Warming Stripes

I. Einleitung

In diesem Praktikum werden Sie die *Warming Stripes*[1] (Abbildung 1) von Ed Hawkins[2] zeichnen. Diese stellen für den Zeitraum seit 1850 eine farbliche Visualisierung der Änderung der globalen Jahresmitteltemperatur dar. Damit veranschaulichen sie den weltweiten Temperaturanstieg, der durch durch den menschgemachten Klimawandel verursacht wird.

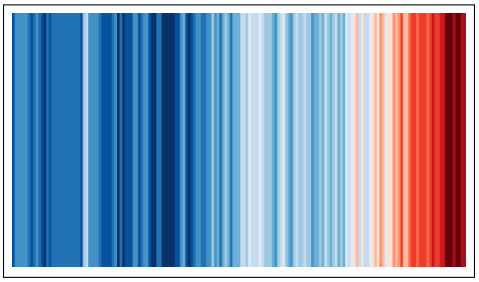

Abbildung 1: Warming Stripes; mittlere globale Temperaturanomalie 1850–2022[3].

Sie werden die *Warming Stripes* in ein eigenes Grafik-Fenster zeichnen. Damit dies möglich ist, binden Sie in Ihr Projekt mehrere Bibliotheken ein: die *SDL2*[4], eine umfangreiche Grafikbibliothek, sowie *gip_gfx* eine von uns entwickelte, leichtgewichtige Wrapper-Bibliothek, die Ihnen die Verwendung der *SDL2* erleichtern soll.
In diesem Praktikum installieren Sie zunächst die benötigte Software. Sie bauen Ihr Projekt auf den Ergebnissen des vorherigen Praktikums auf und verarbeiten dieselbe Eingabedatei wie zuvor. Anstelle einer Text-Ausgabe werden Sie die eingelesenen Daten farbcodiert darstellen.

II. Vorbereitung

Installieren Sie die benötigte Software wie in den folgenden Abschnitten erläutert.

II.1. CMake

CMake[5] ist ein sogenanntes *Build System*. Dies kann Ihnen das Kompilieren – im Fall von *CMake* auch das Testen, Installieren und Verpacken – größerer Softwaresysteme erleichtern. In diesem Praktikum benötigen Sie es, um die erforderlichen Bibliotheken zu bauen und zu installieren. Es ist mindestens die Version 3.23 erforderlich.

II.1.1 Linux

Möglicherweise liefert der Paketmanager Ihrer Distribution *CMake* in einer ausreichend neuen Version mit. Installieren Sie *CMake* über Ihren Paketmanager und fragen Sie die Version mit folgendem Kommandozeilenbefehl ab:

```
$ cmake --version
```

Sollte es sich nicht mindestens um Version 3.23 handeln, deinstallieren Sie das Paket wieder. Laden Sie nun eine passende *CMake*-Binär-Distribution als `.tar.gz`-Datei von der Download-Seite[6] herunter. Entpacken Sie diese Datei in einen geeigneten Ordner auf Ihrer Festplatte. Hierfür eignen sich z.B. `/opt/cmake/` oder `~/opt/cmake/`. Fügen Sie anschließend den absoluten Pfad des entpackten Unterordners `bin` der Umgebungsvariable `PATH` hinzu, damit Sie *CMake* aus der Kommandozeile aufrufen können.

[1] `https://showyourstripes.info`
[2] `https://research.reading.ac.uk/meteorology/people/ed-hawkins/`
[3] Quelle: #ShowYourStripes `https://showyourstripes.info`, Ed Hawkins. CC-BY.
[4] `http://www.libsdl.org`
[5] `https://cmake.org`
[6] `https://cmake.org/download/`

II.1.2 macOS

Unter macOS können Sie *CMake* wie unter Linux manuell installieren. Die Schritte sind dieselben wie unter Linux.
Sollten Sie den Paketmanager *Homebrew*[7] verwenden, können Sie *CMake* auch mit folgendem Befehl installieren:

```
% brew install --cask cmake
```

II.1.3 Windows

Unter Windows steht Ihnen ein Installer zur Verfügung. Laden Sie diesen (`.msi`-Datei) von der Download-Seite[8] herunter und führen ihn aus. Achten Sie darauf, *CMake* zur Umgebungsvariable `PATH` hinzuzufügen.

