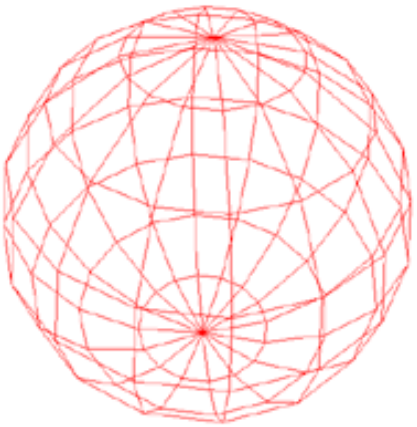
HASSO – PLATTNER – INSTITUT
für Softwaresystemtechnik GmbH

Prof. Dr. rer. nat. habil. Jürgen Döllner
Dipl.-Math. Florian Kirsch
Dipl.-Math. Marc Nienhaus

Aufgabe 3: Tessellation einer Kugel (5 + 5 Punkte)

Beschreiben Sie zwei Verfahren zur Tessellation einer Kugeloberfläche. Die Kugel habe den Radius *r* und ihr Mittelpunkt liege im Koordinatenursprung.

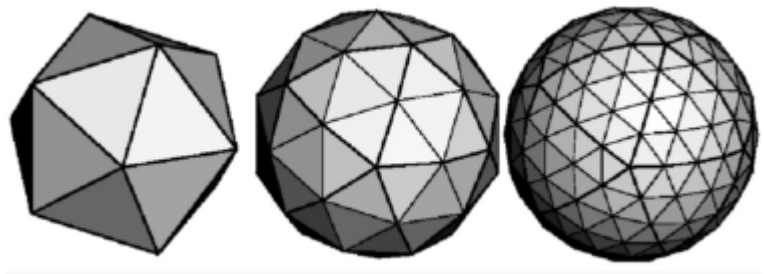


Tessellation einer Kugel mit Vierecken

a) Beschreiben Sie die Oberfläche mathematisch durch eine Viereckstessellation: topologisch betrachtet verlaufen die linken und rechten Kanten der Vierecke in Nord-Süd-Richtung, während die oberen und unteren Kanten senkrecht dazu verlaufen, also von Ost nach West. Die Tessellation soll also anschaulich der Einteilung der Kugeloberfläche in Breiten- und Längengrade entsprechen.

(Hinweis: Trigonometrische Funktionen)

b) Konstruieren Sie rekursiv die Kugeloberfläche durch eine Dreieckstessellation. Gehen Sie dabei von einem Tetraeder (vierseitiger Körper, dessen Seitenflächen jeweils kongruente gleichseitige Dreiecke sind) aus und unterteilen Sie in jedem Rekursionsschritt die Dreiecke des Tetraeders in vier gleich große Dreiecke. Die Tiefe der Rekursion soll durch einen Parameter festgelegt werden. Geben Sie die Koordinaten des Tetraeders an.



Tessellation einer Kugel mit Dreiecken

Beschreiben Sie beide Verfahren; Sie können zur Beschreibung die Verfahren in einer Pseudo-Code-Schreibweise formulieren.

Aufgabe 4: Implementierung (5* Punkte)

Implementieren Sie ein Tessellationsverfahren für Kugeloberflächen unter Verwendung des bereitgestellten Programmrahmens (`CGSphere`).

Die Bearbeitung dieser Aufgabe ist freigestellt und zum Erlangen der vollen Punktzahl aus den Übungsaufgaben nicht erforderlich.

