Digitales Sommersemester 2020:

Datenvisualisierung und GPU-Computing

Übung 1: Eine Datenquelle

21.04.2021

Andreas Beckert

andreas.beckert@uni-hamburg.de

Regionales Rechenzentrum, Visual Data Analysis Group,

FB Informatik Arbeitsgebiet „Scientific Visualization and Parallel Processing“

Termine

- 4-stündige Übung alle 2 Wochen Mittwochs um 14:00 Uhr, mit Ausnahme der ersten 2 Wochen
- Termine:
  - Ü1: 14.04.2021, 14:00 – 16:00
  - **Ü2: 21.04.2021, 14:00 – 16:00**
  - Ü3: 05.05.2021, 14:00 – 18:00
  - Ü4: 19.05.2021, 14:00 – 18:00
  - Ü5: 02.06.2021, 14:00 – 18:00
  - Ü6: 16.06.2021, 14:00 – 18:00
  - Ü7: 30.06.2021, 14:00 – 18:00

Ziel und Ablauf der Übung

Ziel: Umsetzung von in der Vorlesung gelernten theoretischen Inhalten in praktischen Programmierübungen.

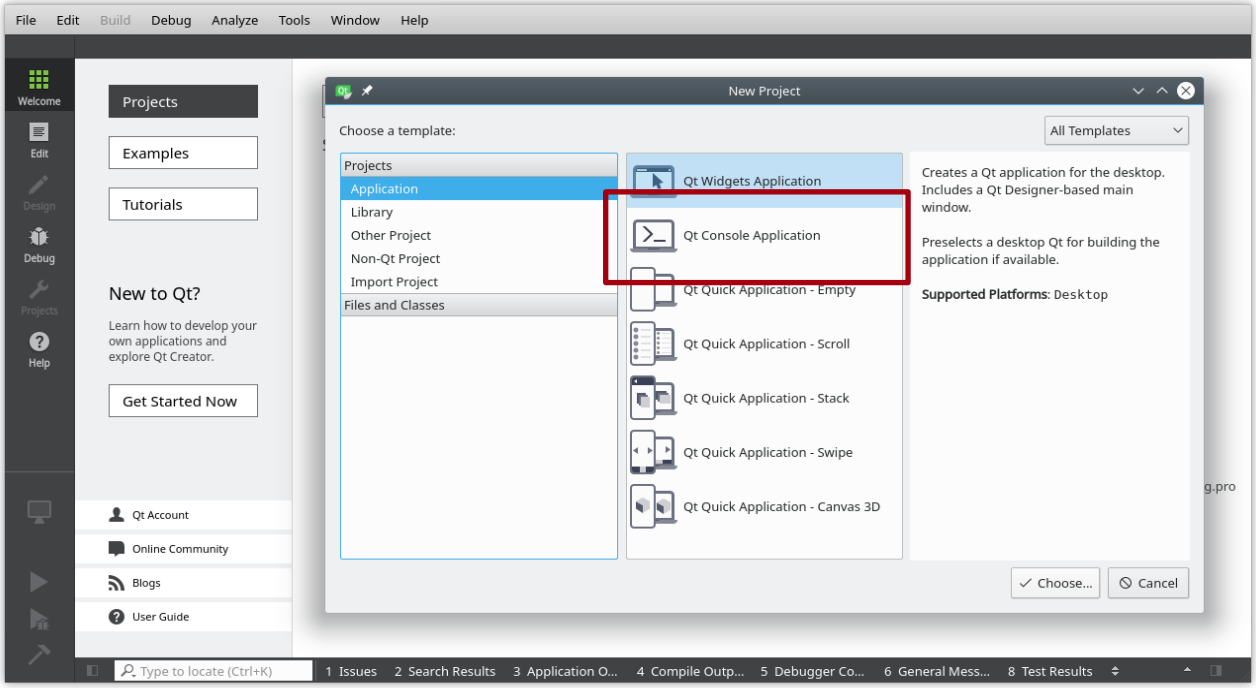
Ablauf: Konzeption und Implementierung eines simplen Visualisierungssystems in C++ mit OpenGL.

- Programmieren mit C++ und Qt
- Programmieren mit OpenGL
- Konzeption und Implementierung einzelner Stages der Visualisierungs-Pipeline: Datenquelle, Mapper, Rendering

- Break-Out Rooms?

