



**Fachhochschule  
Kaiserslautern**

University of  
Applied Sciences

- Projektarbeit -

## **Dancer**

**Interaktive Visualisierung  
Grafikprogrammierung**

von

**Dominik Marx, 855466  
Christian Schmidt, 855017**

16. Mai 2008

**Fachhochschule Kaiserslautern  
Fachbereich Informatik und Mikrosystemtechnik  
Studiengang „Digitale Medien“**

**Verantwortlicher Betreuer: Prof. Manfred Brill**

# Inhaltsverzeichnis

---

1. Einleitung .....	4
1.1. Um was geht es?.....	4
1.2. Exposé.....	4
2. Pflichtenheft.....	4
2.1. Zielbestimmung.....	4
<i>Muss-Kriterien:</i> .....	4
<i>Kann-Kriterien:</i> .....	4
2.2. Produkteinsatz .....	5
2.3. Produktübersicht.....	5
2.4. Produktfunktionen .....	5
2.5. Anforderung an die Abgabe.....	5
2.6. Anforderung an die Entwicklungsumgebung.....	5
3. Vorgehensweise und Realisierung .....	6
3.1. Die Idee .....	6
3.2. Das Konzept.....	6
3.3. Die Umsetzung.....	6
4. Freiheitsgrade des Modells .....	7
4.1. Grafische Ansicht – alles in Einem .....	7
4.2. Grafische Ansicht – drei Ansichten .....	8
4.3. Beschreibung der Freiheitsgrade.....	9
5. Tanzbewegungen und Sequenzen .....	10
5.1. Vorgehen für die Programmierung einer Tanzsequenz.....	10
5.2. Moonwalk / YMCA .....	11
6. Kür-Elemente .....	12
6.1. Weiterentwicklung der Freiheitsgrade .....	12
6.2. Mehrere Tänzer in der Szene .....	12
6.3. Mehrere Lichtquellen .....	13
6.4. Schatten .....	13

7. Das Software-Design .....	14
7.1. Designübersicht.....	14
7.2. Detailbeschreibung einzelner Komponenten.....	15
<i>Mehrere Tänzer</i> .....	15
<i>Licht</i> .....	15
<i>Schatten</i> .....	16
8. Interaktionsmöglichkeiten .....	17
8.1. Tastaturbelegung.....	17
8.2. Maustastenbelegung.....	20
9. Screen-Captures .....	21
10. Anhang .....	23
10.1. QUELLEN.....	23
10.2. Nutzungsrechte.....	23
10.3. Arbeitsverteilung.....	23
10.4. Ehrenwörtliche Erklärung.....	23
11. Notizen .....	24

# 1. Einleitung

---

## 1.1. Um was geht es?

Dieses Semesters soll als Aufgabe für das Wahlpflichtfach Grafikprogrammierung, bei Prof. Manfred Brill als Betreuer, ein bestehendes OpenGL-Programm, das mit Hilfe der vlgGraphicsEngine ein einfaches Modell eines Tänzers ausgibt, so erweitert werden, dass das Modell in der Anwendung interaktiv bewegt werden kann.

## 1.2. Exposé

Ziel des Projektes wird es sein, dass bei diesem „Tänzer“ komplette Tanzbewegungen abrufbar sind.

# 2. Pflichtenheft

---

## 2.1. Zielbestimmung

### *Muss-Kriterien:*

Fristgerechte Abgabe des Projektes am 19.Mai 2008, inklusive der dazugehörigen Dokumentation, Quellcode und einem ausführbarem Programm.

Desweiteren müssen folgende Aufgaben auf jeden Fall bearbeitet werden:

- Jedes Gelenk muss mindestens einen Freiheitsgrad haben
- Mindestens ein Gelenk muss zwei Freiheitsgrade aufweisen
- Arme und Beine dürfen den Oberkörper nicht berühren oder durchdringen
- Es muss mindestens eine vorprogrammierte Tanzbewegung erzeugt werden
- Die Bewegung der Arme und Beine und die globale Position des Tänzers ist durch die Tastatur oder die Maus interaktiv steuerbar
- Die vorprogrammierten Sequenzen müssen interaktiv abrufbar sein

### *Kann-Kriterien:*

- Mehrere interaktive Tänzer in der Szene
- Weiterentwicklung der Freiheitsgrade und der „Haut“
- Weiterentwicklung der Kinematik
- Bewegung des Tänzers mit Hilfe von GLSL-Shadern
- Bringen Sie dem Tänzer das Gehen oder das Heben von Gegenständen bei
- Schatten auf dem Tanzboden
- Cube Maps und Shadow Maps
- Mehr Lichtquellen; Kopplung von Lichtquellen und Tanzbewegungen
- Andere fortgeschrittene Themen in OpenGL

## 2.2. Produkteinsatz

### *Zielgruppe:*

Studenten der FH Zweibrücken, Studieninteressierte am Tag der offenen Tür, Professoren mit Interesse im Bereich der Grafikprogrammierung.

## 2.3. Produktübersicht

Im Wahlpflichtfach Grafikprogrammierung des SS08 wird als Abgabe ein ausführbares Programm entstehen.

## 2.4. Produktfunktionen

Die Visual Studio 2008 Projektdateien, sowie das kompilierte Programm, können zur Ansicht, oder zur beliebigen Weiterverarbeitung genutzt werden.

## 2.5. Anforderung an die Abgabe

Bei der Anforderung an die Abgabe haben wir uns das Ziel gesetzt, dass eine folgende Gruppe das Projekt ohne weitere Probleme weiter bearbeiten kann. Hierbei soll das grundlegende Erstellen anhand von Beispielen den Arbeitsablauf verdeutlichen. Die Abgabe selbst wird als gedruckte Version der Dokumentation mit CD/DVD abgegeben. Die CD/DVD enthält hierbei nochmals die Dokumentation in digitaler Form, sowie den im Rahmen der Projektarbeit entstandenen Quellcode und eine mit Doxygen erstellte C++ Dokumentation.

## 2.6. Anforderung an die Entwicklungsumgebung



Wir werden als Entwicklungsumgebung Visual Studio 2008 verwenden.



Zur Bildbearbeitung wird die Testversion von Adobe Photoshop CS3 zum Einsatz kommen.

## 3. Vorgehensweise und Realisierung

---

### 3.1. Die Idee

Da wir grundlegende Kenntnisse der Grafikprogrammierung im Fach Computergrafik und Computeranimation und Visualisierung sammeln konnten, werden wir versuchen, diese Kenntnisse Schritt für Schritt anzuwenden, um die gestellten Kriterien zu erfüllen. Im Zuge dessen, werden wir unserer Kreativität freien Lauf lassen.

