Grundlagen der Programmierung 2: Praktikumsaufgabe 5

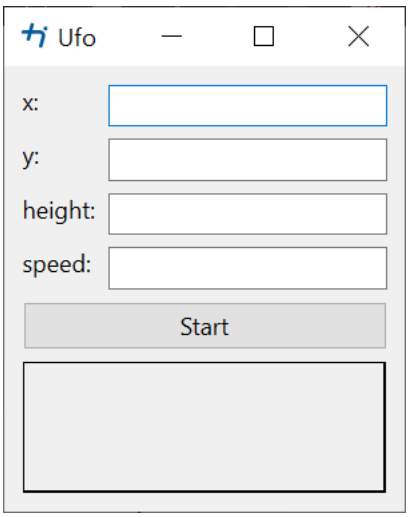
Prof. Dr. Robert Gold

Technische Hochschule Ingolstadt
Sommersemester 2024

In dieser Aufgabe wird eine einfache GUI mit Qt für Flüge mit Vertical-Ufos erstellt. Die GUI besteht nur aus einem Fenster. Von den bereits erstellten Klassen wird nur die Klasse `VerticalThread` geändert. Es werden ein Hauptprogramm und zwei Klassen `MainWindow` und `MainWidget` für die GUI hinzugefügt. Bevor Sie beginnen, sollten Sie die Dateien *vertical_thread.h* und *vertical_thread.cpp* der vierten Aufgabe am besten unter anderen Namen z.B. *pa4_vertical_thread.h*, *pa4_vertical_thread.cpp* kopieren.

Es gibt keine neuen Unit-Tests für diese Aufgabe. Die GUI soll manuell mit verschiedenen Testflügen getestet werden. Die Abgabe erfolgt aber trotzdem über APA. Bitte auch die nicht geänderten Dateien abgeben.

a) Zuerst erstellen wir ein Fenster, in dem nach dem Programmstart die Werte für *x*, *y*, *height* und *speed* (Zielkoordinaten, Flughöhe, Fluggeschwindigkeit) eingeben werden können. Der Button dient zum Starten des Fluges. Unter dem Button soll sich ein `QLabel` befinden, in dem ein dreizeiliger Text angezeigt werden kann.



Erstellen Sie die GUI in den Dateien *ui_main.cpp*, *ui_window.h* (Klasse `MainWindow`) und *ui_widget.h* (Klasse `MainWidget`).

Den Rahmen um den `QLabel` unterhalb des Buttons erhalten sie durch Aufruf der Methoden

```
plabel->setFrameShape(QFrame::Box);
plabel->setFrameShadow(QFrame::Plain);
```

(wenn der Label mit `plabel` bezeichnet ist). Initialisieren Sie den Text dieses Labels mit `"\n\n\n"`.

Achten Sie darauf, dass das Fenster genauso aussieht, wie in den Abbildungen gezeigt!

Das Starten des Fluges mit Hilfe von Signal und Slot wird erst in den nächsten Teilaufgaben implementiert. Das Programm ist aber schon lauffähig.

b) Legen Sie in *ui_widget.h* das folgende private Attribut an:

```
VerticalThread *vthread;
```

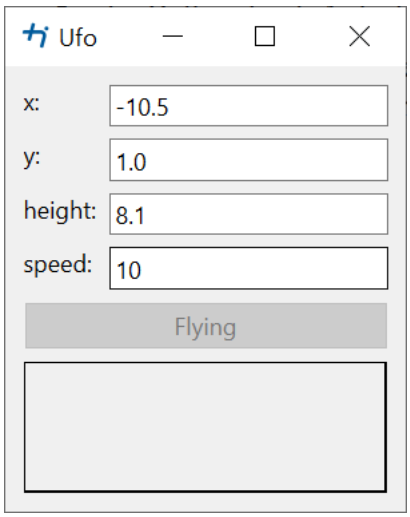
Im Konstruktor der Klasse `MainWidget` soll `vthread` mit einem neuen VerticalThread initialisiert werden.

c) Fügen Sie nun in *ui_widget.h* einen Slot `startUfo` und einen Aufruf der Methode `connect` ein, die das Signal `clicked` des Buttons mit dem Slot verbindet. Verwechseln Sie den Slot `startUfo` nicht mit der gleichnamigen Methode der Klasse `VerticalThread`. Der Slot soll noch ohne Funktionalität sein. Das kommt in der nächsten Teilaufgabe.