Praktischer Test #2

- Neues „Qt Console“ Projekt
  - Build system = qmake
- Ändere die Vorlage in unser „Hello World“ Beispiel
- Kompilieren und starten
  - Erste Einstellung: Output unten in „Application Output“ anzeigen



Praktischer Test #2

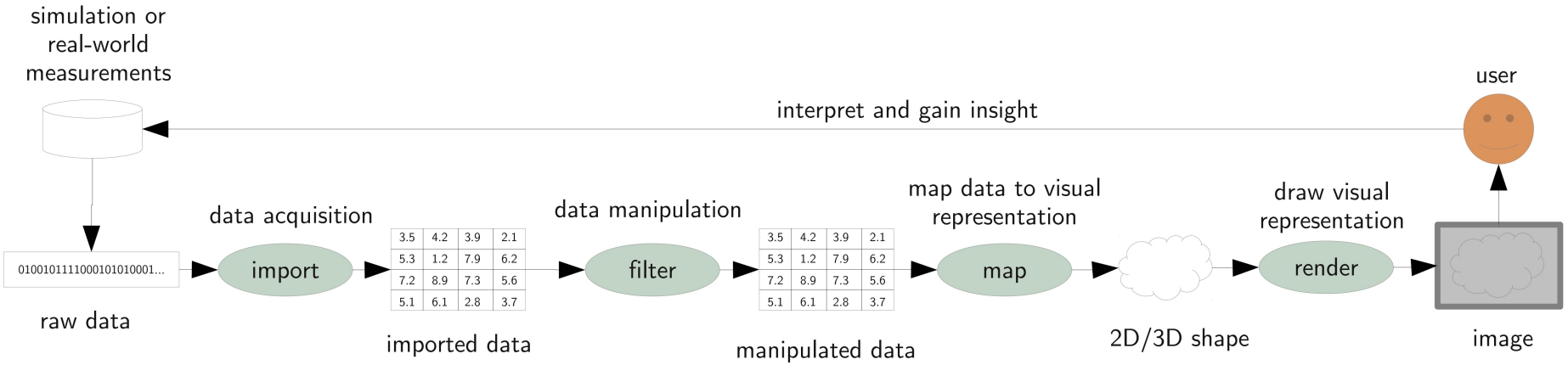
- „Hello World“ im QtCreator:

```
1 // my first program in C++
2 #include <iostream>
3
4 int main()
5 {
6   std::cout << "Hello World!";
7 }
```

Basics C++ – Anwendung im QtCreator

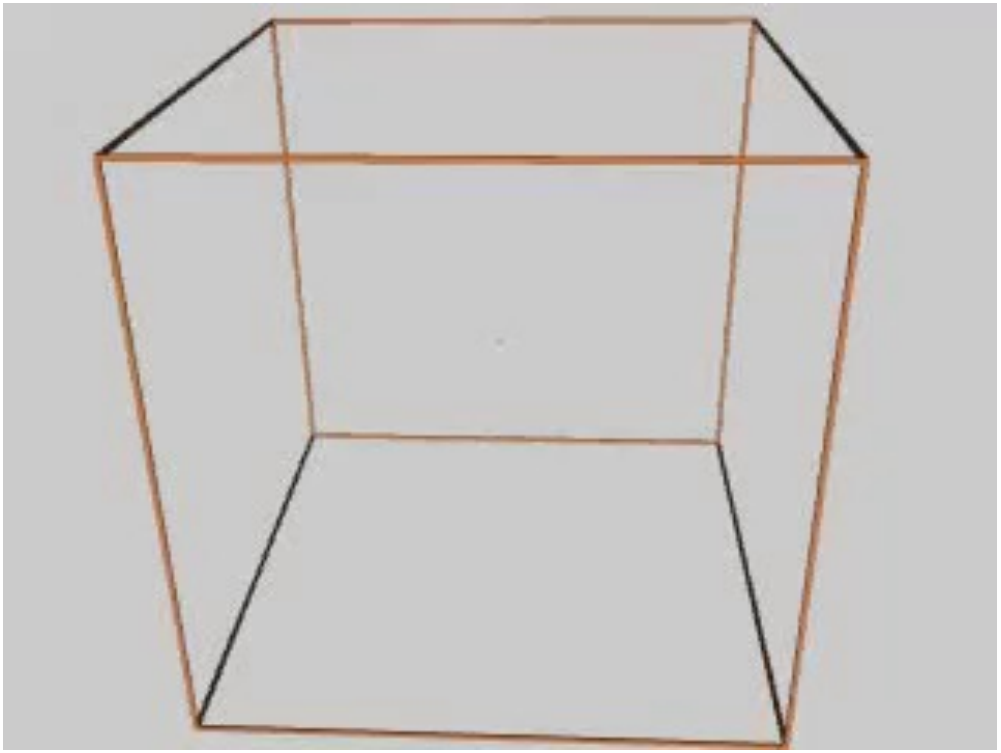
- Was ist ein „Projekt“ (was in etwa passiert unter der Haube)?
- Zufügen von vorhandenen und erzeugen neuer Dateien/Klassen zu einem Projekt
- Window-Layouts: Split-Ansicht
- Shortcuts: Umschalten zwischen Header und Source, Kompilieren, Starten
- Erzeugen neuer Methoden
- Quellcode-Konventionen („CamelCase“, Klammersetzung, etc.)
- Hilfe-Funktionen

Übung 1: Implementierung einer Datenquelle



Übung 1: Implementierung einer Datenquelle

Wir wollen ein erstes C++ Programm entwerfen, welches eine Datenquelle eines „synthetischen Tornados“ implementiert. Die Daten aus dieser Quelle wollen wir in den Folgeübungen auf unterschiedliche Weise visualisieren.