### 3.2. Das Konzept

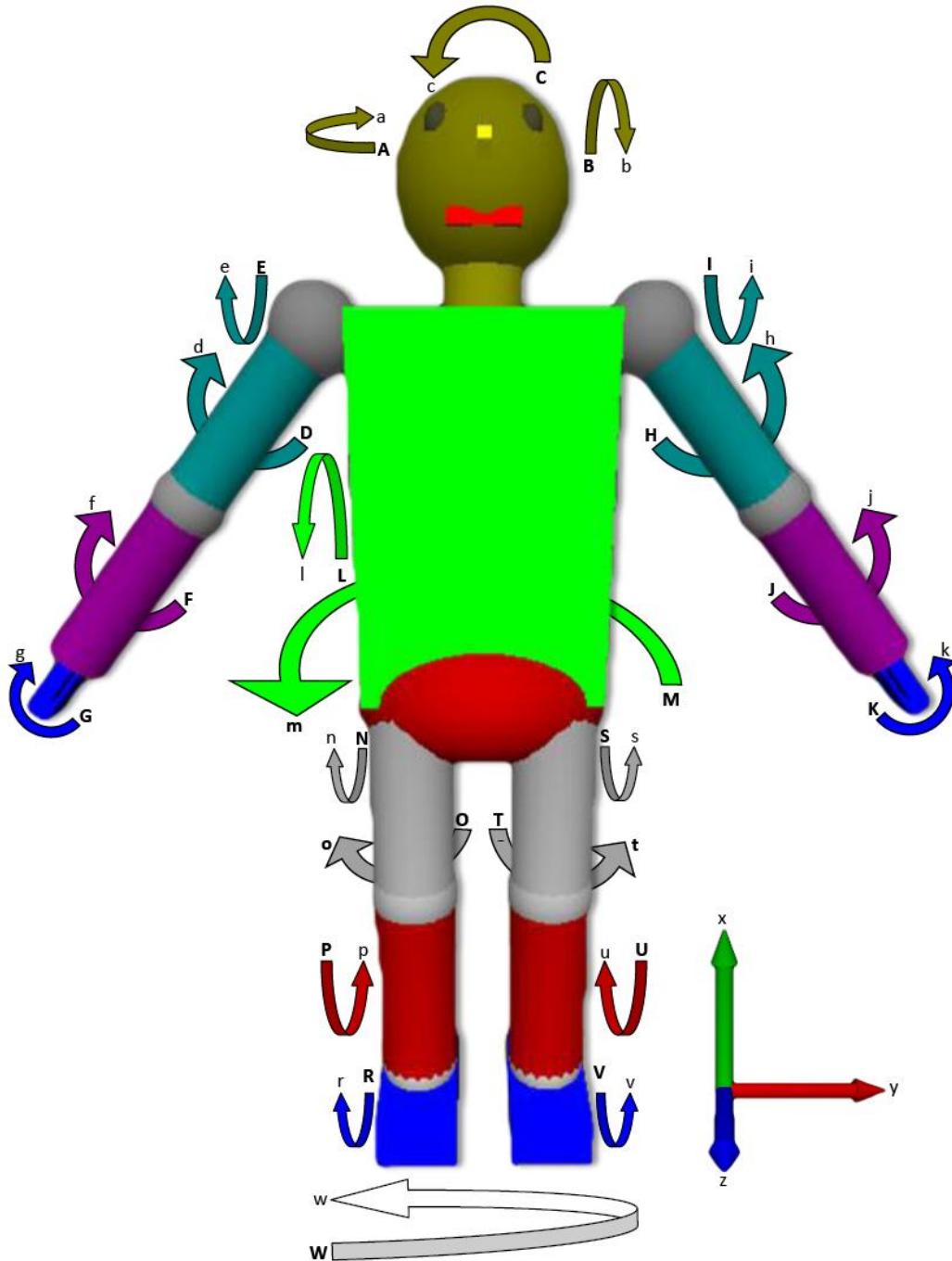
Da wir dem Tänzer das Tanzen beibringen sollen, werden wir uns zuerst der Aufgabe widmen, alle Gliedmaße und Körperteile in einem gewissen Rahmen frei zu bewegen. Wenn wir diese Hürde genommen haben, kann mit dem Programmieren der Tanzbewegungen begonnen werden, die interaktiv abrufbar sein sollen. Anschließend wenden wir uns den Extras zu, wie zum Beispiel das Hinzufügen mehrere Lichtquellen usw.

