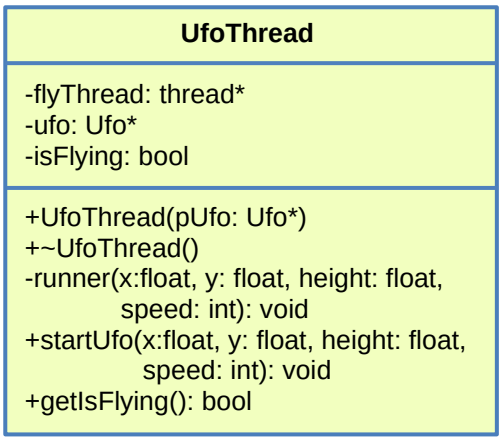
Praktikumsaufgabe 4 zur Vorlesung Grundlagen der Programmierung 2

Prof. Dr. Robert Gold

Technische Hochschule Ingolstadt
Sommersemester 2025

Der Flug eines Ufos kann je nach Ziel ganz schön lange dauern. Während des Fluges ist das Programm blockiert. Um Abhilfe zu schaffen, lassen wir das Ufo jetzt in einem Thread fliegen.

Die bisher erstellten Klassen bleiben unverändert. Neu hinzu kommt eine Klasse `UfoThread` in den Dateien *ufo_thread.h* und *ufo_thread.cpp*. Die Klasse hat ein Thread-Attribut und folgt damit dem Muster hat-ein.



- Das Attribut `flyThread` ist ein Pointer auf einen Thread. Er soll mit `nullptr` initialisiert werden. Durch die Abfrage `flythread != nullptr` kann im Programm abgefragt werden, ob ein Thread existiert.
- Das Attribut `ufo` ist ein Pointer auf ein Ufo, also auf ein Objekt der Klasse `Vertical` oder `Ballistic`.
- Das Attribut `isFlying` soll `true` sein, wenn das Ufo fliegt und `false` sein, wenn ein Flug beendet wurde. Der Initialwert ist `false`.
- `UfoThread` ist der Konstruktor und `~UfoThread` ist der Destruktor.
- Die Methode `runner` ist die Thread-Funktion. Sie fliegt das Ufo `ufo` nach (x, y, 0.0) auf Flughöhe `height` mit Geschwindigkeit `speed`. Die Parameter sollen const sein.
- Die Methode `startUfo` startet den Thread. Die Parameter von `startUfo` werden an die Thread-Funktion durchgereicht. Die Parameter sollen const sein.

- Die Methode `getIsFlying` ist ein Getter für das Attribut `isFlying`. Der Getter soll const sein.

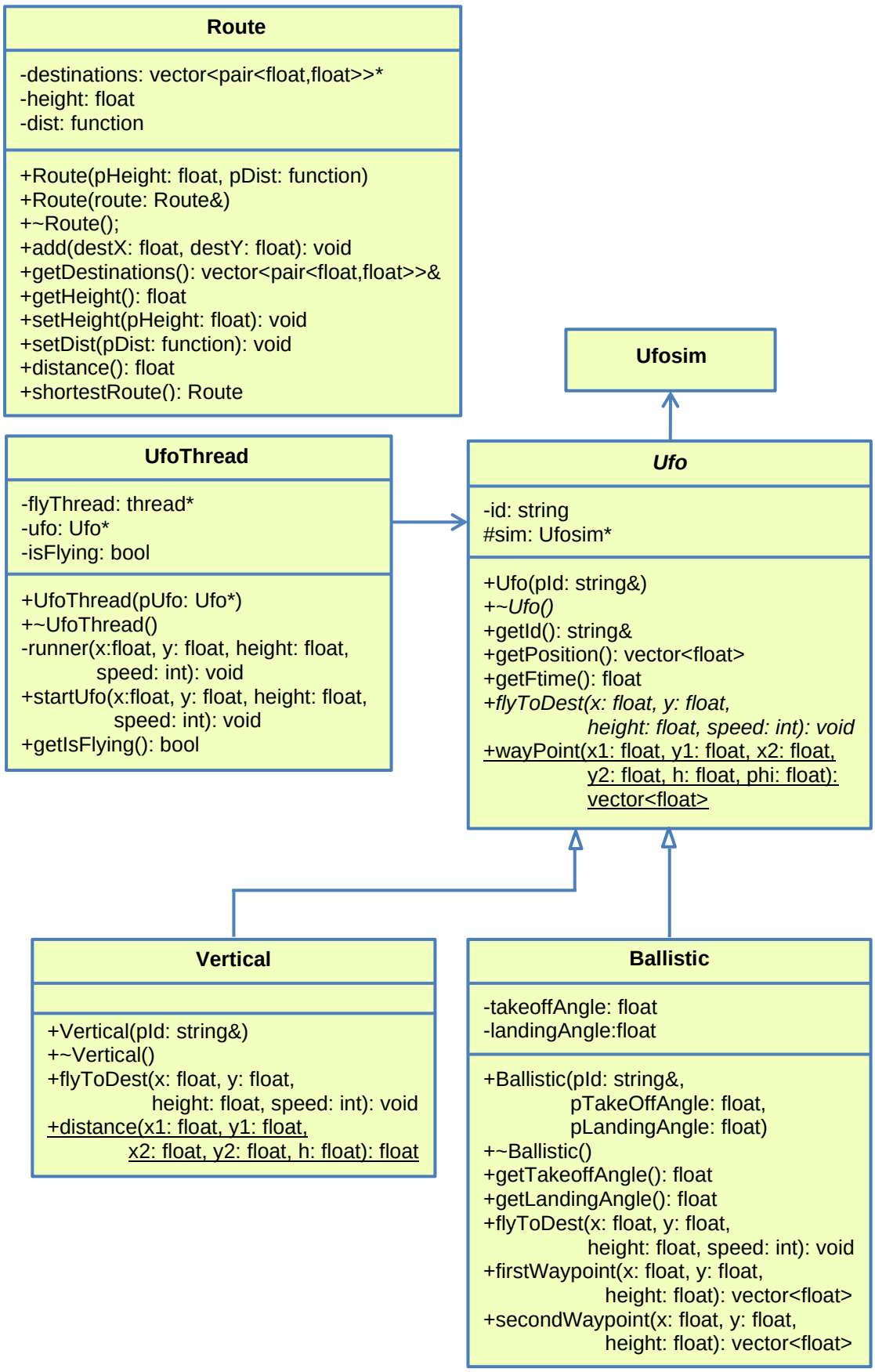
Achten sie darauf, dass der Thread mit `join()` geschlossen wird und zwar immer, bevor ein neuer Thread gestartet wird und wenn das `UfoThread`-Objekt aufhört zu existieren. Die Methode `join()` darf aber nur aufgerufen werden, falls ein Thread-Objekt existiert, sonst gibt es einen Nullpointer-Fehler.

