Sternenkarte (12 Punkte)

In diesem Projekt schreiben wir ein C Programm, das eine Sternenkarte einliest und sie als Bild ausgibt.

1 Vorbereitung

Falls Sie nicht am ersten Projekt oder dem Vorkurs teilgenommen haben, müssen Sie Ihr Git noch konfigurieren. Führen Sie dazu die folgenden Befehle aus:

- `git config --global user.name "Vorname Nachname"`
- `git config --global user.email "kennung@stud.uni-saarland.de"`

Dabei ersetzen Sie Vorname und Nachname durch Ihren Vor- bzw. Nachnamen und kennung durch Ihre studentische Kennung.

Klonen Sie das Projekt-Depot in einen Ordner mit dem Namen `project2`:

- `git clone https://prog2scm.cdl.uni-saarland.de/git/project2/$username project2`

Ersetzen Sie dabei `$username` durch Ihren Benutzernamen. Wir möchten wir Sie drauf hinweisen, dass unser Server nur aus dem Universitätsnetz erreichbar ist.

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 sternen> [<datei mit sternbild>...]
```

`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:

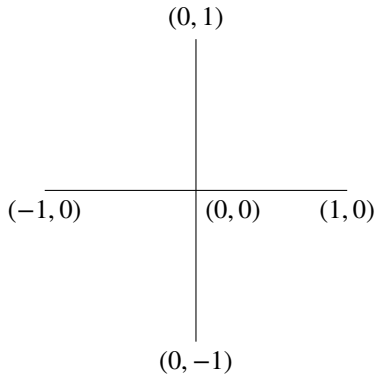
1. x-Koordinate des Sterns.
2. y-Koordinate des Sterns.
3. z-Koordinate des Sterns. Wir werden die z-Koordinate im Folgenden nicht mehr brauchen, sie kann also ignoriert werden.
4. Die Henry-Draper-Nummer[1] des Sterns. Dies ist eine eindeutige Identifikationsnummer des Sterns.
5. Die Helligkeit (magnitude) des Sterns.
6. Die Harvard revised number. Eine weitere Identifikationsnummer des Sterns, die wir im Folgenden nicht benötigen werden.

Zum Beispiel:

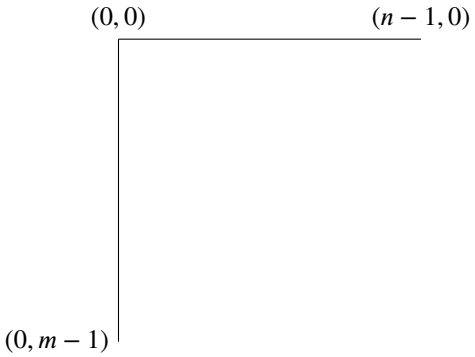
```
0.994772 0.023164 -0.099456 28 4.61 3
```

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 $n \times m$ Pixel hat und wir keine Kommazahlen haben, müssen wir die Koordinaten in die Bildkoordinaten transformieren. Im Bildkoordinatensystem eines Bildes mit $n \times m$ Pixeln hat die linke obere Ecke die Koordinate $(0, 0)$ und die rechte untere Ecke die Koordinate $(n - 1, m - 1)$:



Koordinatensystem der Sternenliste



Koordinatensystem des Bildes

Die Koordinatentransformation einer Position (x, y) aus dem Koordinatensystem der Sterne in das Bildkoordinatensystem wird durch die folgenden Funktionen beschrieben:

$$b(x) = \frac{1 + x}{2} * (n - 1)$$
$$b(y) = \frac{1 - y}{2} * (m - 1)$$

2.2 Das Bild

Das struct `image` (siehe `image.h`) repräsentiert ein rechteckiges 2D-Bild als ein Paar aus Breite und Höhe (`w` = width, `h` = height) und einem Zeiger auf eine Reihung von `int`s. In dieser Reihung liegen die Zeilen des Bildes hintereinander. Sie hat also Platz für `w`×`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.

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

3 Aufgaben

3.1 Schreiben des Bildes (2 Punkte)

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 Einlesen und Zeichnen der Sterne (4 Punkte)

Implementieren Sie die fehlenden Funktionen in `star.c`