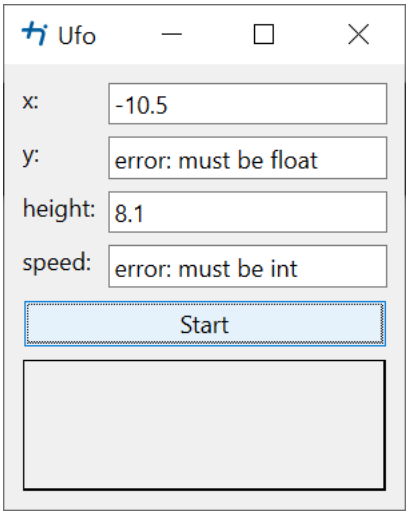
d) Nun wird der Slot `startUfo` der Klasse `MainWidget` ergänzt. Darin werden

- die Methode `startUfo` mit dem Objekt, auf das `vthread` zeigt, aufgerufen, dabei werden die Werte für *x*, *y*, *height* und *speed* aus den Eingabefeldern übergeben
- der Text unterhalb des Buttons mit `"\n\n\n"` überschrieben
- die Beschriftung des Buttons auf `"Flying"` geändert
- der Button durch Aufruf der `QPushButton`-Methode `setEnabled(false)` deaktiviert.

Das Ufo kann schon gestartet werden. Nach dem Landen bleibt das Fenster aber „eingefroren“ und das Programm muss mit dem Schließen des Fensters beendet werden.



e) Der Benutzer kann in die Eingabefelder beliebige Zeichenreihen eingeben. Wenn diese keine gültigen Zahlen darstellen, kommt es zu Fehlern. Ergänzen Sie deshalb den Slot `startUfo` um eine Überprüfung der Eingaben. Bei einem Fehler soll eine Fehlermeldung in dem Eingabefeld angezeigt werden, in dem der Fehler auftritt. In der folgenden Abbildung waren die Eingaben von *y* und von *speed* ungültig.



f) Wenn der Flug des Ufos beendet ist, soll das Fenster benachrichtigt werden. Dazu müssen wir die Klasse `VerticalThread` ergänzen. Zur Vorbereitung soll die Klasse in der Datei *vertical_thread.h* zusammengefasst werden. Die Implementierungen der Methoden sollen in der Klassendeklaration erfolgen. Die Datei *vertical_thread.cpp* wird nicht mehr benötigt.

g) Jetzt kommt die Ergänzung der Klasse `VerticalThread`.

- Die Klasse soll nun von `QObject` abgeleitet werden.
- Dazu muss `QObject` inkludiert werden.
- Damit Signale verwendet werden können, muss am Anfang der Klasse die folgende Zeile eingefügt werden.

```
Q_OBJECT
```

- Nach Beendigung des Fluges soll ein Signal an das Fenster gesendet werden. Fügen Sie in die Klasse die beiden folgenden Zeilen ein:

```
signals:
    void stopped(std::vector<float>);
```

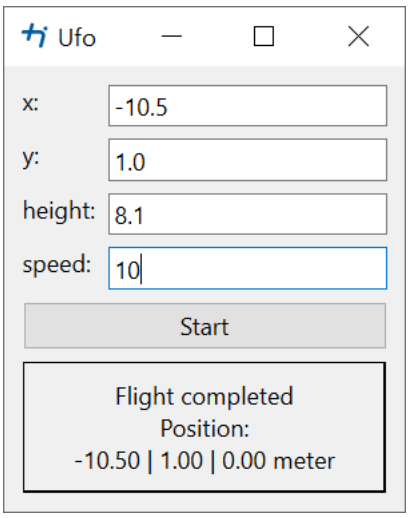
- Dieses Signal soll am Ende der Methode `runner` ausgelöst werden:

```
emit stopped(vert->getPosition());
```

  Dadurch wird die Position des Ufos nach der Landung an das Fenster gesendet.

