§1. Code Style

(1) Code Style ist Allman / BSD.

Beispiel:

```
int f(int x, int y, int z)
{
   if (x < foo(y, z))
   {
       haha = bar[4] + 5;
   }
   else
   {
       while (z)
       {
           haha += foo(z, z);
           z--;
       }
       return ++x + bar();
   }
}
```

(2) Man beachte, dass einzeilige Code-Blöcke ebenfalls geklammert werden!

(3) Der Rückgabewert einer Funktion steht in derselben Zeile wie der Methodenkopf.

(4) Zwischen mindestens Binären Operatoren und Variablen sind ausnahmslos Leerzeichen zu setzten. Bei unären müssen keine Leerzeichen gesetzt werden, es sei denn diese erleichtern die Lesbarkeit.

Ja:

```
int x = 5 + y;
```

Nein:

```
int x=5+y;
c ++
```

(5) Werden Blöcke von Variablen oder sehr ähnlichen Aufrufen geschrieben, sind Tabs so zu setzten, dass man die Struktur des Zugriffs gut erkennen kann.

Beispiel:

```
glVertex3f(v[a], v[a + 1], v[a + 2])
glVertex3f(v[b], v[b + 1], v[b + 2]);
glVertex3f(v[c], v[c + 1], v[c + 2]);
```

Das hilft, Fehler bei der Nummerierung der Indizes zu finden. Das "Gliedernde Element" (z.B. der Zuweisungsoperator wird dazu auf die minimale Tab-Position eingerückt, die benötigt wird, um eine saubere Ausrichtung des Blockes zu erreichen.

(6) Tabs sind 4-Stellen weit und werden durch Leerzeichen ersetzt.

(7) Die 80-Zeichen-Grenze wird bei uns auf "in etwa 100 Zeichen" angehoben. Umbrüche bei Funktionen mit langer Parameterliste erfolgen folgendermaßen:

```
return getIndexedArrayf(
            n, m_num_points,
            &m_points,
            &m_indexed_points );
```

(8) Alle Pointer werden bei der Deklaration mit `NULL` initialisiert und nachdem sie freigegeben wurden ebenfalls wieder genullt.

```
float* foo = NULL;
```

(9) Der Stern für den Pointer steht hinter dem Typ. Nicht vor dem Variablennamen (ich gebe zu das ist Geschmackssache). Auf gar keinen Fall wird der Typ mit dem Variablennamen multipliziert:

```
float * foo = NULL;
```

(10) Jede Variable wird in einer eigenen Zeile deklariert.

§2. Dateistruktur

(1) Alle Dateinamen beginnen mit einem Kleinbuchstaben.

(2) Header haben die Dateiendung `.h`. Die dazugehörigen Implementierungen haben die Endung `.c` und werden jeweils in ein eigenes Objektfile übersetzt.

(3) In der Regel wird Code mit verwandter Funktionalität in Modulen zusammengefasst, die jeweils aus einer `.h` und einer `.c`-Datei bestehen.