

Sternenkarte (15 Punkte)

In diesem Projekt schreiben wir ein kleines C Programm, das eine Sternenkarte einliest und sie als Bild ausgibt. Lernziele sind hierbei:

- Einlesen und Schreiben von Dateien
- Anlegen von einfachen Datenstrukturen in C
- Zusammengesetzte Datentypen, ein Hauch von Objekten

1 Programmierungsumgebung

Dieses Projekt schreiben wir in C. Sie benötigen daher eine C-Programmierungsumgebung und einige Werkzeuge. Sie erhalten das Programmierprojekt unter

`https://prog2scm.cdl.uni-saarland.de/git/project2/\$NAME`

`$NAME` ist durch Ihren Benutzernamen zu ersetzen.

2 Übersicht

Ihr Programm `stars` soll eine Liste von Sternen einlesen und diese dann in ein Bild zeichnen. Die Sterne sind im Verzeichnis `data` in einer Datei mit dem Namen `stars.txt` gegeben. Im Verzeichnis `data` sind weitere Dateien, die Sternbilder beschreiben. Diese werden gegen Ende des Projekts benötigt. Ihr Projekt hat folgenden Aufbau:

- Die Datei `star.c` enthält alle Unterprogramme, die Operationen auf Sternen ausführen. Die Datei `star.h` enthält die dazugehörigen Deklarationen der Unterprogramme und Datentypen.
- Die Datei `image.c` enthält alle Unterprogramme, die auf Bildern arbeiten. Ein Unterprogramm `image_draw_line` ist bereits vorimplementiert. Die Datei `image.h` enthält die dazugehörigen Deklarationen der Unterprogramme und Datentypen.
- Die Datei `main.c` enthält das Hauptprogramm. Es ist dafür verantwortlich, die Liste der Sterne einzulesen, die Sterne und die Sternbilder zu zeichnen, und das Ergebnisbild zu schreiben. Hierzu bedient es sich der Unterprogramme der anderen Übersetzungseinheiten.

Um das Programm zu bauen (übersetzen, assemblieren und binden), benutzt man das `Makefile`, das dem Projekt beiliegt: Durch `make` auf der Kommandozeile wird das Programm gebaut. Zusätzlich wird das Testprogramm `test` erzeugt (siehe Abschnitt 5). Das Programm `stars` wird wie folgt aufgerufen:

```
./stars <breite des bildes in pixel> <datei mit den sternern> [<sternbilddatei>...]
```

`stars` ist der Name des Programms, danach die Größe des zu erzeugenden Bildes in Pixeln (das Bild ist quadratisch, daher ist hier nur eine Zahl anzugeben), danach die Liste der Sterne und, optional, eine Liste von Dateien, die Sternbilder beschreiben (siehe unten). Das Programm erzeugt als Ausgabe eine Datei `stars.pbm` (siehe Abschnitt 4).

2.1 Die Sterne

Die Datei `stars.txt` enthält eine Liste von Sternen. Jede Zeile beschreibt einen Stern und hat sechs Felder:

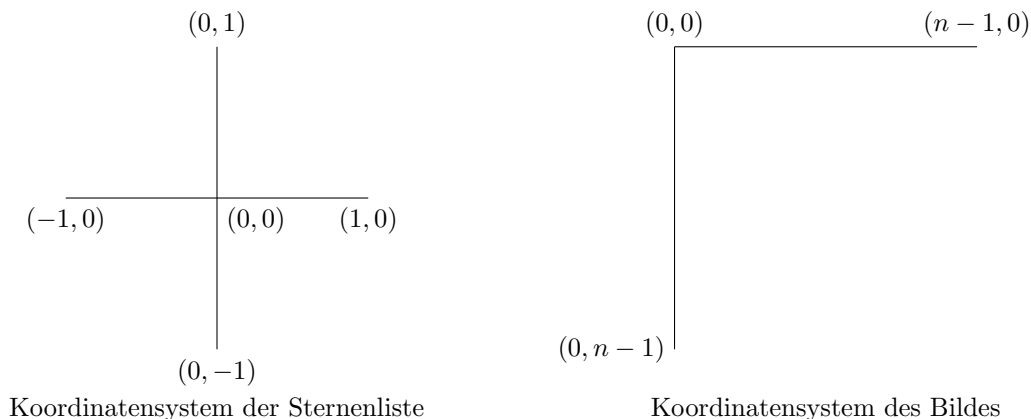
1. x -Koordinate des Sterns.
2. y -Koordinate des Sterns.
3. z -Koordinate des Sterns. Wir werden die z -Koordinate im folgenden nicht mehr brauchen, sie kann also ignoriert werden.
4. Die Henry-Draper-Nummer¹ des Sterns. Sie trägt keinerlei Bedeutung und ist lediglich eine eindeutige Identifikationsnummer des Sterns.
5. Die Helligkeit (magnitude) des Sterns.
6. Die Harvard revised number. Eine weitere Identifikationsnummer des Sterns, die wir im folgenden nicht benötigen werden.

