

## Übungen zu Graphikprogrammierung in C++

### Abgabetermin: 20. Juni 2005

Am 21. Juni wird von jedem Team ca. 5 Minuten die Lösung der Aufgabe 22 in dem wöchentlichen Treffen präsentiert. Jedes Team wählt hierfür einen Präsentator aus.

*Was man wissen muss:*

- *Texture Mapping*
- *Modelview Matrix*
- *Display Lists*
- *glut Interaktions- & Darstellungsfunktionalität*

### Aufgabe 21 (H) Partikelsysteme - Vulkanausbruch

Mit Partikelsystemen kann man Bewegungen bzw. Krafteinwirkungen auf Objekte einer Szene modellieren. Beispielsweise könnte man Regen, Schneetreiben oder Bewegungen von Schilf im Wind visualisieren. In dieser Übung soll ein Vulkanausbruch mittels Partikelsystemen modelliert werden. Überlegen sie sich ein Modell, dass das Schleudern eines Gesteinsbrocken aus der Erde simuliert. Gestalten sie ihr Programm so, dass durch Benutzereingaben die Intensität des Vulkans verändert werden kann.

- Entwerfen Sie eine Klassenhierarchie mit der Oberklasse `Particle` und einer Unterklasse `Lava`. In `Particle` sollen alle Attribute und Methoden implementiert werden, die Partikel gemeinsam haben, d.h. Position, Geschwindigkeit und Beschleunigung (*engl.: acceleration*). Sie können auch noch Variablen implementieren, welche bestimmen, wie lange ein Partikel überlebt, ob ein Partikel aktiv ist oder welche Farbe ein Partikel hat (anhand des Rot-, Grün- und Blauwertes).  
Es soll zwei abstrakte Methoden geben, `update( float s )` und `render( void )`, die in der Unterklasse implementiert werden sollen. Verwenden Sie für das Rendering eines Steins eine passende 3D-Komponente<sup>1</sup>.
- Implementieren Sie eine Klasse `ParticleEmitter`, die mehrere Partikel verwalten kann. Diese soll möglichst einfach auf die Attribute der Klasse `Particle` zugreifen können. Sie können hierzu entweder strikt objektorientiert geeignete Getter- und Settermethoden verwenden, oder das *friend*-Konzept verwenden. Ist eine Klasse `A` in einer Klasse `B` als *friend* deklariert, so kann `A` auf alle `private` und `protected` Attribute der Klasse `B` zugreifen. Dies hat nebenbei auch noch den Vorteil, dass nun nur die Klasse `A` auf die privaten Attribute zugreifen kann, andere Klassen allerdings nicht (weil man keine Getter- und Settermethoden implementiert hat).

---

<sup>1</sup>in `glut` sind schon vorgefertigte Strukturen implementiert, z.B. `glutSolidTeapot`, `cubes`, `spheres` oder ähnliches

Schreiben Sie eine main-Funktion und ein Fenstersystem mit glut in der Datei `mainA21.cpp`<sup>2</sup>. Hierbei sollen v.a. display- und keyboard-Funktionen definiert werden. Mit der update-Funktion soll eine Animation simuliert werden. Diese soll immer stattfinden, nicht nur wenn man eine bestimmte Benutzereingabe macht. Implementieren Sie dazu eine geeignete Funktion `idle()` und übergeben Sie diese an glut mit `glutIdleFunc(idle);`.

- c) Nun soll eine Intensitätsanzeige implementiert werden. Mittels den Tasteneingaben `i` und `Shift+i` soll die Intensität des Vulkans verringert, bzw. erhöht werden<sup>3</sup>. Zeigen Sie die Intensität in dem glut-Fenster an. Dies kann folgendermaßen implementiert werden. Zuerst sollten alle Attribute (z.B. `GL_CURRENT_BIT`, usw.) auf den Stack gelegt werden (mit `glPushAttrib(GL_ALL_ATTRIB_BITS)`). Dann kann ein String erstellt werden und dieser mittels `glRasterPos2f(...)` an die richtige Position gesetzt werden. Mit der Funktion `glutBitmapCharacter(GLUT_BITMAP_8_BY_13, string[i]);` kann der String gezeichnet werden, wobei jeder einzelne Character durchlaufen werden muss.

