AniTMT
Users Guide

Dokumentation zu AniTMT

Bedienung der Einzelprogramme
und
Referenz zur Animation Description Language

1. September 2000

Inhaltsverzeichnis

Teil I.
Einleitung

Filme wie Toy Story und A Bug's Life haben uns eindrucksvoll gezeigt, welche Qualität mit vollständig computer-generierten Filmen inzwischen erreicht werden kann. Die Erstellung solcher Filme – wenn auch in wesentlich bescheidenerer Form – ist für uns ein großer Traum, seit wir uns mit Raytracing beschäftigen.

Für unsere ersten Versuche haben wir den Raytracer POV-Ray verwendet. Er hat uns im Bereich der Standbilder überzeugt, doch die darin integrierte Animationsfähigkeit ist in der Anwendung zu umständlich. Allerdings haben wir auch keine andere Software gefunden, die eine angemessene Lösung bietet.

Aus diesem Grund haben wir uns dazu entschlossen, ein eigenes Animationssystem zu entwickeln. Als Name hierfür haben wir AniTMT gewählt, wobei TMT für die Anfangsbuchstaben unserer Nachnamen steht (Theofel, Moser, Trautmann).

Teil II.
Was ist AniTMT ?

AniTMT ist ein Programmpaket, das zu Erstellen von Computeranimationen mit bis zu fotorealistischer Qualität dienen soll. Das Programm dient als Aufsatz für einen Raytracer, insbesondere POV-Ray der die Einzelbilder, des Filmes berechnet. Besondere Merkmale des Programms sind die flexiblen Möglichkeiten eine Animation zu definieren und eine Verteilung der Rechenlast über ein Netzwerk.

Das Programm befindet sich derzeit noch in der Entwicklung. Fehler in der Funktionsweise sind leider nicht ausgeschlossen. Wir sind jedoch für Fehlermeldungen, Verbesserungsvorschläge und andere Rückmeldungen aufgeschlossen. Wenn Interesse besteht, kann auch direkt an der Entwicklung mitgedacht und gearbeitet werden.
Das Paket besteht aus folgenden Einzelprogrammen:

- `anitmt-calc`
  berechnet die Szenendateien für die Einzelbilder, aus der Orginalszene und einer Animationsbeschreibung.

- `anitmt-server`
  verteilt die einzelnen Szenenbeschreibung an unterschiedliche Rechner in einem Netzwerk.

- `anitmt-client`
  ist das Gegenstück zu `anitmt-server`, das die Szenenbeschreibungen entgegennimmt, das Einzelbild berechnen und das Ergebnis zurückschickt.

- `anitmt-view`
  berechnet die Animation auf einem einzelnen Computer, zu um Previews zu erstellen oder falls kein Netzwerk zur Verfügung steht.

- `tga2avi`
  wandelt die tga-Datein in ein unkomprimiertes avi um.

- `mkavi`
  wandelt die tga-Datein in ein cram16 kompremiertes avi um.

Eines unserer Ziele ist es, das AniTMT auf jedem Computer läuft. `anitmt-calc` und `anitmt-view` sind in C++ geschrieben und laufen daher theoretisch auf jedem Betriebsytem auf dem ein C++-Compiler zur Verfügung steht. Getestet wurde es bereits unter Linux, BeOS und einigen Windows-Versionen.

Die Netzwerkverteilung ist mit Perl realisiert. Da `anitmt-server` den System-V-Standard benutzt, scheiden hier leider einige Betriebssyteme als Serversystem aus. Auch der Client läuft bislang nur unter Linux, an der Portierung für andere Systeme wird jedoch gearbeitet.