Zum Beispiel:

```
0.994772 0.023164 -0.099456 28 4.61 3
```

2.1.1 Das Koordinatensystem

Die x - und y -Koordinaten der Sterne sind als Gleitkommazahlen gegeben und befinden sich im Intervall $[-1, 1]$. Da das Ausgabebild n Pixel hat und wir keine Kommazahlen haben, müssen wir die Koordinaten in die Bildkoordinaten transformieren. Im Bildkoordinatensystem eines Bildes mit n Pixeln hat die linke obere Ecke die Koordinate $(0, 0)$ und die rechte untere Ecke die Koordinate $(n - 1, n - 1)$:



2.2 Das Bild

Das struct `image` (siehe `image.h`) repräsentiert ein 2D-Bild als ein Paar aus Breite und Höhe (`w` = width, `h` = height) und einem Zeiger auf eine Reihung von `ints`. In dieser Reihung liegen die Zeilen des Bildes hintereinander. Sie hat also Platz für $w \times h$ Pixel. Die Farbe jedes Pixels ist als Rot-, Grün-, und Blauwert zwischen 0 und 255 kodiert. Die drei Bytes, die zur Speicherung der Farbe nötig sind, werden in dem `int` gemeinsam gespeichert: Rot von Bit 16–23, Grün von Bit 8–15, Blau von Bit 0–7.

Ein Beispiel: Möchte man in einem Bild mit den Ausmaßen 100×200 den Pixel mit den Koordinaten $(10, 50)$ mit dem Farbwert $(64, 32, 16)$ bemalen, so muss man in der Reihung den Eintrag $100 \times 50 + 10$ auf den Wert `0x00402010` setzen.

¹siehe auch: http://en.wikipedia.org/wiki/Henry_Draper_Catalogue

3 Aufgaben

3.1 Einlesen und Zeichnen der Sterne, Schreiben des Bildes (10 Punkte)

1. Implementieren Sie die fehlenden Funktionen in `star.c`

- `star_coord_to_pixel` transformiert die Koordinaten des Sterns in das Bildkoordinatensystem (siehe oben). Beginnen Sie hier; diese Funktion ist leicht. Führen Sie alle Rechnungen in `double` aus und wandeln Sie die berechnete Koordinate mit einer expliziten Typumwandlung in einen `int` um. Dies schneidet den Nachkommateil einfach ab. Verwenden Sie also *keine* kaufmännische Rundung!
- `read_star_from_file` liest einen Stern aus einer Zeile der übergebenen Datei indem die Felder des übergebenen structs gesetzt werden. Machen Sie sich mit der C-Funktion `fscanf` vertraut; sie kann verwendet werden, um den Text einer Zeile der Sternendatei in Daten zu konvertieren. Die Dokumentation einer C-Funktion *x* können Sie auf der Kommandozeile mit `man 3 x` nachschlagen.

Die Funktion soll eine 1 zurück liefern, wenn der Stern ordnungsgemäß gelesen wurde. Andernfalls soll 0 zurückgegeben werden. Dies können Sie dann im Hauptprogramm verwenden, um das Lesen korrekt zu beenden.

- `star_plot` zeichnet einen Stern auf das Bild. Zeichnen Sie hierzu an die entsprechende Bildkoordinate einen weißen Pixel.

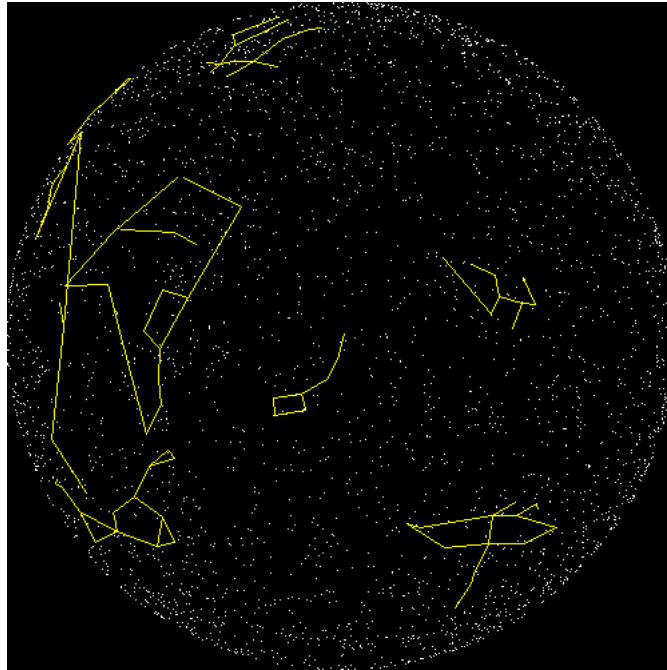
2. Implementieren Sie die fehlenden Funktionen in `image.c`:

