TE3D – Eine 3D Terminal Engine

TE3D soll es ermöglichen, Grafiken im Terminal anzuzeigen (sowohl 2D als auch 3D). Die grafischen Funktionalitäten für das Zweidimensionale soll SDL (Version 2) übernehmen, worunter die Verwaltung von Zeichenflächen und Grafikfunktionen (z. B. das Zeichnen einer Linie) fällt. Die 3D-Engine wird eigenhändig geschrieben und in Form eines Software-Rasterizers implementiert, auf Hardware wird also bei dem Rendering-Prozess verzichtet. Die Engine soll die gängigsten Funktionalitäten unterstützen wie Modellverwaltung, Tiefenpuffer und Kamerasysteme.  
Nach dem Rendering in eine Bitmap wird das Bild mithilfe eines „ASCII-Art“-Converters in Zeichen umgewandelt, die dann ins Terminal geschrieben werden. Dabei soll auch Farbe unterstützt werden (mithilfe von Terminal-Steuerzeichen).

TE3D soll so wenig externe Bibliotheken verwenden wie möglich und plattformunabhängig sein.

*Design*

TE3D ist in verschiedene Layer aufgeteilt, in ihrer Gesamtheit als Pipeline bezeichnet. Die Hauptlayer in der Pipeline sind:

* Terminalausgabe

Die Terminalausgabe wird mit einer Renderfunktion (void TE3D\_Render()) erreicht, die einen Zeichenpuffer anspricht, der ein Array von ASCII-Zeichen (Datentyp char) und deren entsprechende Farbe beinhaltet (Struktur TE3D\_Surface). Nach dem Rendern wird die Position des Terminalstreams zurückgesetzt und es kann erneut gerendert werden. Die Renderfrequenz kann vom Benutzer angegeben werden.

* ASCII-Converter

Der ASCII-Converter ist für die Konvertierung von Bild zu Text verantwortlich. Mit  
void TE3D\_ASCII\_Convert(struct SDL\_Surface image, struct TE3D\_Surface target)  
soll ein Bild in ein ASCII-Art-Bild konvertiert werden.

* SDL 2 Schnittstelle + 2D Rasterizer

Die Schnittstelle kapselt alle nötigen Methoden zur Bildflächenverwaltung (SDL\_Surface) und Grafikfunktionen für zweidimensionale Zeichnungen. Für das 2D-Zeichnen gibt es das Pre- und Postrendering, d. h. es kann gezeichnet werden bevor der 3D-Rasterizer mit seiner Zeichnung beginnt und hinterher. Damit können z. B. Hintergrundbilder gemalt werden.

* 3D Rasterizer

Der 3D-Software-Rasterizer ist zuständig für alle 3D Funktionalitäten und besitzt ebenfalls eine Pipeline.

* + Rendering

Malt die fertige Projektion der Vektoren in die SDL\_Surface der Schnittstelle.

* + Texturierung und Pixeltransformation

Zuständig für die Ausfüllung von Flächen (nur Farbe, keine Bitmap-Texturen) und weiterer Pixeltransformationen, die vom Benutzer implementiert werden können (Schattierung, Tiefenpuffer-Effekte etc.). Die Transformation kann entweder mithilfe von Callback-Funktionen oder Matrizen realisiert werden, wobei Callback-Funktionen hierfür vermutlich die effizientere Lösung wären.

* + Vektortransformation

Transformiert die eingegebenen Vektoren. Meistens wird hier eine Projektionsmatrix verwendet um die Vektoren auf eine Ebene abzubilden. Die Matrizen sind Strukturen mit den Bezeichnungen TE3D\_Matrix4x4f, TE3D\_Matrix4x4f etc. und Vektoren werden TE3D\_Vector4f, TE3D\_Vector3f etc. genannt. In der entsprechenden Code-Datei sind alle nötigen Funktionen für mathematische Rechnungen mit diesen enthalten.

* + Verwaltung und Speicherung der Vektoren

Regelt das Speichern der Vektoren, besonders die Speicherung von Vektorgruppen (Modellen), auf die einzelne Transformationen angewendet werden kann. Zur Modellverwaltung werden verkettete Listen verwendet.

*Quellcodestil*

* Verwendung von // für wenig-zeilige Kommentare.
* Kommentare auf Englisch.
* Beschreibende Kommentare über den jeweiligen Funktionen (Beschreibung der Funktionalität, Parameter etc.)
* „{“ in neuer Zeile.
* „TE3D“ als Präfix für alle öffentlichen Funktionen.

*Projektstruktur*

|  |  |
| --- | --- |
| graphics3d.c / graphics3d.h | 3D-Rasterizer |
| graphics.c / graphics.h | 2D-Rasterizer und SDL Schnittstelle |
| ascii\_converter.c / ascii\_converter.h | ASCII-Art-Converter |
| list.c / list.h + stack.c / stack.h | Verkette Listen |
| core.c | Kernfunktionalität |

*Einteilung*

|  |  |
| --- | --- |
| 3D-Rasterizer | Mischa Krüger |
| 2D-Rasterizer und Schnittstelle | Frank Zimdars |
| ASCII-Art-Converter | Gordon Kemsies |
| Kern und verkettete Listen | Ammar Al-Qaiser |