### 3.3. Die Umsetzung

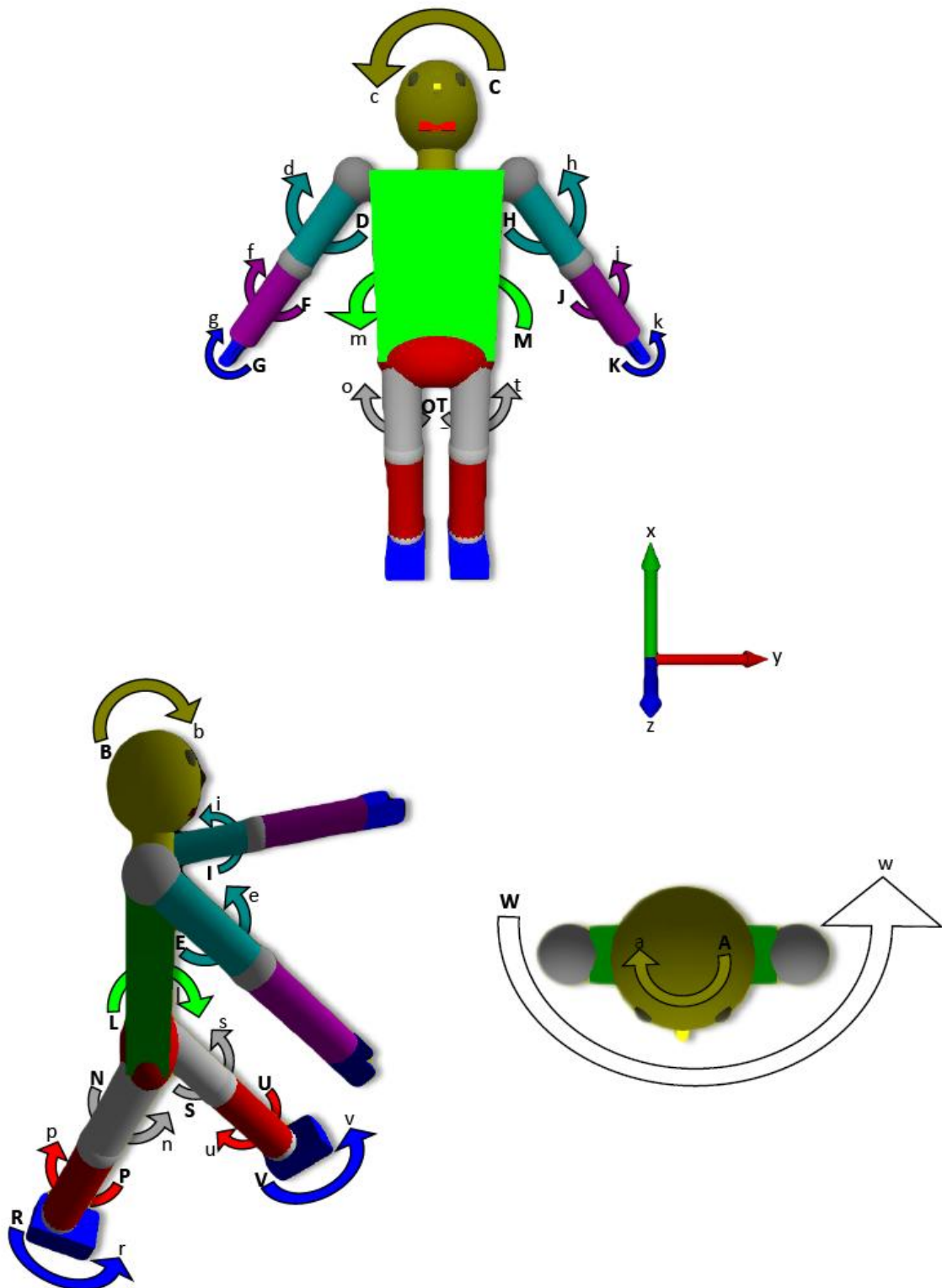
Nach einer Einarbeitungsphase in den bereits vorhandenen Quellcode, begannen wir den ursprünglichen Tänzer zu erweitern. Hierbei entschlossen wir uns zuerst dafür, dem Modell Hände, Füße, einen Hals und ein Gesicht, sowie alle weiteren Kugelgelenke einer vereinfachten menschlichen Statur zu geben. Orientiert an diesen nun neu vorhandenen Kugelgelenken, erweiterten wir den Tänzer um die geforderten Freiheitsgrade. Nachdem es uns dann möglich war, die einzelnen Komponenten des Tänzers gezielt zu steuern, programmierten wir schließlich noch eine Funktion, für die Tanzanimationen. Weitere Extras die von uns in das ursprüngliche Modell eingeführt wurden, sind zum Beispiel die Rotation des Lichtes während der Tanzbewegung.

# 4. Freiheitsgrade des Modells

## 4.1. Grafische Ansicht – alles in Einem



## 4.2. Grafische Ansicht – drei Ansichten



### 4.3. Beschreibung der Freiheitsgrade

Wie in obigen Grafiken gut zu erkennen ist, lässt sich der Tänzer mittels der Buchstaben auf der Tastatur steuern.

#### Kopf – 3 Freiheitsgrade

- a/A: drehen nach rechts und links
- b/B: nicken nach vorne und hinten
- c/C: schütteln nach rechts und links

#### Oberarm rechts – 2 Freiheitsgrade

- d/D: zur Seite hoch und runter
- e/E: nach vorne hoch und runter

#### Unterarm rechts – 1 Freiheitsgrad

- f/F: beugen hoch und runter

#### Hand rechts – 1 Freiheitsgrad

- g/G: hoch und runter

#### Oberarm links – 2 Freiheitsgrade

- h/H: zur Seite hoch und runter
- i/I: nach vorne hoch und runter

#### Unterarm links – 1 Freiheitsgrad

- j/J: beugen hoch und runter

#### Hand links – 1 Freiheitsgrad

- k/K: hoch und runter

#### Oberkörper – 2 Freiheitsgrade

- l/L: beugen vor und zurück
- m/M: kippen nach links und rechts

#### Oberschenkel rechts – 2 Freiheitsgrade

- n/N: bewegen zur Seite hoch und runter
- o/O: bewegen nach vorne hoch und runter

#### Unterschenkel rechts – 1 Freiheitsgrad

- p/P: beugen nach vorne und hinten

#### Fuß rechts – 1 Freiheitsgrad

- r/R: bewegen hoch und runter

#### Oberschenkel links – 2 Freiheitsgrade

- s/S: bewegen zur Seite hoch und runter
- t/T: bewegen nach vorne hoch und runter

Unterschenkel links – 1 Freiheitsgrad

u/U: beugen nach vorne und hinten

Fuß links – 1 Freiheitsgrad

v/V: bewegen hoch und runter

Dancer – 4 Freiheitsgrade ;-)

w/W: drehen um die y- Achse nach links und rechts

x/X: bewegen in positiver und negativer x-Richtung

y/Y: bewegen in positiver und negativer y-Richtung

z/Z: bewegen in positiver und negativer z-Richtung

## 5. Tanzbewegungen und Sequenzen

---

### 5.1. Vorgehen für die Programmierung einer Tanzsequenz

Möchte man eine Tanzsequenz erstellen, bewegt man den Tänzer in die gewünschte Zielposition, der ersten Tanzstellung und drückt einmal die Raute(#)-Taste. Das Drücken der Raute(#)-Taste bewirkt ein Speichern der aktuellen Parameter in die Datei

**dance\_positions[0-9].txt**

im Projektverzeichnis.

Als nächsten Schritt kopiert man den Inhalt der Datei in das Array danceInformations(4-9) der Funktion `DancerEngine::dance0-9()` der Klasse `DancerEngine.cpp`.

Hier ein kurzes Beispiel:

```
void DancerEngine::dance2()
{
    GLfloat danceInformations2[26*5] = {
        //1-Y
        0, 0.125, 0, 0, 180, 0, 39, 16, 35, -56, 316, 10, -38, 27, 1, -1, 180, 0, 180, 180, 180, 0, 180, 180, 0, 20,
        //2-M
        0, 0.075, 0, 0, 180, 0, -20, 180, 1, -56, 380, 179, -1, 28, 0, -1, 178, 9, 180, 177, 180, -9, 180, 180, 0, 20,
        //3-C
        0, 0.075, 0, 0, 180, 0, 0, 2, 66, -56, 180, 6, -54, 54, 0, -1, 178, 0, 180, 177, 180, 0, 180, 180, 0, 20,
        //4-A
        0, 0.075, 0, 0, 180, 0, -17, 3, 0, -56, 199, 180, 0, -44, 1, -1, 178, 0, 180, 177, 180, 0, 180, 180, 0, 20,
        //5-Ausgangsposition
        0, 0.125, 0, 0, 180, 0, 180, 0, 0, 0, 180, 0, 0, 0, 0, 0, 180, 0, 180, 180, 180, 0, 180, 180, 0, 5
    };
    dancer1->letsDance(danceInformations2, (sizeof(danceInformations2)/sizeof(danceInformations2[0])),2);
}
```