- `image_init` initialisiert das übergebene struct `image`. Hierzu werden die als Parameter übergebenen Dimensionen in das struct eingetragen sowie ein entsprechend großer Bildpuffer angelegt und ebenfalls im struct vermerkt. Das Bild soll initial vollständig schwarz sein.
- `image_destroy` gibt den Speicher der mit `image_init` angefordert wurde, wieder frei.
- `image_draw_pixel` malt einen Pixel mit der angegebenen Farbe auf das Bild. Beachten Sie, dass Sie in dem Fall, dass die angegebenen Koordinaten nicht auf das Bild passen, nichts zeichnen dürfen.
- `image_write_to_file` schreibt das Bild in eine Datei. Das Dateiformat ist in Abschnitt 4 erläutert.

3.2 Aufgabe 2 (5 Punkte)

Implementieren Sie das Zeichnen von Sternbildern. Im Verzeichnis `data/` liegen Dateien, die auf `_lines.txt` enden. Sie enthalten pro Zeile ein durch Komma getrenntes Paar von Draper-Nummern. Durch ziehen einer Linie im Bild zwischen den Sternen mit den entsprechenden Draper-Nummern entsteht das Sternbild. Implementieren Sie eine Funktion, die eine Sternbild-Datei nimmt und das Sternbild in das Bild zeichnet. Verwenden Sie die Farbe reines Gelb dafür.

Das Endergebnis sollte so aussehen:



4 Ausgabedateiformat

Das Bild wird in der Variante P3 des *portable bitmap format* ausgegeben.² Für ein Bild der Breite w und der Höhe h ist die Bilddatei wie folgt aufgebaut:

- In Zeile 1 steht P3
- In Zeile 2 stehen die Breite und Höhe des Bildes
- In Zeile 3 steht die Zahl 255.
- Dann folgen h Zeilen mit jeweils w Pixeln. Jeder Pixel besteht aus drei Zahlen, die jeweils den Rot-, Grün-, und Blauwert der Farbe des Pixels darstellen. In jeder Komponente ist 0 der kleinste, 255 der größte Wert.

Zum Beispiel besteht das Bild

```
P3
2 2
255
255 255 255 0 0 0
0 0 0 255 255 255
```

aus einem weißen und schwarzen Pixel in der ersten Zeile und einem schwarzen und weißen Pixel in der zweiten Zeile.

5 Tests

Schauen Sie sich die mitgelieferte Test-Suite in `test.c` an. Dort sind einige Tests, die die Funktionen Ihres Projekts einzeln überprüfen. Sie können alle Tests mit dem Befehl `./tests` laufen lassen. Außerdem können Sie einen einzelnen Test mit Nummer `num` mit dem Befehl `./tests num` laufen lassen. Diese Funktionalität ist insbesondere interessant, solange noch einige Teile Ihrer Implementieren **aborten** oder **segfaulten** sollten.

²Siehe auch: http://en.wikipedia.org/wiki/Netpbm_format

Die mitgelieferten Tests bilden die *public* Tests. Um überhaupt Punkte in diesem Projekt zu erzielen, müssen Sie alle public Tests bestehen. Diese werden außerdem in regelmäßigen Abständen zusammen mit geheimen *daily* Tests auf unserem Testserver ausgeführt. Sie erhalten über den Ausgang der Tests eine Benachrichtigung per Email. Nach Ende des Projekts wird Ihre Abgabe mithilfe weiterer *secret* Tests ausgewertet. Wir ziehen dazu den Zustand Ihres git-Depots *vom 24.5.2016 um 23:59* heran.

6 Hinweise zur Bearbeitung des Projektes

Es folgen noch einige Hinweise, die Sie bei der Bearbeitung des Projekts beachten sollen:

- Die Kommentare in Ihrer Projektvorlage sind Teil der Spezifikation.
- Bedenken Sie, dass wir Ihr Projekt auch gegen andere Sternenkarten testen.
- Auch wenn **stars** nur quadratische Bilder erzeugen kann, muss Ihre **image** Implementierung auch mit rechteckigen Bildern zurecht kommen.
- Die Helligkeit der Sterne wird zwar im Projekt selbst nicht benötigt, wir testen aber, ob Sie das Datum korrekt einlesen und setzen.