- `star_coord_to_pixel` transformiert die Koordinaten des Sterns in das Bildkoordinatensystem (siehe Abschnitt 2.1.1). 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!
- `star_read_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` *x* nachschlagen.

  Die Funktion soll eine 1 zurück liefern, wenn der Stern ordnungsgemäß gelesen wurde. Andernfalls soll 0 zurückgeben 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.

3.3 Die Main Datei (2 Punkte)

Vervollständigen Sie die Datei `main.c` in dem Sie die in Aufgabenteil 3.1 und 3.2 implementierten Funktionen passend aufrufen. Mehr Informationen dazu finden Sie in der Datei `main.c`.

3.4 Zeichnen von Sternenbildern (4 Punkte)

Implementieren Sie das Zeichnen von Sternbildern in der Funktion `draw_constellation_from_file`. 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. Falls die angegebenen Draper-Nummern nicht existieren, dürfen Sie nichts zeichnen. Sie sollen in Ihrer Implemetierung die schon für Sie geschriebene Funktion `image_draw_line` nutzen. Hinweise zu ihrer Nutzung finden sie in Abschnitt 6.
Ein Beispiel für das Endergebnis sehen Sie in Abbildung 1.

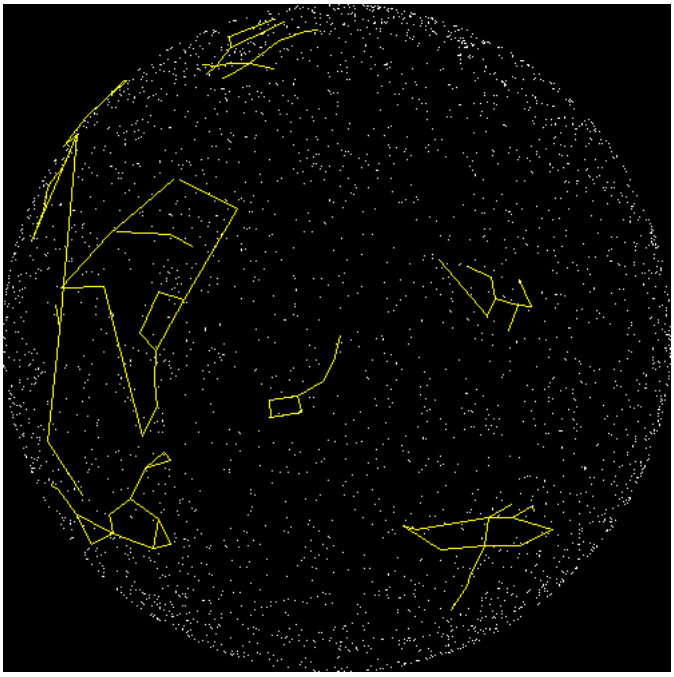


Abbildung 1: Ein Beispiel für die Ausgabe eines funktionierenden Programms

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 getrennt durch ein Leerzeichen
- 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 `./test` laufen lassen. Außerdem können Sie einen einzelnen Test mit Nummer `num` mit dem Befehl `./test num` laufen lassen. Diese Funktionalität ist insbesondere interessant, solange noch einige Teile Ihrer Implementieren `abort`en oder `segfault`en sollten.

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

Die mitgelieferten Tests bilden die *public* Tests. Um Punkte für eine (Teil-)Aufgabe zu erhalten, müssen Sie alle public Tests für diese bestehen. Die public Tests 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 *eval* Tests ausgewertet. Wir ziehen dazu den Zustand Ihres git-Depots *vom 21.5.2019 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 (magnitude) der Sterne wird zwar im Projekt selbst nicht benötigt, wir testen aber, ob Sie diesen Wert korrekt einlesen und setzen.
- Die Routine `image_draw_line` ist nicht kommutativ. Das heißt, die Aufrufe
  - `image_draw_line(img, color, x0, y0, x1, y1)` und
  - `image_draw_line(img, color, x1, y1, x0, y0)`

  erzeugen eine leicht andere Ausgabe. Sie müssen für Aufgabe 2 beim Zeichnen der Sternbilder darauf achten, den Start- und Endpunkt richtig zu wählen. Gehen Sie hierfür wie folgt vor:
  - Alle Sterndaten S sollten in einer Reihung aus `structs` vorliegen.
  - Lesen Sie in `draw_constellation_from_file` ein Paar Draper-Nummern d_1, d_2 ein.
  - Durchsuchen Sie nun S vorwärts nach den Draper-Nummern d_1 und d_2.
  - Der erste Treffer ist der Starpunkt, der zweite der Endpunkt.