## Aufgabe 22 (H) Der Grand Canyon

In dieser Aufgabe soll anhand von Höhendaten der Grand Canyon modelliert werden. Hierfür wird das Konzept von Shading eingeführt. Das Shading ist dafür zuständig, je nach Lage des Oberflächenpatches und unter Verwendung der zuvor definierten Lichtquelle die Polygone mit authentischen Intensitätswerten zu rendern. Das Modell des Canyons besteht aus einer Höhenkarte (Graustufen Bitmap) und einer zugehörigen Texture Map (RGB Bitmap). Die Intensitätswerte der Höhenkarte repräsentieren den z-Wert zu den zugehörigen x,y-Werten. In der Texture Map finden sie die RGB-Farbwerte für jedes Oberflächen-patch. Die Daten können Sie im SVN unter `progprakt/solution/data` finden. `gcanyon.bmp` enthält die Höhendaten, `gcanyonTex.bmp` die Texture Map.

- Als erste Teilaufgabe sollen die Höhendaten gerendert werden. Erstellen Sie ein OpenGL Programm basierend auf den vorhergehenden Übungen. Lesen Sie das Bitmap File mit den Höhendaten ein. Um die Karte zu rendern, werden jeweils nebeneinander liegende Punkte als Viereck (`GL_QUADS`) gezeichnet. Die x und z Koordinaten entsprechen den x und y Koordinaten aus der Höhenkarte, die y Koordinate erhalten Sie aus dem Intensitätswert. Höhere Punkte sollen heller gezeichnet werden, tiefer gelegene dunkel. Wahlweise können Sie auch den verschiedenen Höhen Farben zuweisen (tief: grün, hoch: grau).
- Implementieren Sie jetzt Interaktionen durch Mauseingaben. Der Benutzer soll das Modell rotieren, verschieben und skalieren können.
- Um das Modell realer zu gestalten, soll nun anstatt der Farben die Texture Map auf die Oberfläche des Canyons gelegt werden. Laden Sie dazu die Map in ein `CAMP::Bitmap` und initialisieren Sie eine Textur damit. Mit `glTexCoord2f` sprechen Sie verschiedenen Koordinaten aus der Texture Map an. Beachten Sie, dass diese im Bereich `[0;1]` liegen.
- Sie werden vielleicht festgestellt haben, dass das Rendering des Modelles viel Rechenzeit beansprucht. Um dies zu beschleunigen, können Sie DisplayLists verwenden. OpenGL stellt dazu die Funktionen `glNewList`, `glEndList` und `glCallList` zur Verfügung.
- Damit wir unserem Ziel, ein Spiel zu programmieren, näher kommen, soll nun ein Fly-Through Modus implementiert werden. Befindet man sich in diesem Modus, kann der Benutzer mit den Cursor Tasten nach links/rechts, oben/unten fliegen und durch 'a' und 'y' beschleunigen und bremsen.

---

<sup>2</sup>Sollten Sie Hilfe mit glut benötigen, können Sie auf die main-Funktionen der letzten Aufgabenblätter zurückgreifen oder im Internet unter <http://www.lighthouse3d.com/opengl/glut/> die Funktionalitäten nachlesen

<sup>3</sup>Shift-Abfrage mit glut: `GLUT_ACTIVE_SHIFT`

- Bis jetzt passiert noch nicht viel in unserer kleinen Welt. Um das zu ändern, binden Sie den Partikel Vulkan aus der vorhergehenden Aufgabe ein. Setzen Sie diesen auf eine Bergkuppe und lassen Sie es brodeln.