II.2. SDL

SDL[9] – *Simple DirectMedia Layer* – ist eine umfangreiche Grafikbibliothek. Im Praktikum verwenden Sie diese Bibliothek – wenn auch nicht direkt –, um z. B. ein Grafik-Fenster anzuzeigen und in dieses zu zeichnen.
Am besten bauen und installieren Sie *SDL* unter Linux, macOS und Windows selbst aus dem Quelltext. Dabei erleichtert Ihnen *CMake* die Arbeit, sodass Sie lediglich wenige Befehle in der Konsole eingeben müssen. Die Schritte sind unter Linux, macOS und Windows überwiegend identisch.
Laden Sie sich zunächst den Quelltext von *SDL* von der Download-Seite[10] herunter. Entpacken Sie anschließend die heruntergeladene Datei in einen sinnvollen Ordner auf Ihrer Festplatte.
Öffnen Sie ein Terminal und wechseln Sie darin in den eben entpackten Ordner – verwenden Sie unter Windows am besten die *Git Bash* – siehe Installation von Git im ersten Praktikum.
Legen Sie als erstes einen Unterordner an, in dem Sie *SDL* erstellen wollen und wechseln Sie in diesen Ordner:

```
$ mkdir UNTERORDNER
$ cd UNTERORDNER
```

Dabei steht `UNTERORDNER` für den gewünschten Namen des Unterordners.
Legen Sie nun einen neuen Ordner auf Ihrer Festplatte an, in den Sie *SDL* installieren werden. Verwenden Sie dazu folgenden Befehl:

```
$ mkdir -p INSTALLATIONSORDNER
```

Dabei steht `INSTALLATIONSORDNER` für den Pfad der gewünschten Ordners – z.B. `/opt/sdl2/` oder `~/opt/sdl2` oder unter Windows `/c/Programme/sdl2/` (bzw. `C:\Programme\sdl2`).
Lassen Sie *CMake* nun Ihre Projektmappe konfigurieren (Zeile 1 – Windows siehe unten), die Projektmappe bauen (Zeile 2) und *SDL* installieren (Zeile 3):

```
$ cmake -DCMAKE_INSTALL_PREFIX=INSTALLATIONSORDNER ..
$ cmake --build .
$ cmake --install .
```

Verwenden Sie unter Windows mit *MinGW* folgenden Befehl anstelle von Zeile 1:

```
$ cmake -G "MinGW Makefiles" -DCMAKE_INSTALL_PREFIX=INSTALLATIONSORDNER ..
```

Fügen Sie anschließend die absoluten Pfade sowohl des gewählten Installationsordners selbst als auch des darin enthaltenen Unterordners `bin` der Umgebungsvariable `PATH` hinzu.

[7] `https://brew.sh`
[8] `https://cmake.org/download/`
[9] `https://www.libsdl.org`
[10] `https://github.com/libsdl-org/SDL/releases/latest`

II.3. gip_gfx

gip_gfx ist eine schlanke Wrapper-Bibliothek um *SDL* und soll Ihnen die Arbeit mit *SDL* erleichtern. Diese Bibliothek wird von uns entwickelt und Ihnen zur Verfügung gestellt.
Auch *gip_gfx* bauen und installieren Sie mit *CMake*. Die Schritte sind unter Linux, macOS und Windows überwiegend identisch.
Laden Sie sich zunächst den Quelltext von *gip_gfx* aus dem Moodle-Kurs oder von der Praktikums-Seite[11] herunter. Entpacken Sie anschließend die heruntergeladene Datei in einen sinnvollen Ordner auf Ihrer Festplatte. Verfahren Sie wie beim Bauen und Installieren von *SDL*. Als Installationsordner verwenden Sie z.B. `/opt/gip_gfx/` oder `~/opt/gip_gfx` oder unter Windows `/c/Programme/gip_gfx/` (bzw. `C:\Programme\gip_gfx`).
Fügen Sie nach erfolgreicher Installation den absoluten Pfad des gewählten Installationsordners der Umgebungsvariable `PATH` hinzu.

