Turtle-Graphics

In diesem Praktikum werden Sie Turtle-Graphics[1] zeichnen. Dazu lesen Sie eine Datei ein, die Steuer-Kommandos für einen Zeichen-Roboter enthält. Sie implementieren die zu diesen Kommandos gehörenden Funktionen des Roboters. Dabei nutzen Sie die bereits bekannte Grafikbibliothek um Linien in ein Fenster zu zeichnen.

I. Hintergrund

Turtle-Graphics nutzt einen kleinen Roboter der mit einem Stift zeichnet, indem er sich vorwärts bewegt oder nach links bzw. rechts dreht. Entstanden ist es mit tatsächlichen Robotern, die programmiert werden konnten. In diesem Praktikum programmieren Sie einen „virtuellen" Roboter, der sich in einem Fenster auf Ihrem Bildschirm bewegt und dabei Linien zeichnet. Da es hier lediglich um die fertige Zeichnung geht, wird der Roboter selbst nicht gezeichnet.
Der Roboter startet in der Fenstermitte mit Blickrichtung nach rechts und hat den Stift angehoben. Mit entsprechenden Kommandos können Sie den Stift senken und heben. Sie können den Roboter eine angegebene Strecke in Blickrichtung geradeaus laufen lassen. Bei abgesenktem Stift wird auf diese Weise eine Linie gezeichnet, bei angehobenem Stift wird nicht gezeichnet. Weiterhin können Sie den Roboter um einen angegebenen Winkel nach links oder rechts drehen. Der Roboter verändert dabei seine Position nicht und zeichnet dementsprechend nicht. Ein Beispiel dazu finden Sie in Abbildung 1.

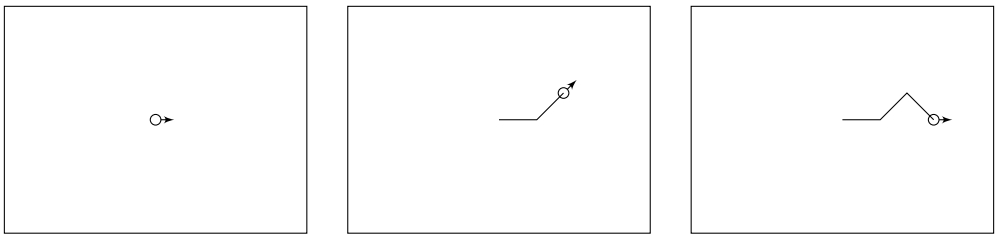

Abbildung 1: Der Roboter im Ausgangszustand (links). Die Zeichnung und der Roboter nach Absenken des Stifts, einem Schritt, einer Drehung um 45° nach links und einem weiteren Schritt (Mitte). Das Ergebnis nach einer weiteren Drehung nach rechts, einem weiteren Schritt und einer Drehung nach links (rechts).

II. Turtle Graphics

1. Erste Schritte

(a) Implementieren Sie eine Konsolenanwendung, die ein Grafik-Fenster öffnet.

(b) Aktivieren und deaktivieren Sie das Zeichnen wie in einer der vorherigen Praktikumsaufgaben. Zeichnen Sie mithilfe folgender Funktion mehrere Linien in das Fenster:

```
void gip_gfx_draw_line(int start_x, int start_y, int end_x, int end_y);
```

(c) Zeichnen Sie das Beispiel aus Abbildung 1 mit Turtle-Graphics. Implementieren Sie dazu folgende Funktionen:

```
void up(void);
void down(void);
void forward(float distance);
void turn(float angle);
```

Mit `up` und `down` heben und senken Sie den Stift. Mit `forward` bewegen Sie den Roboter um `distance` in Blickrichtung. Bei abgesenktem Stift zeichnen Sie eine Linie. Mit `turn` drehen Sie den Roboter um den Winkel `angle` in Grad. Positive Winkel drehen gegen den Uhrzeigersinn. Speichern Sie den Zustand des Roboters in globalen Variablen.

[1] `https://en.wikipedia.org/wiki/Turtle_graphics`

2. Steuerkommandos aus einer Datei

Erweitern Sie Ihre Konsolenanwendung, sodass sie Eingaben von Steuerkommandos aus einer Datei verarbeiten kann.