Das Array ist wie folgt aufgebaut:

- GLfloat danceInformations[26 \* x] – 26 Parameter pro Tanzstellung mal x Tanzstellungen. In obigem Beispiel gehen wir von 5 Tanzstellungen aus.
- Die einzelnen Parameter bedeuten von rechts nach links:
  - 00 – Globale Position des Tänzers in x-Richtung
  - 01 – Globale Position des Tänzers in y-Richtung
  - 02 – Globale Position des Tänzers in z-Richtung
  - 03 – Rotation des Kopfes um die x-Achse
  - 04 – Rotation des Kopfes um die y-Achse
  - 05 – Rotation des Kopfes um die z-Achse
  - 06 – Rotation des rechten Armes zur Seite
  - 07 – Rotation des rechten Armes nach vorne
  - 08 – Rotation des rechten Unterarmes
  - 09 – Rotation der rechten Hand
  - 10 – Rotation des linken Armes zur Seite
  - 11 – Rotation des linken Armes nach vorne
  - 12 – Rotation des linken Unterarmes
  - 13 – Rotation der linken Hand
  - 14 – Beugen des Oberkörpers nach vorne und hinten
  - 15 – Beugen des Oberkörpers zur Seite
  - 16 – Rotation des rechten Beines nach vorne
  - 17 – Rotation des rechten Beines zur Seite
  - 18 – Rotation des rechten Unterschenkels
  - 19 – Rotation des rechten Fußes
  - 20 – Rotation des linken Beines nach vorne
  - 21 – Rotation des linken Beines zur Seite
  - 22 – Rotation des linken Unterschenkels
  - 23 – Rotation des linken Fußes
  - 24 – Rotation des ganzen Tänzers um die eigene Achse
  - 25 – Geschwindigkeit der ausgeführten Bewegung.
- Danach wird der Funktion letsDance das Array, die Größe des Arrays und die Nummer des aktuellen Tanzes übergeben.

## 5.2. Moonwalk / YMCA

Als Tanzsequenzen haben wir uns an den Vorlagen des Moonwalk von Michael Jackson und dem YMCA Tanz der Village People orientiert.

## 6. Kür-Elemente

---

Als Elemente für die Kür haben wir uns für folgende Punkte entschieden:

### 6.1. Weiterentwicklung der Freiheitsgrade

Die Weiterentwicklung der Freiheitsgrade wurde von uns als erster Schritt in Richtung Kür vollführt. Hierbei haben wir die aus den Pflichtaufgaben mit einem Freiheitsgrad in jedem Gelenk und zwei Freiheitsgraden in mindestens einem Gelenk geforderte Anzahl insofern erfüllt, dass es bei unserer Ausgabe des Tänzers 3 Freiheitsgrade am Kopf und jeweils 2 Freiheitsgrade an den Gelenken der Oberarme, der Oberschenkel und des Rumpfes gibt.

Genauere Angaben zu den jeweiligen Freiheitsgraden und der Steuerung über die Tastatur findet man im Kapitel 4 dieser Dokumentation.

### 6.2. Mehrere Tänzer in der Szene

Das Einfügen mehrerer Tänzer ist mittels der ‘ ‘ ‘ (Anführungszeichen) Taste möglich. Das synchrone Ausführen einer Tanzbewegung dieser einzelnen Tänzer ist möglich, wenn man den Tanz „YMCA – Gruppe“ aufruft. Bei diesem Tanz werden auch automatisch alle Tänzer aktiviert.



Tastaturbelegung: ‚3‘ gedrückt halten

Das Abschalten der weiteren Tänzer ist möglich, indem man die Taste ‘ ‘ ‘ (Anführungszeichen) drückt.

### **6.3. Mehrere Lichtquellen**

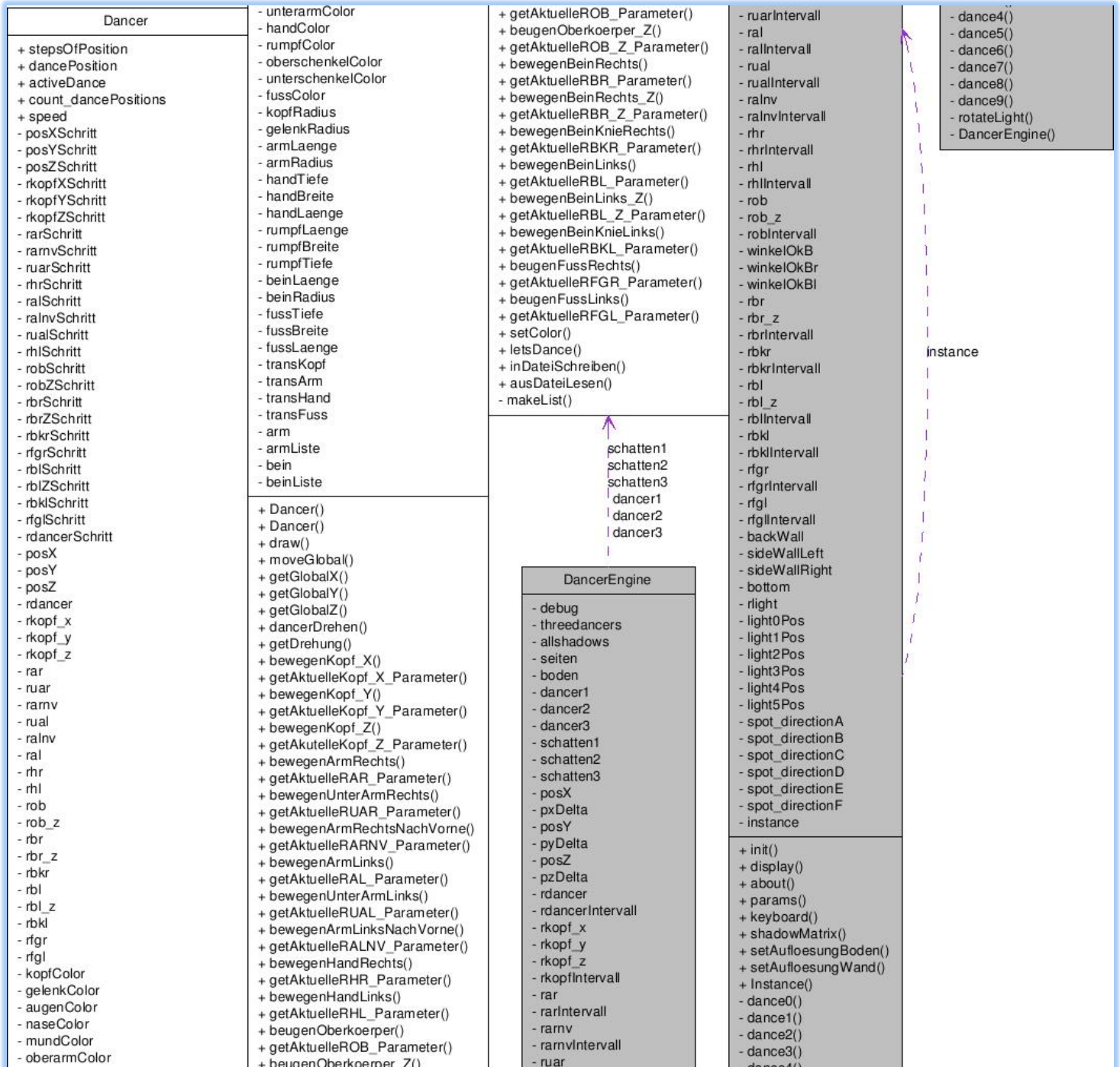
Es wurden zusätzlich fünf Lichtquellen in die Szene gesetzt, wovon vier kreisförmig um und eine mittig über dem mittleren Tänzer angeordnet wurde. Ein Rotieren der äußeren Lichter beim Ausführen der Tanzsequenzen, so wie beim drücken der ‚!‘ – Taste wurde zusätzlich mit integriert.