Teil III.
Übersicht über das Erstellen einer Animation

š

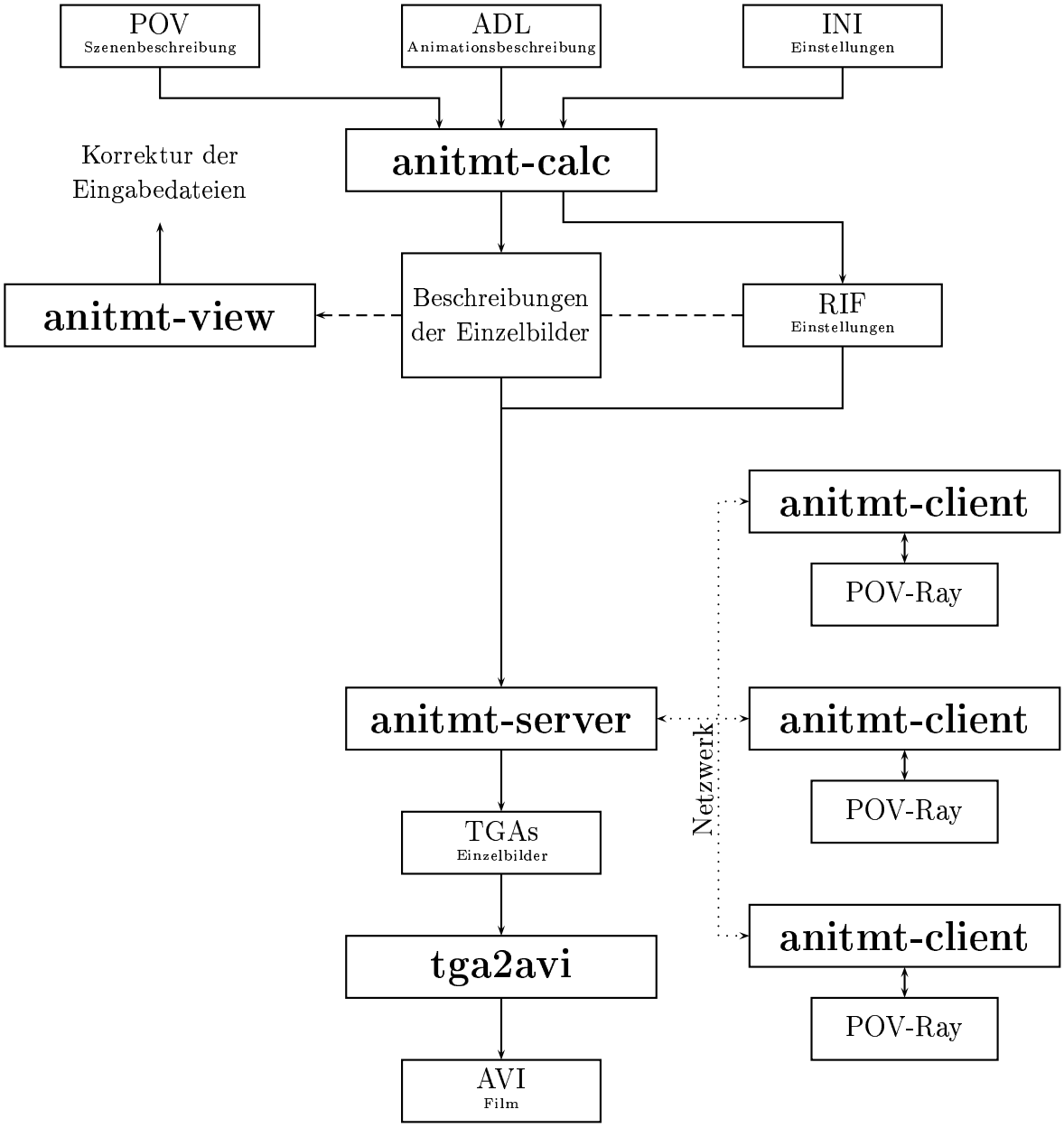

Abbildung 1: Übersicht zum Programmsystem AniTMT

Wenn mit AniTMT eine Animation erstellt wird, verläuft dies nach dem folgenden Prinzip. Als erstes erstellt man nach wie vor die **Szene für seinen Raytracer**. Die Objekte, die später animiert werden sollen, werden mit einer Kommentar-Syntax benannt. Zusätzlich erstellt man für AniTMT eine oder mehrere **ADL-Dateien** (ADL = animation description language). In dieser wird dann für die Objekte eine Bewegung festgelegt. Auch Veränderungen für Variablen können dort definiert werden.