III. Aufgaben

In diesem Teil implementieren Sie eine Anwendung, die Sie zwar aus der Konsole starten, die dann allerdings ein Grafik-Fenster anzeigt. Die Anwendung liest die Eingabedaten, die Sie im vorherigen Praktikum bereits verarbeitet haben und stellt anschließend im Fenster eine farbliche Visualisierung der erhaltenen Daten dar. Dazu legen Sie für jeden Zahlenwert, den Ihre Anwendung liest, anhand einer Farbtabelle eine Farbe fest. Damm zeichnen Sie ein mit dieser Farbe gefülltes Rechteck passender Breite ins Fenster.
Sehen Sie sich zur Vorbereitung Ihre Lösung des vorherigen Praktikums an. Rufen Sie sich besonders in Erinnerung, welche Daten Sie geladen haben, wie sie dies im Programm implementiert haben und wie Sie im Programm auf die Daten zugreifen können.
Beginnen Sie in diesem Praktikum ein neues, leeres Projekt. Sie werden zunächst implementieren, ein Fenster anzuzeigen, zu zeichnen und das Fenster zu schließen. Danach können Sie das Einlesen der Daten aus dem vorherigen Praktikum kopieren und in Ihr aktuelles Programm passend einfügen und anpassen. Sie werden zunächst den *Gcc* aus der Kommandozeile bedienen.

1. **Projekt anlegen**

   Legen Sie ein neues Projekt an. Die enthaltene Anwendung soll zunächst lediglich eine Ausgabe in der Konsole erzeugen. Bauen und testen Sie Ihre Anwendung. Beheben Sie ggf. auftretende Fehler.

2. **Hello gip_gfx**

   Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

   (a) Includieren Sie in Ihre Quelltextdatei den Header `gip_gfx/gip_gfx.h`.

   (b) Rufen Sie folgende Funktion aus *gip_gfx* auf, um ein Fenster anzuzeigen, das 348 Pixel breit und 215 Pixel hoch ist.

   ```
   int gip_gfx_create_window(const char* title, int width, int height)
   ```

   (c) Schicken Sie Ihre Anwendung in eine Endlosschleife, die automatisch auf das Programmende wartet, indem Sie unmittelbar vor der `return`-Anweisung folgende Funktion aufrufen:

   ```
   void gip_gfx_render_loop(void)
   ```

   (d) Bauen Sie Ihre Anwendung mit dem *Gcc* aus der Kommandozeile wie gewohnt. Lesen Sie sorgfältig die erhaltenen Fehlermeldungen.

   (e) Recherchieren Sie die Bedeutung der folgenden Flags, mit denen Sie den *Gcc* aufrufen können:
   - `-I` – (Großbuchstabe i)
   - `-L` – (Großbuchstabe L)
   - `-l` – (Kleinbuchstabe L)

[11] `https://vc.w-hs.de/teaching/gip1-2023/030/p/`

Beheben Sie die aufgetretenen Fehler, indem Sie den Aufruf von `gcc` entsprechend anpassen.
Verwenden Sie in der `gcc`-Befehlszeile folgende Reihenfolge:

- `gcc`
- `-o`-Flag mit Namen der Ausgabedatei
- `-I`- und `-L`-Flags mit entsprechenden Angaben
- Name(n) der Eingabedatei(en)
- `-l`-Flags mit entsprechenden Angaben

(f) Starten Sie Ihre Anwendung. Es sollte sich ein Fenster mit dem angegebenen Titel öffnen. Sobald Sie das Fenster schließen, sollte sich die Anwendung beenden.

3. Erste Rechtecke

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

Nachdem das Fenster erstellt ist (`gip_gfx_create_window`) und bevor Sie die Kontrolle an *gip_gfx* übergeben (`gip_gfx_render_loop`), können Sie in das Fenster zeichnen. Hier werden Sie in diesem Praktikum farbige Rechtecke zeichnen.

(a) Unmittelbar bevor Sie das erste Rechteck zeichnen, rufen Sie die folgende Funktion auf:

```
void gip_gfx_begin_draw(void)
```

Durch diesen Aufruf bereitet *gip_gfx* alles Nötige vor, damit Sie Rechtecke zeichnen können.