### **6.4. Schatten**

Der Tänzer wirft durch verschiedene Lichtquellen, mehrere Schatten auf den Boden, die Seitenwände und die Rückwand. Informationen zur Implementierung findet man im Kapitel 7.2 „Schatten“.

# 7. Das Software-Design

## 7.1. Designübersicht



## 7.2. Detailbeschreibung einzelner Komponenten

### Mehrere Tänzer

Um weitere Tänzer zu erzeugen, initialisiert man zuerst in der `DancerEngine::init` eine neue Instanz „Dancer“, mit den xyz-Werten der Position, an der der neue Tänzer stehen soll.

Bsp.:

```
dancer1 = new Dancer(posX, posY, posZ);
dancer2 = new Dancer(posX+2.6, posY, posZ);
dancer3 = new Dancer(posX-2.6, posY, posZ);
```

In der `DancerEngine::display` werden die einzelnen Tänzer mit der Funktion `draw()` gezeichnet.

Bsp.:

```
dancer1->draw();
dancer2->draw();
dancer3->draw();
```

### Licht

Weitere Lichtquellen initialisiert man zuerst in der `DancerEngine::init` mit folgenden Codezeilen:

```
//Lichtquelle0
glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
glLightfv(GL_LIGHT0, GL_SPECULAR, specularLight);
glEnable(GL_LIGHT0);

//Lichtquelle0 Durchmesser
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0f);
//Lichtquelle0 Intensität
glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 20.0f);

//Lichtquelle1
glLightfv(GL_LIGHT1, GL_AMBIENT, ambientLight);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diffuseLight);
glLightfv(GL_LIGHT1, GL_SPECULAR, specularLight);
glEnable(GL_LIGHT1);

//Lichtquelle1 Durchmesser
glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0f);
//Lichtquelle1 Intensität
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 20.0f);
...
```

In der `gl.h` sind 8 Lichtquellen vordefiniert (`GL_LIGHT0` - `GL_LIGHT7`).

Danach werden die Lichter in der `DancerEngine::display` an ihre Positionen gesetzt.

```
light0Pos[0]=0.0f;
light0Pos[1]=5.0f;
light0Pos[2]=6.5f;
light0Pos[3]=1.0f;
```

Der Lichtkegel wird danach mit Spot Direction in die entsprechend gewünschte Richtung gedreht.

```

spot_directionA[0] = 0.0f;
spot_directionA[1] = -0.807f;
spot_directionA[2] = -0.807f;

glLightfv(GL_LIGHT0, GL_POSITION, light0Pos);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spot_directionA);

glLightfv(GL_LIGHT1, GL_POSITION, light1Pos);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_directionB);
...

```

Parameter-Erläuterung:

Eigenschaft	Standard Wert	Bedeutung
GL_AMBIENT	0.0, 0.0, 0.0, 1.0	Ambienter RGBA Anteil des Lichts
GL_DIFFUSE	1.0, 1.0, 1.0, 1.0	Diffuser RGBA Anteil des Lichts
GL_SPECULAR	1.0, 1.0, 1.0, 1.0	Glanz RGBA Anteil des Lichts
GL_POSITION	0.0, 0.0, 1.0, 0.0	Koordinaten des Lichts
GL_SPOT_DIRECTION	0.0, 0.0, -1.0	Richtung des Spotlights
GL_SPOT_EXPONENT	0.0	Spotlight-Exponent
GL_SPOT_CUTOFF	180.0	Spotlight Sperrwinkel

### Schatten

Zuerst wird in der `DancerEngine::init` eine Instanz „schatten1“ von „Dancer“ initialisiert. Dieser Schatten sollte natürlich die gleiche Startposition haben, wie der Tänzer, der ihn wirft.

Bsp.:

```

schatten1 = new Dancer(posX, posY, posZ);

```

Danach wird festgelegt, auf welche Ebene der Schatten projiziert wird. Dazu benötigt man einen Punkt und eine Normale der Ebene.

Bsp.(hier XZ-Ebene):

```

GLfloat planeNormal[3] = {0.0, 1.0, 0.0};
GLfloat planeP[3] = {0.0, 0.0, 0.0};

```

Des Weiteren benötigt man für jeden geworfenen Schatten eine 4x4-Matrix für die jeweilige Berechnung.

Bsp.:

```

GLfloat shadow0[16];
GLfloat shadow1[16];

```

Jetzt kann die Projektionsmatrix der jeweiligen Schatten bestimmt werden, indem man der Funktion „shadowmatrix“, die 4x4 Matrix, die Normale der Projektionsebene, den Punkt der Projektionsebene und die Position des zutreffenden Lichtes übergibt.

Bsp.:

```

shadowMatrix(shadow0, planeNormal, planeP, light0Pos);

```

Danach muss das Offset eingeschaltet werden, um Artefakte im z-Buffer zu vermeiden. Mehr zu dieser Funktion findet man im red book auf Seite 247 ff.

Bsp.:

```

glEnable(GL_POLYGON_OFFSET_FILL);

```

```
glPolygonOffset(-4.0, -0.002);
```

Letztendlich wird der Schatten gezeichnet. Damit der Schatten nicht die gleichen Farbwerte erhält, wie der Tänzer selbst, müssen die Farbwerte auf schwarz gesetzt werden.