Eine Besonderheit, die uns von vielen anderen Lösungen zur Animation abhebt, ist dabei die Möglichkeit, die Bewegung völlig unterschiedlich definieren. Der Anwender kann selbst entscheiden, mit welche Eigenschaften er seine Flugbahn definiert. Man kann beispielsweise Positionen, Streckenlängen, Zeitpunkte, die Dauer, Geschwindigkeiten oder Beschleunigungen angeben. Hauptsache ist, dass die Animation eindeutig bestimmt wird. Wenn eine Bewegung oder Veränderung durch mehrere Teilstücke zusammengesetzt wird, werden natürlich bei diesen automatisch Positionen, Richtungen und Geschwindigkeiten miteinander gekoppelt. Doch auch Sprünge sind möglich, wenn diese erwünscht sind.

Als drittes benötigt man noch eine **Initialisierungs-Datei** (INI), in der die grundlegenden Einstellungen der Animation, wie Dauer, Auflösung, oder die Anzahl der Bilder pro Sekunde festgelegt werden. Diese INI-Datei, die auch die Informationen über die dazugehörigen ADL-Dateien enthält, wird an das Programm `anitmt-calc` übergeben. `anitmt-calc` geht zunächst die ADL-Datei durch und erstellt aus den Definitionen den Ablauf der Animation. Anschließend wird in die Szenenbeschreibung Positionierungs- und Drehbefehle eingefügt. Für jedes zu berechnende Einzelbild die Positionen und Werte ausgerechnet und in einer Serie von Include-Dateien gespeichert. Der Raytracer kann nun Einzelbilder berechnen, indem ihm die Szenenbeschreibung zusammen mit einer der Include-Dateien übergeben wird.

Hierzu gibt es je nach Gegebenheit zwei Programme, die dies automatisch erledigen. Wer über ein Netzwerk verfügt oder rechenaufwendigere Filme berechnen will, sollte hierzu `anitmt-server` benutzen. `anitmt-server` nimmt das Paket von erstellten Szenenbeschreibungen an und startet ein Projekt. Für dieses Projekt können sich dann mit Hilfe eines eigenen Netzwerkprotokolles, das auf TCP/IP aufbaut, Rechner mit dem Programm `anitmt-client` anmelden. `anitmt-client` bekommt dann die Szenenbeschreibung eines Einzelbildes übermittelt und berechnet dann mit dem Raytracers, der auf dem jeweiligen Rechner installiert sein muss, das Einzelbild im TGA-Format. Das Ergebnis wird an den Server zurückgeschickt und der nächste Auftrag angefordert. Der Server bekommt somit eine Sammlung von Einzelbildern geliefert.

Wenn kein Netzwerk zur Verfügung steht oder nur eine Vorschau berechnet werden soll, gibt es auch das Programm `anitmt-view`. Diese Programm erstellt die Einzelbilder auf einem einzelnen Rechner, indem es den Raytraycer nacheinander mit den berechneten Szenebeschreibungen aufruft.

Damit nachher aus den Einzelbildern ein Film wird, kann man eines unser mitgelieften Tools verwenden. `tga2avi` erstellt unkomprimierte Filme. `mkavi` erstellt ein avi mit cram16 Kompression. Es gibt jedoch sicherlich bessere Programme, um TGA-Bilder in Filme zu verwandeln. Insbesondere, wenn auf gute Kompression wert gelegt wird.

Teil IV. Bedienung der einzelnen Komponenten

1. Bedienung von `anitmt-calc`

1.1. Funktionsweise

`anitmt-calc` bekommt beim Start vom Benutzer gesagt, welche ADL-Dateien zu bearbeiten sind. Dies geschieht entweder über eine INI-Datei (siehe: VIII Die INI-Datei), oder über Parameter. Danach geht das Programm folgendermaßen vor: Als erstes werden die ADL-Dateien eingelesen und die Objekte und Definitionen daraus entnommen.Da die Informationen in der Regel noch nicht vollständig sind, werden dann die restlichen Eigenschaften der Animation ergänzt, indem sie aus anderen Eigenschaften oder Standardwerten ermittelt werden, oder an Nachbarelementen angeglichen werden. Wenn schließlich der Ablauf der Animation eindeutig bestimmt ist, wird für jedes zu berechnende Einzelbild die Positionen und Werte berechnet. Diese werden dann entsprechend in einem extra Verzeichniss abgespeichert, so dass anschließend der Raytracer die Einzelbilder berechnen kann. Im Ausgangsordner wird eine RIF-Datei erstellt (RIF = render information file), die als weiterführende Datei für die Programme `anitmt-view` und `anitmt-server` dient.