Aufgabe 5: Airbrush (15 Punkte)

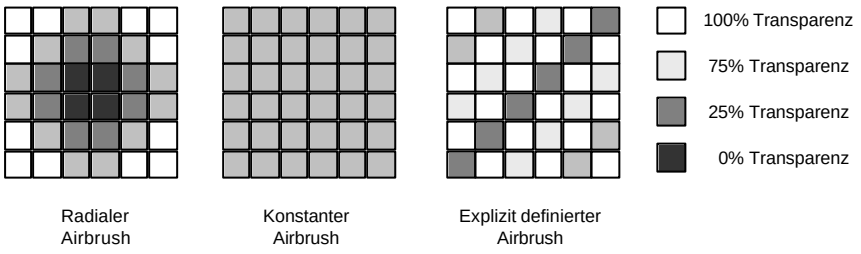
Ziel dieser Aufgabe ist es, ein pixelbasiertes Malprogramm zu erstellen. Zum Einsatz kommt ein *Airbrush*, d.h. eine digitale Sprühpistole, mit der interaktiv Farbe auf die Zeichenfläche aufgetragen wird. Verschiedene Muster und Stärken der Sprühpistole ergeben unterschiedliche

Prof. Dr. rer. nat. habil. Jürgen Döllner
Dipl.-Math. Florian Kirsch
Dipl.-Math. Marc Nienhaus

visuelle Effekte. Wird die gleiche Stelle auf der Zeichenfläche mehrfach übersprüht, dann addiert sich Farbe auf der Zeichenfläche. Bei gedrückter Maustaste wird die Sprühpistole an der momentanen Mausposition angewandt. Der Inhalt der Zeichenfläche wird nicht gespeichert: Wird die Zeichenfläche von einem anderen Fenster überdeckt, dann ist der Inhalt verloren (Single-Buffering).

Ein Airbrush wird intern durch ein 2D-RGBA-Array repräsentiert; übliche Array-Größen sind z.B. 16x16 oder 32x32. Die RGB-Komponenten legen die Farbe, die Alpha-Komponente die Transparenz und damit die Form und Farbstärke fest; Beispiel für Transparenzverteilungen sind:

- *Radiale Transparenzverteilung*: besitzt im Zentrum 0% und zum Rand hin 100% Transparenz.
- *Konstante Transparenzverteilung*: besitzt eine gleichbleibende, benutzerdefinierte Transparenz.
- *Explizite Transparenzverteilung*: wird aus einer Datei eingelesen.



Gegeben ist eine Klasse `CGAirbrush`, die Methoden zur Allokation und Verwendung des Airbrush-Arrays sowie eine Zeichenfläche mit einem Pixelkoordinatensystem bereitstellen soll. Implementieren Sie folgende Methoden:

- `CGAirBrush` : Konstruktor; legt die Größe des Airbrush fest.
- `setConstantPattern` : aktiviert einen Airbrush mit konstante Transparenz.
- `setRadialPattern` : aktiviert einen Airbrush mit radialer Transparenzverteilung.
- `setArbitraryPattern` : aktiviert einen Airbrush mit einer benutzerdefinierten Transparenzverteilung.
- `setColor` : legt die Zeichenfarbe fest, d.h. die RGB-Werte im Airbrush-Array.
- `onMove`, `onButton` : zeichnet mit der Sprühpistole an der momentanen Mausposition.
- `onInit` : schaltet das Blending ein; legt die Hintergrundfarbe fest.
- `onDraw` : löscht die Zeichenfläche.
- `onSize` : aktualisiert das Pixelkoordinatensystem.
- `onKey` : ‚c' löscht den Color-Buffer; ‚k' legt eine konstante, ‚r' eine radiale, ‚e' eine explizite Transparenzverteilung fest; ‚0'-‚7' aktivieren unterschiedliche Zeichenfarben.

Tipp: Durch Kopieren des Airbrush-Pixel-Arrays an eine bestimmte Raster-Position – bei aktivierten Blending – werden die Airbrush-Pixel auf die Zeichenfläche entsprechend der Transparenz zu den bereits vorhandenen Pixeln addiert. Es steht ein Programmrahmen mit den Dateien `cgairbrush.h` und `cgairbrush.cpp` bereit. Überlegen Sie, welche Attribute die Klasse `CGAirbrush` benötigt.

Aufgabe 3 ist alleine zu lösen, die Aufgaben 4 und 5 können zu zweit bearbeitet werden. Schicken Sie Ihre Lösungen bitte bis zum Donnerstag, den 10.5.2001, an `cgi2001@hpi.uni-potsdam.de`.