Bsp.:

```
glPushMatrix();
    glMultMatrixf(shadow0);
    glTranslatef(0.0, 0.1, 0.0);
    schatten->draw();
    schatten->setColor(0.0f, 0.0f, 0.0f);
glPopMatrix();
```

Alle Transformationen, die auf den Tänzer selbst angewendet werden, müssen natürlich auch auf seinen Schatten angewendet werden.

## 8. Interaktionsmöglichkeiten

---

### 8.1. Tastaturbelegung

-----  
Tastaturkommandos  
-----

F1 :	Hilfemenü	
F2 :	-	
F3 :	-	
F4 :	-	
F5 :	-	
F6 :	Flat-/Gouraud-Shading	an/aus
F7 :	Weltkoordinatensystem	an/aus
F8 :	-	
F9 :	-	
F10:	Reset der Projektionseinstellung	
F11:	Reset der Kamerasteuerung	
F12:	Fullscreen	an/aus

-----  
Cursor-Tasten  
-----

Ps1: vergrößern des Öffnungswinkels  
Hme: verkleinern des Öffnungswinkels

-----  
Cursor-Tasten – Examin  
-----

<-: Den Azimuth um 1 vergrößern  
->: Den Azimuth um 1 verkleinern  
up: Die Elevation um 1 vergrößern  
dwn: Die Elevation um 1 verkleinern  
Pup: Betrachterabstand verkleinern  
Pdn: Betrachterabstand vergrößern

-----  
Cursor-Tasten - Pilots View  
-----

<-: Den Gierungswinkel um 1 vergrößern  
->: Den Gierungswinkel um 1 verkleinern  
up: Vorwärtsfahren  
dwn: Rückwärtsfahren  
Pup: schneller fahren  
Pdn: langsamer fahren

-----  
Programm  
-----

ESC: Programm beenden  
q/Q: Programm beenden

-----  
Tanzbewegungen  
-----

0: Ausgangsfigur / Startposition / Licht Reset / Tänzer Reset  
1: Tanzbewegung A - Moonwalk  
2: Tanzbewegung B - YMCA einfach  
3: Tanzbewegung C - YMCA Gruppe  
4: -  
5: -  
6: -  
7: -  
8: -  
9: Tanzbewegung, die direkt aus den in die Datei geschriebenen Parametern  
ausgeführt wird. (TODO)

-----  
Licht  
-----

!: Rotation der äußeren Lichter

-----  
Tänzer  
-----

" : Äußere Tänzer an/aus

-----  
Parameter Ein-/Ausgaben  
-----

? : Ausgabe aller aktuellen Parameter  
 \*/+ : Ausgabe der jeweils aktuellen Parameter an/aus  
 # : Ausgabe der jeweils aktuellen Parameter in dance\_positions.txt  
 @ : Einlesen der jeweils aktuellen Parameter aus dance\_positions.txt

-----  
Kopf  
-----

a/A : Bewegung des Kopfes - drehen rechts/links  
 b/B : Bewegung des Kopfes - nicken vor/zurück  
 c/C : Bewegung des Kopfes - schütteln rechts/links

-----  
Arm rechts  
-----

d/D : Bewegen des rechten Armes - zur Seite hoch/runter  
 e/E : Bewegen des rechten Armes - nach vorne hoch/runter  
 f/F : Beugen des rechten Unterarmes hoch/runter  
 g/G : Bewegen der rechten Hand hoch/runter

-----  
Arm links  
-----

h/H : Bewegen des linken Armes - zur Seite hoch/runter  
 i/I : Bewegen des linken Armes - nach vorne hoch/runter  
 j/J : Beugen des linken Unterarmes hoch/runter  
 k/K : Bewegen der linken Hand hoch/runter

-----  
Oberkörper  
-----

l/L : Beugen des Oberkörpers vor/zurück  
 m/M : Kippen des Oberkörpers rechts/links

-----  
Bein rechts  
-----

n/N: Bewegen des rechten Beines - zur Seite	hoch/runter
o/O: Bewegen des rechten Beines - nach vorne	hoch/runter
p/P: Beugen des rechten Unterschenkels	zurück/vor
r/R: Bewegen des rechten Fußes	hoch/runter

-----  
Bein links  
-----

s/S: Bewegen des linken Beines - zur Seite	hoch/runter
t/T: Bewegen des linken Beines - nach vorne	hoch/runter
u/U: Beugen des linken Unterschenkels	zurück/vor
v/V: Bewegen des linken Fußes	hoch/runter

-----  
Dancer komplett  
-----

w/W: Drehen um die Y-Achse	links/rechts
x/X: Bewegung in X-Richtung	positiv/negativ
y/Y: Bewegung in Y-Richtung	positiv/negativ
z/Z: Bewegung in Z-Richtung	positiv/negativ