(a) Machen Sie sich mit der Eingabe aus Dateien vertraut[2]. Insbesondere werden Sie folgende Funktionen aus der C-Standardbibliothek verwenden:

```
FILE* fopen(const char *filename, const char *mode );
int fclose(FILE* stream );
char* fgets(char* str, int count, FILE* stream );
int sscanf(const char* buffer, const char* format, ... );
```

(b) Falls erforderlich ändern Sie den Kopf der `main`-Funktion in folgende Version:

```
int main(char argc, char* argv[])
```

Recherchieren Sie, wie Sie per Kommandozeile einen Dateinamen an Ihr Programm übergeben und diesen im Programm auslesen können. Ihr Programm soll folgendermaßen aufgerufen werden können:

```
./turtle FILENAME
```

`turtle` ist in diesem Beispiel der Name Ihres Programms, `FILENAME` steht für den Namen der Datei mit Steuerkommandos.

(c) Implementieren Sie das Einlesen von Steuerkommandos aus einer Datei. Die Zeilen der Datei sollen folgendes Format haben:

```
C VALUE
```

`C` steht dabei für ein einzelnes Zeichen, das ein Steuerkommando darstellt. `VALUE` steht dabei für einen optionalen Wert (z.B. Schrittweite, Winkel), den das entsprechende Steuerkommando benötigt. Unterstützen Sie folgende Steuerkommandos:

- `D`: Stift absenken (*down*)
- `U`: Stift anheben (*up*)
- `F`: vorwärts gehen mit angegebener Schrittlänge (*forward*)
- `T`: drehen um angegebenen Winkel (*turn*, positiver Winkel: gegen den Uhrzeigersinn)

Alle übrigen Zeilen werden ignoriert.

Legen Sie eine Steuerdatei an, um das Beispiel aus Abbildung 1 mit Turtle-Graphics zu zeichnen.

(d) Legen Sie eine passende Steuerdatei an, um das Haus vom Nikolaus mit Turtle-Graphics zu zeichnen.

3. Wiederholungen

(a) Legen Sie eine passende Steuerdatei an, um ein Quadrat mit Turtle-Graphics zu zeichnen, das genau mittig im Fenster liegt.

(b) Machen Sie sich mit folgenden Funktionen der C-Standardbibliothek vertraut:

```
int fseek( FILE *stream, long offset, int origin);
long ftell( FILE *stream );
```

[2] `https://en.cppreference.com/w/c/io`

(c) Erleichtern Sie das Zeichnen des Quadrats, indem Sie folgendes Steuerkommando implementieren:

- `R`: wiederhole (*repeat*) sämtliche bis zum Dateiende folgenden Kommandos *n* Mal. Dieses Steuerkommando erfordert die Angabe des Werts *n*.

(d) Zeichnen Sie Fünf- und Sechsecke.

(e) Zeichnen Sie einen Kreis.

4. Zustand speichern und wiederherstellen

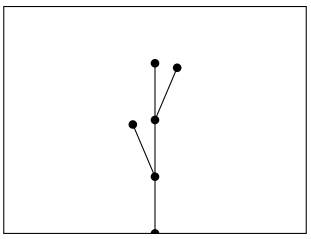

Abbildung 2: Ein einfacher Zweig.

(a) Zeichnen Sie einen einfachen Zweig ähnlich dem in Abbildung 2. Beginnen Sie mittig am unteren Fensterrand. Die einzelnen Positionen des Roboters sind durch Punkte markiert. Alle Schritte sind gleich groß. Legen Sie Schrittweite und Drehwinkel selbst fest.

(b) Erleichtern Sie das Zeichnen des Zweiges, indem Sie folgende Steuerkommandos implementieren:

- `{`: aktuellen Zustand des Roboters speichern
- `}`: zuvor gespeicherten Zustand wiederherstellen

III. Herausforderung

5. Wiederholungen von Blöcken

Erweitern Sie die Möglichkeit der Wiederholung so, dass diese

(a) nicht bis zum Dateiende reichen,

(b) verschachtelt werden können.

Möglicherweise benötigen Sie dazu ein weiteres Kommando

- `E`: beende die Wiederholung (*end repeat*).

Möglicherweise lassen sich aber auch die geschweiften Klammern wiederverwenden, was eleganter wäre. Passen Sie dann aber auf den Roboterzustand auf.