Übung 1: Implementierung einer Datenquelle

Wir wollen ein erstes C++ Programm entwerfen, welches eine Datenquelle eines „synthetischen Tornados“ implementiert. Die Daten aus dieser Quelle wollen wir in den Folgeübungen auf unterschiedliche Weise visualisieren.

- Erstelle ein Programmgerüst mit dem QtCreator.
- Füge eine neue Klasse FlowDataSource zu, die folgende „public“ Methoden enthält:
  - createData(…)– diese Methode soll ein „private“ Array cartesianDataGrid, welches ein kartesisches Gitter vom Typ float repräsentiert, mit Werten befüllen.
  - getDataValue(int iz, int iy, int ix, int ic)– diese Methode soll den Wert an Gitterposition (iz, iy, ix) für Komponente ic zurückgeben.
  - printValuesOfHorizontalSlice(int iz)– diese Methode soll das vertikale (also in z-Dimension) Layer „iz“ des Gitters als Fließkommazahlen auf der Konsole ausgeben.
- Überlege zunächst konzeptionell (d.h. Pseudocode auf Papier), wie die Klasse und ihre Methoden aufgebaut sein sollen.

Übung 1: Datenquelle

Für die Datenerzeugung in createData() bedienen wir uns einer Funktion, die in der Grafik- und Visualisierungsforschung weit verbreitet ist. Der „synthetische Tornado“ von Prof. Roger Crawfis (Ohio State University) erzeugt ein einfaches Vektorfeld („Wind“) mit drei Raumkomponenten auf einem kartesischen Gitter. Der C-Code (Achtung: kein C++) ist hier verfügbar: http://web.cse.ohio-state.edu/~crawfis.3/Data/Tornado/

- Wie sind die Funktionsargumente dieser Funktion gestaltet?
- Wie können wir diese Funktion in unser Programm integrieren, möglichst ohne sie zu verändern?
- Welche Array-Repräsentation wählen wir am besten für unser kartesisches Gitter cartesianDataGrid, um darin Werte aus der Crawfis-Funktion zu speichern (wie wird das Array in der Crawfis-Funktion beschrieben)? Welche Auswirkung hat das auf die Implementierung von getDataValue?
- Ziel: Für ein Gitter von 16^3 Gitterpunkten, welche Werte enthält Layer 10?

Arrays in C++

multidimensional array

```
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT][WIDTH];
int n,m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n][m]=(n+1)*(m+1);
    }
}
```

pseudo-multidimensional array

```
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT * WIDTH];
int n,m;

int main ()
{
  for (n=0; n<HEIGHT; n++)
    for (m=0; m<WIDTH; m++)
    {
      jimmy[n*WIDTH+m]=(n+1)*(m+1);
    }
}
```

Dynamischer Speicher

- http://www.cplusplus.com/doc/tutorial/dynamic/

```
 8  int i,n;
 9  int * p;
10  cout << "How many numbers would you like to type? ";
11  cin >> i;
12  p= new (nothrow) int[i];
13  if (p == nullptr)
14    cout << "Error: memory could not be allocated";
15  else
16  {
17    for (n=0; n<i; n++)
18    {
19      cout << "Enter number: ";
20      cin >> p[n];
21    }
22    cout << "You have entered: ";
23    for (n=0; n<i; n++)
24      cout << p[n] << ", ";
25    delete[] p;
26  }
```

```
1 int * foo;
2 foo = new int [5];
```

```
1 delete pointer;
2 delete[] pointer;
```

Übung 1: Zusatzfragen

Das von der Crawfis-Funktion erzeugt Datenfeld enthält synthetische „Windgeschwindigkeiten“ in den drei Raumrichtungen.

- Wie unterscheiden sich die drei Windkomponenten in ihrer Größenordnung?
- Schreibe eine Funktion, die den Betrag der 3D-Windgeschwindigkeit berechnet. Wie unterscheidet sich diese Größe von den einzelnen Komponenten?
- Wie groß sind Minimum und Maximum der 3D-Geschwindigkeit bei Zeitschritt t=0?
- Wie ändern sich diese Werte mit der Zeit?

Ausblick

Nächste Übung:

- Integration eines OpenGL-Displays in unser Programm.
- (Versionskontrolle mit git)
- Ziel: Erste visuelle Darstellung unseres Datenfeldes, denn: *„The purpose of computing is insight, not numbers.“* – Richard Hamming (1962, Numerical Methods for Scientists and Engineers)