1.2. Parameter

Normalerweise wird `anitmt-calc` eine INI-Datei (siehe: VIII Die INI-Datei) als Parameter übergeben. In dieser stehen die Informationen, welche ADL-Dateien die Animation beinhaltet, welche zusätzlichen Dateien mitkopiert werden und die generellen Einstellungen zur Animation wie zum Beispiel Bildgröße, Dauer und Bilder pro Sekunde.

1.3. Bekannte Probleme und häufige Fehler

Eine wichtige Voraussetzung für das Funktionieren von `anitmt-calc` ist, das die ADL im Unix-Format abgespeichert ist und nicht etwa im DOS-Format. Dies ist notwendig, egal unter welches Betriebssystem Sie AniTMT benutzen. Auch die POV-Ray-Szenendateien sollten im UNIX-Format gespeichert sein, da es sonst beim Verändern dieser öfters zu Problemen kommt.

Weiter ist wichtig, dass das Animationsverzeichnis existiert, und dass `anitmt-calc` in dieses Verzeichnis schreiben kann. Das Animationsverzeichnis wird in der INI-Datei durch den Eintrag `ani_dir` festgelegt.

Probleme gibt es derzeit mit größeren Animationen mit vielen Referenzen. Dort kann es vorkommen, dass die Verweise nicht richtig aufgelöst werden und durch zu früh gesetz-

te Standardwerte falsche Werte entstehen, Sprünge in der Animation können die Folge sein. Es kann auch vorkommen, dass `anitmt-calc` die Definitionen gar nicht auflösen kann. Diese Probleme werden jedoch in absehbarer Zeit mit einem neuen System gelöst werden.

Ein weiterer Fehler entsteht, wenn die Include-Dateien nicht vorhanden sind. `anitmt-calc` erstellt dann im Animationsverzeichniss leere Kopien der Dateien. POV-Ray stört dies im wesentlichen nicht, jedoch fehlen die dort definierten Objekte und Werte. Deswegen sollten auch Standard Include-Dateien wie beispielsweise `colors.inc` mit in den Ausgangsordner kopiert werden.

2. Bedienung von `anitmt-view`

2.1. Funktionsweise

`anitmt-view` berechnet den von `anitmt-calc` verbereiteten Film mit Hilfe des Raytracers POV-Ray. Das Programm entnimmt aus der RIF-Datei (render information file), die `anitmt-calc` erzeugt hat, die wichtigen Informationen über den Film. Zusätzlich muss man dem Programm noch den Pfad für den POV-Ray-Aufruf geben. Auch Optionen, wie zum Beispiel, den Film nur teilweise berechnen oder mit weniger Bilder pro Sekunde sind möglich.

`anitmt-calc` nimmt für jedes Einzelbild immer eine der Frame-Include-Dateien (`f0000.inc`, `f0001.inc`, ...), benennt diese in `frame.inc` und lässt von dem Raytracer das Bild berechnen. Danach kommt das nächste Bild dran, bis alle gewünschten Bilder berechnet sind.

2.2. Parameter

Die folgende Liste erklärt die möglichen Parameter. Parameter, die Standardwerte besitzen, müssen nicht angegeben weden, wenn diese korrekt sind.

- `-p` *POV-Ray-Aufruf*
  Standard: `povray`
  Gibt an, wie POV-Ray aufgerufen werden soll. Geben Sie hier den Pfad und die ausführbare Datei von POV-Ray an, je nachdem, wo sich dieses Programm auf ihrem System befindet. Bitte beachten Sie, dass auch unter Windows der Pfad mit Schrägstrichen (/) angegeben werden muss, und nicht etwa mit Backslash (\). Verwenden sie von POV-Ray möglichst Kommandozeilen-Versionen, und keine mit zusätzlichem GUI (grafische Benutzeroberfläche).

- `-r` *RIF-Datei*
  Standard: `ani.rif`
  Gibt die von `anitmt-calc` erzeugte RIF-Datei (`render information file`) an, die die Informationen über die Animation beinhaltet.

- `-s` *Erstes_Bild*
  Standard: `0`
  Gibt das erste Bild eines zu berechnenden Ausschnittes aus der Animation an.

- `-e` *Letztes_Bild*
  Standard: Letztes Bild der gesammten Animation
  Gibt das letzte Bild eines zu berechnenden Ausschnittes aus der Animation an.