(b) Ein farbig gefülltes Rechteck können Sie durch entsprechenden Aufruf folgender Funktion zeichnen:

```
void gip_gfx_fill_rect(int start_x, int start_y, int width, int height,
                       uint8_t r, uint8_t g, uint8_t b)
```

`start_x` gibt die x-Koordinate (horizontal), `start_y` die y-Koordinate (vertikal) der linken oberen Ecke des Rechtecks an. `width` und `height` geben die Breite und die Höhe des Rechtecks an. Der Nullpunkt des Fensters liegt links oben.
`r` (Rot-Wert), `g` (Grün-Wert) und `b` (Blau-Wert) geben die Farbe des Rechtecks im RGB-Farbraum[12] an. `r`, `g` und `b` liegen dabei im Bereich von 0 bis 255.

(c) Unmittelbar nachdem Sie alle Rechtecke gezeichnet haben, rufen Sie die folgende Funktion auf:

```
void gip_gfx_end_draw(void)
```

Durch diesen Aufruf bereitet *gip_gfx* alles Nötige vor, damit die Zeichnung während des Aufrufs von `gip_gfx_render_loop` im Fenster angezeigt werden kann.

(d) Zeichnen Sie mehrere farbige Rechtecke.

4. Warming Stripes

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

(a) Kopieren Sie nun den Code zum Lesen der Eingabedatei mit den Temperaturdaten aus Ihrem vorherigen Projekt. Das grobe Format kennen Sie bereits. Inhaltlich exakter lautet es

```
ZEILENANZAHL
JAHR1,ANOMALIE1
JAHR2,ANOMALIE2
...
JAHRn,ANOMALIEn
```

[12] `https://en.wikipedia.org/wiki/RGB_color_model`

`ZEILENZAHL` gibt an, wie viele Datenzeilen (außer dieser Zeile) die Datei enthält. `JAHR1` bis `JAHRn` sind die Jahreszahlen (1850 bis 2023) für die eine Anomalie in der Datei gespeichert ist. `ANOMALIE1` bis `ANOMALIEn` sind die Temperatur-Anomalien des entsprechenden Jahres. Dabei handelt es sich um die Abweichung der jeweiligen mittleren Jahrestemperatur vom Mittelwert der mittleren Jahrestemperaturen der Referenzperiode 1961–1990[13].

(b) Ermitteln Sie mit Ihrem Ergebnis des vorherigen Praktikums die minimale und maximale Temperatur-Anomalie.

(c) Normieren Sie mit Hilfe des eben ermittelten Mimimums und Maximums nun im Programm in diesem Praktikum die eingelesenen Temperatur-Anomalien auf einen Wertebereich von 0 bis 1.

(d) Zeichnen Sie mithilfe der Funktion `gip_gfx_fill_rect` für jeds Jahr ein Rechteck, sodass alle Rechtecke nebeneinander liegen und das Fenster vollständig gefüllt ist. Verwenden Sie als Farbe für das Rechteck eine Graustufe, die durch die normierte Anomalie festgelegt ist; 0 wird schwarz gezeichnet, 1 wird weiß gezeichnet.

(e) Verwenden Sie nun folgende Farbskala:

Anomalie	R	G	B	Anomalie	R	G	B
$\leq -\frac{7}{8} \cdot 0.75$	8	48	107	$(0, \frac{1}{8} \cdot 0.75]$	254	224	210
$(-\frac{7}{8} \cdot 0.75, -\frac{6}{8} \cdot 0.75]$	8	81	156	$(\frac{1}{8} \cdot 0.75, \frac{2}{8} \cdot 0.75]$	252	187	161
$(-\frac{6}{8} \cdot 0.75, -\frac{5}{8} \cdot 0.75]$	33	113	181	$(\frac{2}{8} \cdot 0.75, \frac{3}{8} \cdot 0.75]$	252	146	114
$(-\frac{5}{8} \cdot 0.75, -\frac{4}{8} \cdot 0.75]$	66	146	198	$(\frac{3}{8} \cdot 0.75, \frac{4}{8} \cdot 0.75]$	251	106	74
$(-\frac{4}{8} \cdot 0.75, -\frac{3}{8} \cdot 0.75]$	107	174	214	$(\frac{4}{8} \cdot 0.75, \frac{5}{8} \cdot 0.75]$	239	59	44
$(-\frac{3}{8} \cdot 0.75, -\frac{2}{8} \cdot 0.75]$	158	202	225	$(\frac{5}{8} \cdot 0.75, \frac{6}{8} \cdot 0.75]$	203	24	29
$(-\frac{2}{8} \cdot 0.75, -\frac{1}{8} \cdot 0.75]$	198	219	239	$(\frac{6}{8} \cdot 0.75, \frac{7}{8} \cdot 0.75]$	165	15	21
$(-\frac{1}{8} \cdot 0.75, 0]$	222	235	247	$> \frac{7}{8} \cdot 0.75$	103	0	13

