

Prozedurale Programmierung WS 2015/16

Projekt, Woche 1

Teil 1: Spezifikation

Teil 2: Design

- a) Programmstruktur
- b) Wichtigste Funktionen
- c) Zentrale Datentypen
- d) Gemeinsamer Codestil
- (i) Einrückung

Es werden Tabs mit einer Breite von 4 Leerzeichen benutzt.

(ii) Klammern

Wo möglich, werden geschweifte Klammern gesetzt (**if**, **else**, Funktionen, Strukturen, **typedefs**, etc.)

```
if ( x ) {  
  
}
```

Das **else**-Statement beginnt in derselben Zeile wie die vorhergehende schließende geschweifte Klammer.

```
if ( x ) {  
} else {  
}
```

Ausdrücke in Klammern werden mit Leerzeichen umgeben.

```
if ( x ) {  
  
}
```

statt

```
if (x) {  
  
}
```

und

```
x = ( y * 0.5f );  
  
statt  
x = (y * 0.5f);
```

Monday 4th January, 2016

(iii) Gleitkommaliterale

Immer die Genauigkeit eines Literals mit angeben, es sei denn, **double** wird explizit benötigt.

```
float f = 0.5f;
```

statt

```
float f = 0.5;
```

und

```
float f = 1.0f;
```

statt

```
float f = 1.f;
```

(iv) Funktionsnamen

Funktionen beginnen mit einem Großbuchstaben:

```
void Function( void );
```

In Funktionen, die mehrere Wörter enthalten, beginnt jedes Wort mit einem Großbuchstaben:

```
void ThisFunctionDoesSomething( void );
```

Die Standard-Kopfzeile für Funktionen ist:

```
/*  
=====
```

Funktionsname

Zweck

```
=====
```

**/*

(v) Variablen

Variablenamen beginnen mit einem Kleinbuchstaben.

```
float x;
```

In Variablenamen mit mehreren Wörtern beginnt jedes Wort bis auf das erste mit einem Großbuchstaben.

```
float maxDistanceFromPlane;
```

(vi) typedef

Namen von **typedefs** folgen derselben Konvention wie variablen, wobei ihnen ein **_t** an den Namen angehängt wird.

```
typedef int fileHandle_t;
```

Monday 4th January, 2016

(vii) **struct**-Namen

Namen von **structs** folgen derselben Konvention wie **typedefs**.

```
struct renderEntity_t;
```

(viii) **enum**

Namen von **enums** folgen ebenfalls derselben Konvention wie **typedefs** und **structs**. Die **enum**-Konstanten werden komplett in Großbuchstaben geschrieben. Mehrere Wörter werden untereinander durch Unterstriche getrennt.

```
enum contact_t {  
    CONTACT_NONE,  
    CONTACT_EDGE,  
    CONTACT_MODELVERTEX,  
    CONTACT_TRMVERTEX  
};
```

(ix) **Rekursive Funktionen**

Namen von rekursiven Funktionen enden in **_r**.

```
void WalkBSP_r( int node );
```

(x) **Makros**

Makros werden komplett in Großbuchstaben geschrieben. Wörter werden untereinander durch Unterstriche getrennt.

```
#define SIDE_FRONT 0
```

(xi) **const**

const soll wo immer möglich genutzt werden:

```
const int *p; // pointer to const int  
int * const p; // const pointer to int  
const int * const p; // const pointer to const int
```

Nicht benutzen:

```
int const *p;
```

(xii) **struct**

Die Standardkopfzeile für eine **struct** ist:

Monday 4th January, 2016

```
/*
```

```
=====
```

Beschreibung

```
=====
```

```
*/
```

Felder von **structs** folgen derselben Benennungskonvention wie Variablen. Felder sollten eingerückt sein, um ein tabellenähnliches Aussehen zu erreichen.

```
struct player_t {  
    char *      name;  
    texture2D_t texture;  
    float       position;  
    int         points;  
};
```

Das * eines Zeigers gehört zum Typ des Feldes und sollte daher aus Gründen der Lesbarkeit links bleiben.

(xiii) Dateinamen

Jede **struct** mit ihren dazugehörigen Funktionen sollte in einer eigenen Quellcodedatei sein, außer es ergibt Sinn, mehrere kleinere Structs in eine Datei zu gruppieren. Der Dateiname sollte derselbe sein wie der Name der **struct** mit einem Großbuchstaben am Anfang, aber ohne das **_t**. Die Dateien für **struct winding_t** wären dementsprechend **Winding.h** und **Winding.c**.

(xiv) Sprache

Der Code wird mit englischen Bezeichnern und Kommentaren geschrieben.