## 8.2. Maustastenbelegung

-----  
Maustasten – Examin  
-----

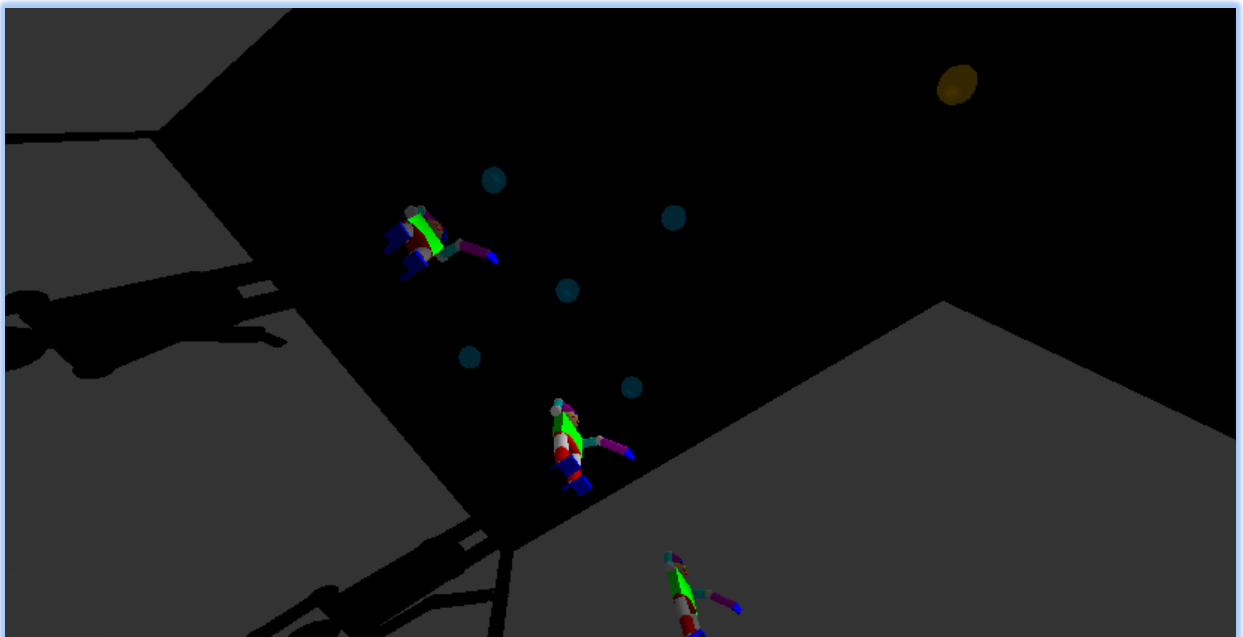
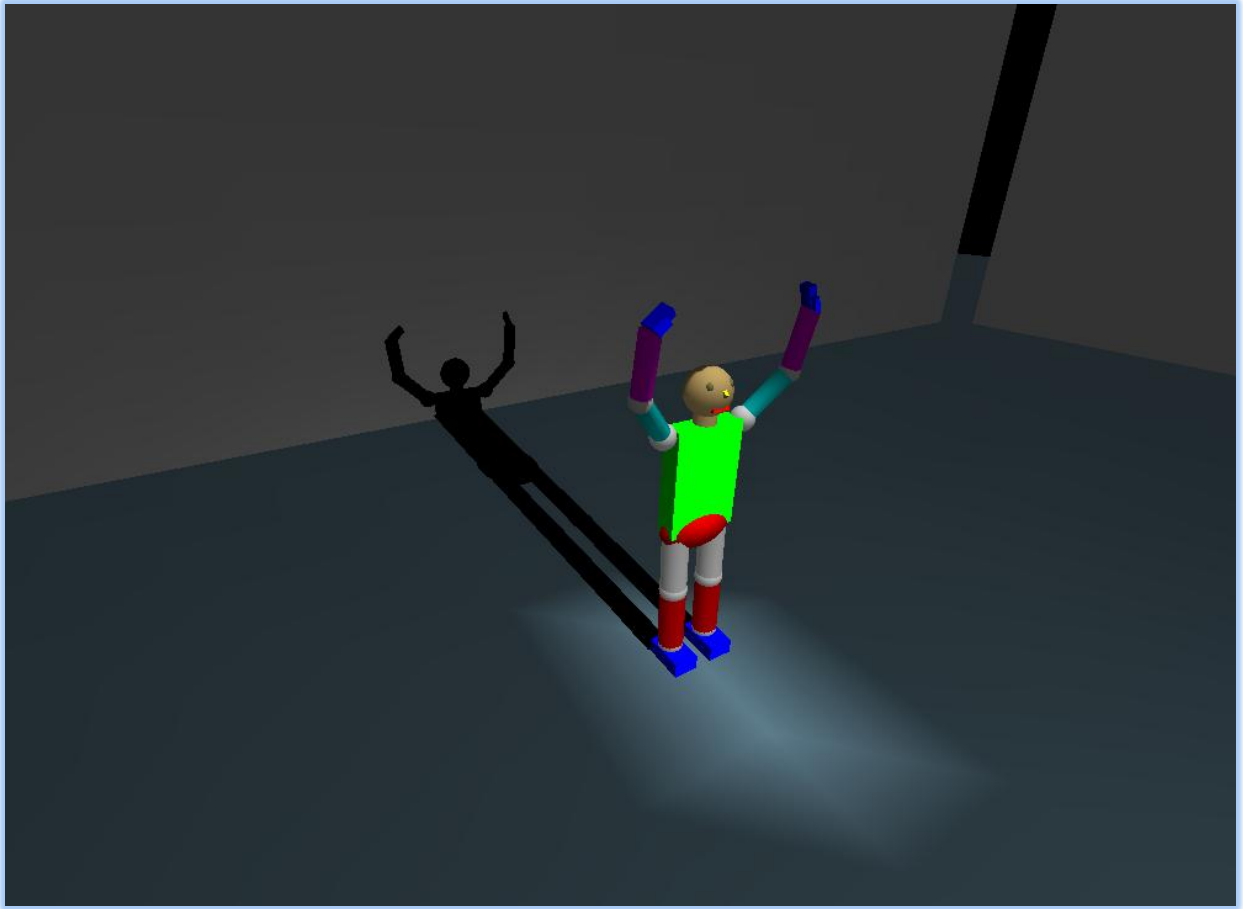
Links: Trackball  
Mitte: Pan  
Rechts: Zoom