Neben dem Unit-Test können Sie auch das folgende Hauptprogramm für Testläufe verwenden:

```
#include <iostream>
#include "ballistic.h"
#include "vertical.h"
#include "ufo_thread.h"

int main() {
    Vertical *ufo = new Vertical("r2d2");
    //Ballistic *ufo = new Ballistic("r3d3", 80, 80);
    UfoThread uthread(ufo);

    // fly from (0.0, 0.0, 0.0) to (5.0, -1.5, 0.0)
    // at altitude 4.0 with 10 km/h in a new thread
    uthread.startUfo(5.0, -1.5, 4.0, 10);

    // in the time until landing other work could be done
    while (uthread.getIsFlying())
    {
        std::cout << '.';
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }

    // fly from (5.0, -1.5, 0.0) to (-3.0, 0.0, 0.0)
    // at altitude 8.0 with 5 km/h in a new thread
    uthread.startUfo(-3.0, 0.0, 8.0, 5);

    return 0;
}
```

Die Methode `startUfo` kann nach dem Landen des Ufos auch nochmals aufgerufen werden, um das Ufo ein weiteres Mal fliegen zu lassen. Der zweite Flug startet dabei an der aktuellen Position, also ungefähr am Ziel des ersten Fluges. Dabei wird ein weiterer `uthread` gestartet. Damit nicht mehrere Threads gleichzeitig existieren, muss der letzte Thread vor dem Starten des nächsten Threads gejoint werden.

Insgesamt haben wir jetzt:



Im Unit-Test *pa4_utest.cpp* wird ein Flug gestartet und danach 15 Sekunden auf die Beendigung gewartet. Damit die Wartezeit ausreicht, sollte im Unit-Test der Speedup in

ufo.cpp auf 4 gesetzt sein und die Geschwindigkeit 10 km/h betragen. Sie müssen dazu nichts ändern, das ist bereits in *pa4_utest.cpp* so eingetragen. Ansonsten sind der Speedup und die Geschwindigkeit natürlich in den vorgegebenen Grenzen frei wählbar.

Die Abgabe besteht aus den Dateien *ballistic.h*, *ballistic.cpp*, *vertical.h*, *vertical.cpp*, *ufo.h*, *ufo.cpp*, *route.h*, *route.cpp*, *ufo_thread.h*, *ufo_thread.cpp*.

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

Bitte überprüfen Sie vor der Abgabe, ob sich das Projekt fehlerfrei erstellen lässt:

```
g++ -Wall -std=c++20 -I"C:\Program Files\boost_1_87_0" ballistic.cpp
vertical.cpp ufo.cpp route.cpp ufo_thread.cpp ufosim.cpp pa4_utest.cpp -o
pa4.exe
```

Alle Unit-Tests in *pa4_utest.cpp* müssen fehlerfrei laufen. Starten Sie dazu das Programm mit dem Befehl

```
pa4
```

Das Ergebnis muss

```
*** No errors detected
```

sein.