5. Neuer Referenzzeitraum

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

Für die Temperatur-Anomalien seiner *Warming Stripes* wählte Ed Hawkins den Referenzzeitraum 1971–2000. Um dieselben Farbstreifen zu erhalten, ist es erforderlich, die eingelesenen Anomalien zuerst umzurechnen und dann zu zeichnen.

Bestimmen Sie dazu zunächst die mittlere Temperatur-Anomalie der Jahre 1971–2000. Ziehen Sie dann diesen Mittelwert von sämtlichen eingelesenen Temperatur-Anomalien ab. Danach verwenden Sie die Farbskala von oben.

Wenn Sie diese Schritte in einem Programmdurchlauf ausführen wollen, durchlaufen Sie die Daten zweimal: Einmal zum Ermitteln der mittleren Anomalie, einmal zum entsprechenden Umrechnen und Zeichnen. Finden Sie einen möglichst einfachen Weg, Ihr Programm die Eingabedaten zweimal durchlaufen zu lassen.

Für Fortgeschrittene: Laden Sie die Daten zunächst in ein Array.

IV. Weiterführende Aufgaben

Mit den folgenden Aufgaben können sie weiter üben.

6. Temperatur-Anomalie zusätzlich als Balkenhöhe

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

Eine alternative Visualisierung der Temperatur-Anomalien stellt diese als farbiges Balkendiagramm dar[14]. Auf diese Weise wird der weltweite Temperaturanstieg noch offensichtlicher.

Zeichnen Sie dazu negative Temperatur-Anomalien von der vertikalen Mitte (Anomalie-Nullpunkt) nach unten, positive Anomalien nach oben. Die Balkenhöhe entspricht dabei dem Wert der Temperatur-Anomalie. Orientieren Sie sich für die jeweilige Balkenhöhe am zuvor ermittelten Minimum und Maximum, sodass kein Balken abgeschnitten wird. Verwenden Sie für negative und positive Anomalien dieselbe Skalierung.

[13] `https://www.metoffice.gov.uk/hadobs/hadcrut5/data/current/download.html`
[14] `https://showyourstripes.info/b`

7. CO_2-Gehalt der Atmosphäre als Balkenhöhe

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

Um die Korrelation zwischen Temperaturanstieg und CO_2-Gehalt der Atmosphäre zu veranschaulichen, können Sie mit der Balkenhöhe auch den CO_2-Gehalt darstellen. Dazu benötigen Sie allerdings weitere Daten. Diese finden Sie auf den Webseiten der der *European Environment Agency*[15] sowie des *Global Monitoring Laboratory* für das Mauna-Loa-Observatorium[16] in Hawaii.

Fügen Sie diese Datensätze so zusammen, dass diese zu den Datensätzen der Temperatur-Anomalien passen: Schränken Sie z. B. die abgedeckten Zeiträume entsprechend ein, erzeugen Sie fehlende Daten beispielsweise durch lineare Interpolation etc. Schreiben Sie dazu ggf. weitere Programme, die die entsprechende Daten-Vorverarbeitung übernehmen.

[15] `https://www.eea.europa.eu/data-and-maps/daviz/atmospheric-concentration-of-carbon-dioxide-5`
[16] `https://gml.noaa.gov/ccgg/trends/mlo.html`