

Wintersemester 2011/12

Übungen zu Computergraphik I - Blatt 2

Abgabe am 16. 11. 2011

Aufgabe 1 (C++-Programmierung, 7+5+4+2 Punkte)

Die folgenden Aufgabenteile sollen in das Framework (siehe Webpage) integriert werden. Die Dateinamen sind gleich benannt wie die jeweilige Klasse. Wie in der C++-Wiederholung erklärt, sollen die Deklarationen in *.h und die Definition in *.cpp Dateien geschrieben werden.

- a) Implementieren Sie eine Klasse für $n \times m$ -Matrizen mit Floating-Point-Zahlen als Werten. Die Klasse sollte folgende Funktionalität bieten:

```
class Matrix
{
public:
    Matrix( unsigned int n, unsigned int m );
    Matrix( const Matrix& m );
    ~Matrix();

    float *operator [] ( unsigned int i );

    const Matrix& operator = ( const Matrix& m );

    const Matrix& operator += ( const Matrix& m );
    const Matrix& operator -= ( const Matrix& m );
    const Matrix& operator *= ( const Matrix& m );
    const Matrix& operator *= ( float s );

    void print();

protected:
    ... // Ueberlegen Sie sich die interne Instanzvariable
};

Matrix operator + ( const Matrix& lhs, const Matrix& rhs );
Matrix operator - ( const Matrix& lhs, const Matrix& rhs );
Matrix operator * ( const Matrix& lhs, const Matrix& rhs );
Matrix operator * ( const Matrix& m, float s );
```

Die Wirkung der Funktionen sei wie folgt definiert:

- `Matrix(unsigned int n, unsigned int m);` Der Konstruktor bekommt die Anzahl der Zeilen und Spalten der Matrix übergeben.
- `Matrix(const Matrix& m);` Kopierkonstruktor

- `~Matrix()`; Destruktor (Hinweis: Denken Sie daran, daß Sie im Konstruktor dynamisch Speicher alloziiert haben!)
- `float *operator [] (unsigned int i)`; Zugriffoperator, der den Zugriff mittels `a[i][j]` auf die Matrix erlaubt.
- `const Matrix& operator = (const Matrix& m)`; Zuordnungsoperator, der es erlaubt, Ordnung und Inhalt einer Matrix `m` zu kopieren.
- `const Matrix& operator *= (float s)`; bzw `Matrix operator * (const Matrix& m, float s)`; Multiplikation jedes Eintrags der Matrix mit dem skalaren Faktor `s`.
- `void print()`; Gibt die Matrix auf `stdout` aus.

Die restlichen Funktionen entsprechen der bekannten Matrizen-Multiplikation, -Addition bzw. -Subtraktion. Bei all diesen Operationen soll die Dimension der Matrizen überprüft und eine Fehlermeldung ausgegeben werden, wenn die Matrizen nicht kompatibel sind.

Achtung: Vermeiden Sie bei der Überladung der Operatoren mit fast gleicher Funktionalität redundanten Code (siehe C++ Wiederholung, Folie 41)

- b) Leiten Sie von der Matrix-Basisklasse eine Klasse `Vec3` zur Repräsentation von 3D-Vektoren ab. Der Header sollte folgendermassen aussehen:

```
class Vec3 : public Matrix
{
public:
    Vec3();
    Vec3( const Vec3& v );

    float& operator [] ( unsigned int i );

    float normalize( );
};

float operator * ( const Vec3& lhs, const Vec3& rhs );
Vec3 cross( const Vec3& lhs, const Vec3& rhs );
float norm2( const Vec3& v ) ;
```

Die Wirkung der Funktionen sei wie folgt:

- `float operator [] (unsigned int i)`; Zugriffoperator, der den Zugriff mittels `a[i]` auf den Vektor erlaubt.
- `float operator * (const Vec3& lhs, const Vec3& rhs)`; Liefert das Skalarprodukt zweier Vektoren.
- `Vec3 cross(const Vec3& lhs, const Vec3& rhs)`; Berechnet das Kreuzprodukt/Vektorprodukt zweier Vektoren.
- `float norm2(const Vec3& v)`; Gibt die euklidische Norm des Vektors zurück.
- `float normalize()`; Normalisiert einen Vektor und gibt zusätzlich die Norm zurück.

Bemerkung: In sogenanntem „production code“ würde man die Klasse `Vec3` nicht direkt von `Matrix` ableiten.

- c) Schreiben Sie ein kleines Programm zum Testen Ihrer Klassen.

- d) Angenommen, Sie würden aus der Matrix-Basisklasse eine Klasse `Mat3` für 3×3 Matrizen ableiten. Die Klasse enthielte folgende Funktionen:

```
class Mat3 : public Matrix
{
public:
    Mat3();
    Mat3( const Mat3& m );
};
```

```
Vec3 operator * ( const Mat3& m, const Vec3& v );
float det( const Mat3& m );
```

Die Wirkung der Funktionen sei wie folgt:

- `Vec3 operator * (const Mat3& m, const Vec3& m);` Eine normale Matrix-Vektor-Multiplikation wird durchgeführt.
- `float det(const Mat3& m);` Berechnet die Determinante der Matrix.

Wäre Ihre Implementierung der Klasse `Mat3` effizient? Begründen Sie ihre Antwort.