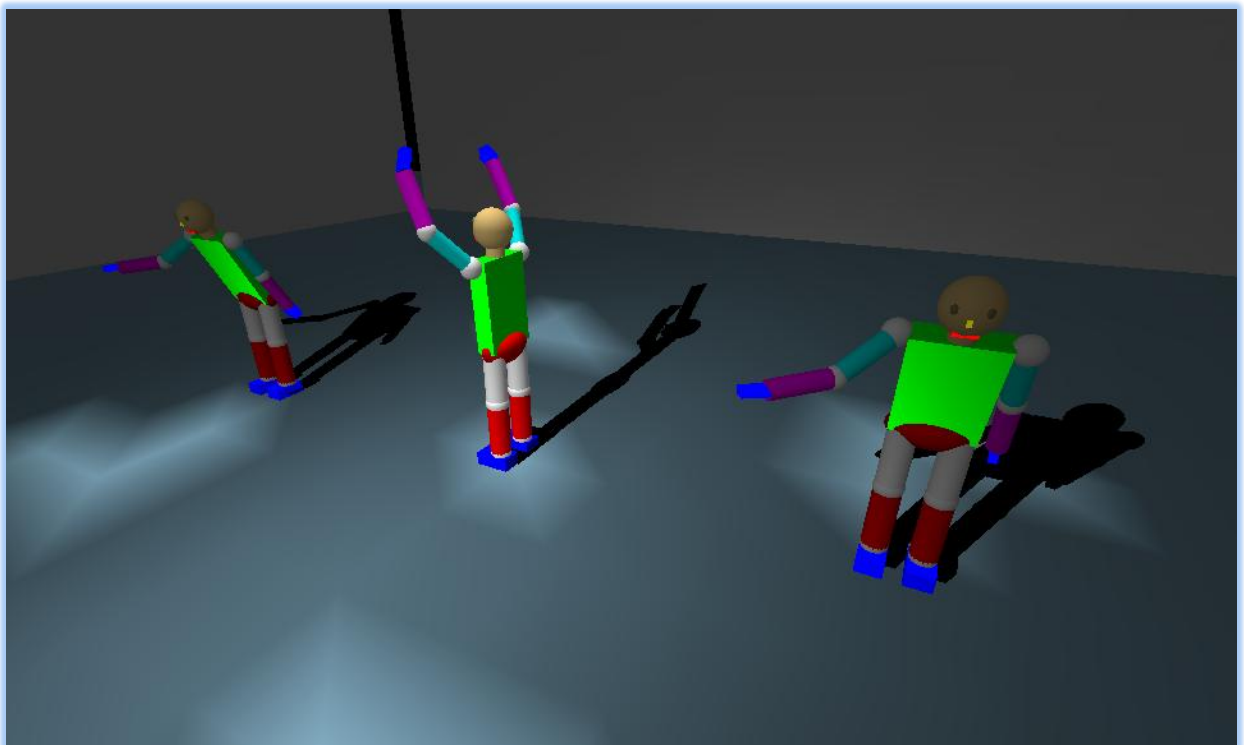
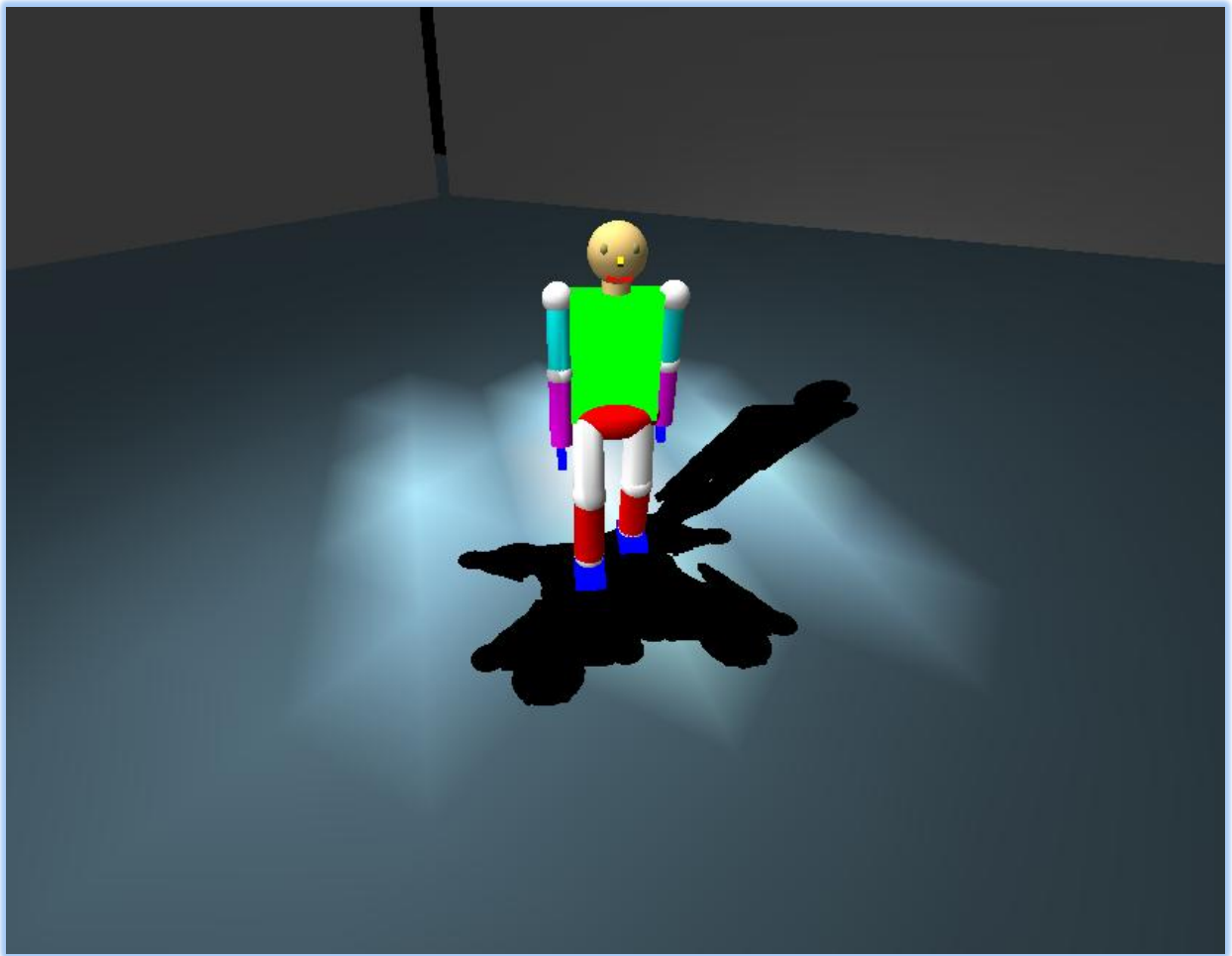
-----  
Maustasten - Pilots View  
-----

Links: Gierung/Neigungswinkel  
Mitte: Rollwinkel  
Rechts: Fahren

## 9. Screen-Captures

---





# 10. Anhang

---

## 10.1. QUELLEN

Aufzeichnungen zur Computergrafik Vorlesung aus dem Wintersemester 2006/2007 der Fachhochschule Kaiserslautern, Standort Zweibrücken.

Zur Erstellung des UML Diagramms auf Seite 14 und die HTML Dokumentation auf CD/DVD und im Internet unter: <http://dancer.christian-michael-schmidt.de/>

Doxygen: <http://www.stack.nl/>

GraphVis: <http://www.graphviz.org/>

## 10.2. Nutzungsrechte

Das alleinige Nutzungsrecht liegt bei den Studenten Dominik Marx, 855466 und Christian Schmidt, 855017.

Weiterhin dürfen die Modelle im Rahmen der Lehrveranstaltungen an der Fachhochschule Zweibrücken an die Studenten weitergegeben und an öffentlichen Veranstaltungen, wie zum Beispiel im Rahmen des Tages der offenen Tür, gezeigt werden.

## 10.3. Arbeitsverteilung

Hiermit erklären wir, **Christian Schmidt** und **Dominik Marx**, dass wir beide zu gleichen Teilen am Projekt gearbeitet haben, womit keine getrennte Auflistung der Arbeitsstunden erforderlich ist.

## 10.4. Ehrenwörtliche Erklärung

Hiermit erkläre wir, **Dominik Marx**, geboren am 25.09.1983 in Saarbrücken und **Christian Schmidt**, geboren am 06.09.1983 in Bad Kreuznach ehrenwörtlich, dass wir unsere Projektarbeit im Wahlpflichtfach Grafikprogrammierung mit dem Titel: „**Dancer**“ selbstständig und ohne fremde Hilfe angefertigt haben und keine anderen als in der Abhandlung angegebenen Hilfen benutzt haben; dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Saarbrücken, den 16.05.2008  
[**Dominik Marx**]

Zweibrücken, den 16.05.2008  
[**Christian Schmidt**]

# 11. Notizen

---

Hier finden Sie Platz für ihre Notizen.