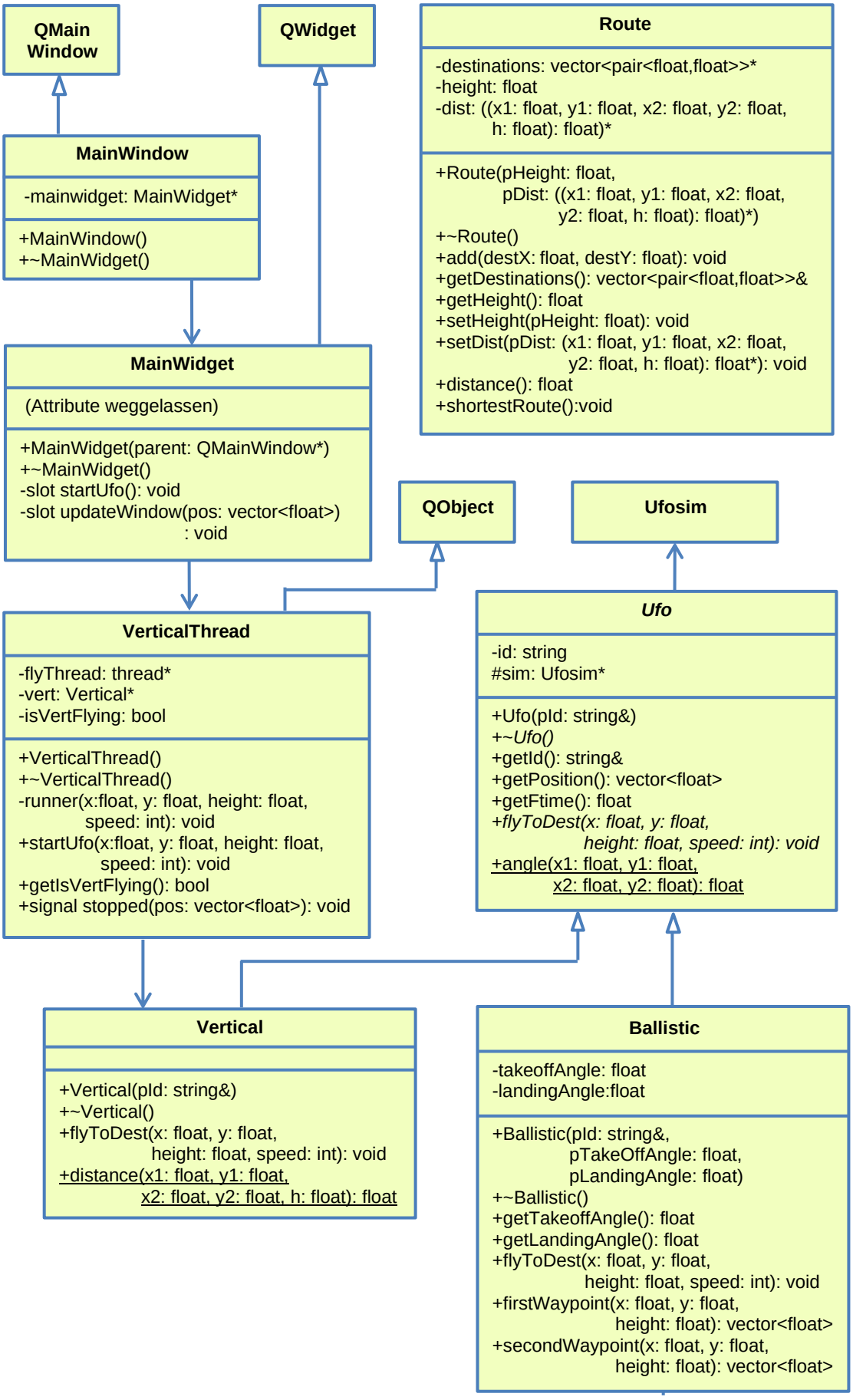
h) Wir brauchen noch einen Slot `updateWindow` in *ui_widget.h*, in dem

- ein Text, wie in der Abbildung unten gezeigt, auf dem Label `plabel` angezeigt wird
- die Beschriftung des Buttons auf `"Start"` geändert wird
- der Button durch Aufruf der `QPushButton`-Methode `setEnabled(true)` aktiviert wird.



i) Als Letztes wird durch Aufruf der Methode `connect` das Signal `stopped` von `vthread` mit dem Slot `updateWindow` verbunden. Sowohl im Signal als auch im Slot muss der Parametertyp `std::vector<float>` angegeben werden.

Insgesamt haben wir jetzt das folgende Klassendiagramm:



Hinweise:

- Die GUI soll nicht mit einem grafischen Designer (z.B. Qt Designer) sondern im Code erstellt werden.
- Die Anordnung der Elemente im `MainWidget` soll durch ein Layout und den Layout Manager festgelegt werden.
- Die C++-Version kann in der pro-Datei durch die Zeile

  ```
  CONFIG+=c++2a
  ```

  spezifiziert werden.

Die Abgabe besteht aus den Dateien *ballistic.h, ballistic.cpp, vertical.h, vertical.cpp, ufo.h, ufo.cpp, route.h, route.cpp, vertical_thread.h, ui_main.cpp, ui_window.h, ui_widget.h.*

Alle Parameter, alle Referenzrückgaben und alle Methoden sollten, soweit möglich, `const` sein.

Bitte überprüfen Sie vor der Abgabe, ob sich das Projekt fehlerfrei mit Qt erstellen lässt.