchen" zerteilen, die jeweils nur Sekundenbruchteile dauern dürfen (wenn es länger als 50msec dauert, ruckelt die Ausgabe).

```
 public:
    ConsoleWindow(QWidget *parent = nullptr, unsigned width = 64,
                  unsigned height = 48);
```

Der Konstruktor lägt ein Fenster mit vorgegebener Anzahl Zeichen (widt h= Anzahl Spalten, height = Anzahl Zeilen) an.

```
    void setCharacter(int x, int y, char character);
    char getCharacter(int x, int y);
```

Mit get- und setCharacter() können Sie einzelne Zeichen auf den Bildschirm schreiben. Sie können alle ASCII-Zeichen benutzen, z.B. mit Character-Konstanten 'A' (Achtung: In C++ keine "Gänsefüsschen", das geht übel schief). Der Inhalt des Bildschirms wird nach jedem onRefresh()-Event in das QtFenster übertragen, dass Sie auf Ihrem Monitor sehen.

Es gibt auch einige rudimentäre „Komfortfunktionen":

```
void writeString(int x, int y, std::string text);
```

schreibt eine längere Zeichenkette, und

```
void clear(char character = ' ');
```

löscht den gesamten virtuellen Bildschirm (setzt ein festes Zeichen, Standard ist das Leerzeichen).

Mit der Funktion

```
char getPressedKey();
```

können Sie erfragen, welche Taste gerade auf der Tastatur gedrückt ist. Die Abfrage wird direkt vor dem Aufruf on „onRefresh()“ das letzte Mal aktualisiert. Unterstützt werden nur ASCII-Zeichen (d.h. z.B. keine Umlaute oder F-Tasten). Für Cursortasten gibt es die vier Spezialcodes

```
static const char CURSOR_LEFT = 1;
static const char CURSOR_RIGHT = 2;
static const char CURSOR_UP = 3;
static const char CURSOR_DOWN = 4;
```

sowie die Null

```
static const char NO_KEY = 0;
```

falls keine Taste gedrückt ist.

Übung zur Vorbereitung

Wir empfehlen eine kleine Anwendung als Vorbereitung zur Prüfung zu implementieren. Eine schöne Aufgabe, die gut zur Vorbereitung auf die Prüfung geeignet ist, aber schnell erledigt werden kann, ist die Implementation des bekannten Computerspiels „SNAKE“:

https://de.wikipedia.org/wiki/Snake_(Computerspiel)

Einige von Ihnen kennen es vielleicht als kleines Spiel, das mit Geräten wie z.B. den meisten Feature-Phones der Firma Nokia mitgeliefert wurde.