

Computergraphik 1. Hausaufgabe

Dokumentation

Sascha Ebert

MatNr: 177182

`sascha.ebert@s2006.tu-chemnitz.de`

`https://github.com/Sighter/CG1_HA1`

26. November 2010

1 Anwendung

Das Programm wird mit der folgenden Syntax aufgerufen:

```
>> cgl_ha1 <argument>
```

Wobei <argument> mit einem Dateiname eines Files des Typs `cgp1` zu ersetzen ist. Der Aufruf darf derzeit nur ein Argument besitzen.

Dies erstellt nun eine Datei mit dem Schema: `filename.tga`. Ist jedoch eine Datei gegeben die nicht die Endung `'.cgp1'` besitzt, wird die Ausgabedatei unter dem Standardname `'output.tga'` gespeichert.

Da das Programm unter Linux entstanden ist, ist es leider nicht möglich eine `.exe`-Datei mit anzuhängen. Allerdings liegt ein `makefile` bei, aus welchem sie die genauen Compiler-Kommandos entnehmen können.

2 Erklärungen

Ich habe mich für die C++-Implementation der Aufgabe entschieden, da es den aktuelleren und sicheren Stand der Dinge widerspiegelt. Im Folgenden sind die Quelltextdateien und die dazugehörigen Datenstrukturen jeweils kurz erläutert.

2.1 `cgl_ha1.h`

In dieser Datei sind die Klassen-*Deklarationen* für die Klassen `cgp1` und `puzzleBlock` platziert. Des weiteren finden sich hier Typendefinitionen, um mit den von ihnen vorgegebenen Spezifikationen besser umgehen zu können (`UBYTE`, `USHORT`, `UINT`)

2.1.1 Class: `cgp1`

Die Klasse `cgp1` repräsentiert das gleichnamige Format. Sie beinhaltet eine Methode zum lesen des Eingabe-Files, welche die Membervariablen der Klasse mit den korre-

spondierenden Informationen füttert. Zum schreiben wird die Methode `writeToTga()` verwendet. Als drittes wichtiges Element enthält die Klasse ein Feld von Zeigern auf `puzzleBlock`-Instanzen welche die einzelnen Puzzelblöcke verwalten.

2.1.2 Class: `puzzleBlock`

Wie oben genannt, wird diese Klasse benutzt um die relevanten Informationen für einzelne Puzzelblöcke zu speichern. Das wichtigste Element hier ist das 2-Dimensionale Feld `UBYTE **m_pPixelField` um die Pixeldaten abzulegen. Dabei wird das Zeigerkonzept verwendet um die Daten dynamisch im Heap zu organisieren.

2.2 `cg1_ha1.cpp`

In dieser Quellcodeldatei werden die Methoden, Initiatoren und Destruktoren für die 2 Haupt-Klassen ¹ *definiert*. Allerdings war es nötig einen Destruktor manuell aufzurufen um Valgrind² zufrieden zu stellen.

Des weiteren sind jeweils `print`-Funktionen definiert, welche ich zum Kontrollieren des Lesevorgangs verwendet habe. Diese werden allerdings nicht mehr aufgerufen.

2.3 `cg1_ha1_main.cpp`

In dieser Quelldatei wird die `Main`-Funktion des Programmes definiert. Genauer werden hier die Programmargumente ausgewertet und unter Nicht-Verwendung der `string`-library (weder `c` noch `c++`) recht umständlich der Dateiname umgeschrieben um die Endung `.cg1` in `.tga` zu wandeln.

Ich bitte vielmals um Entschuldigung ihr Benennungsschemata nicht eingehalten zu haben, aber - ohne verbesserlich sein zu wollen - eine `main`-Funktion gehört nun einmal in eine separate Datei. Standards sind ja nu mal da um eingehalten zu werden.

¹Meines Erachtens ist es nach ISO-Standard vorgesehen für jede Klasse ein Header-File und ein `.cpp`-File zu benutzen.

²Programm zum finden von Memory-Leaks