- `-j` *Schrittweite*
  Standard: `1`
  Gibt eine Schrittweite für das Berechnen der Bilder an. Schrittweite `1` bedeutet, dass jedes Bild berechnet wird, `2` entspricht jedem zweiten Bild, `3` jedem dritten Bild, und so weiter.

- `-a`
  Erstellt ein unkomprimiertes AVI-Video nach dem Berechnen der Bilder mit dem Aufruf `tga2avi`.

- `-A`
  Erstellt ein cram16 kompremiertes AVI-Video nach dem Berechnen der Bilder mit dem Aufruf `mkavi`.

- `-ppm`
  Zwingt POV-Ray dazu, keine TGA-Bilder zu erstellen, sondern PPM-Bilder.

- `-redo`
  Nimmt einen abgebrochenen Berechnungsvorgang wieder auf. Die Bilder werden anhand der Dateigröße auf ihre Vollständigkeit untersucht. Fehlende Bilder werden neu berechnet und Bilder, die noch nicht fertig berechnet wurden werden dann zu Ende berechnet. Auch wenn in einer Animation einige Bilder fehlerhaft sind können diese einfach gelöscht werden und geschickt mit der redo-Funktion neuberechnet werden.

- `-help`
  Zeigt die Kurzhilfe des Programmes an.

2.3. Bekannte Probleme

Es ist wichtig, dass der Pfad des Programmes POV-Ray auch unter Windows immer mit Schrägstrichen (/) angegeben wird und nicht mit Backslash (\).

Für bestimmte Versionen von POV-Ray ist es wichtig, dass schon vor der Verwendung von `anitmt-calc` in der INI-Datei der Parameter `params= -p` angegeben wird (`-p` = pause_when_done). Ansonsten hält POV-Ray nach jedem Bild an und wartet auf einen Tastendruck. Meist ist es auch ratsam, die Darstellung von POV-Ray mit `-d` abzuschalten.

3. Bedienung von `tga2avi` und `mkavi`

3.1. Funktionsweise

Die beiden Programme `tga2avi` und `mkavi` sind beide dazu da, ein AVI-Video aus einer Reihe von TGA-Bildern zu erzeugen. Der Unterschied zwischen den beiden Programmen besteht darin, dass `tga2avi` nur unkomprimierte Videos erzeugt. `mkavi` erstellt dagegen cram16-komprimierte Videos, die etwas kleiner, dafür aber auch einen gewissen Qualitätsverlust haben. Beide Programme sind jedoch nur als Notlösung gedacht. Es gibt mit Sicherheit bessere Video-Formate und Kompressionsverfahren und bessere Programme, um derartige Videos zuerstellen. Wir empfehlen, derartige andere Programme zu verwenden, falls diese auf ihrem System vorhanden sind!

3.2. Parameter

Die meisten Parameter sind für beide Programme identisch. Für den Aufruf von `tga2avi` oder `mkavi` müssen sie in der Regel nur die Nummer des letzten Bildes mit „`-e` Nummer“ angeben.

Die folgenden Parameter gelten sowohl für `tga2avi` als auch für `mkavi`:

- `-e` *Letztes_Bild*
  Gibt die Nummer des letzten Bildes an. Beachten Sie dabei, dass die Nummerierung mit 0 beginnt.

- `-s` *Erstes_Bild*
  Standard: `0`
  Gibt die Nummer des ersten Bildes an.

- `-f` *fps*
  Standard: `24`
  Gibt die Anzahl der Bilder pro Sekunde an (fps = frames per second).

- `-j` *Schrittweite*
  Standard: `1`
  Gibt eine Schrittweite der Bilder an, die für das Video verwendet werden sollen. Schrittweite `1` bedeutet, dass jedes Bild verwendet wird, `2` entspricht jedem zweiten Bild, `3` jedem dritten Bild, und so weiter. Die Anzahl der Bilder pro Sekunde wird dann auch automatisch entsprechend halbiert oder gedrittelt. Die Schrittweite kann verwendet werden, um schneller fertige Ergebnisse zu erhalten, wenn `anitmt-view` mit derselben Schrittweite gestartet wurde.

- `-i` *Dateinamen_der_Bilder*
  Standard: `f%04d.tga`
  Gibt die Dateinamen der Eingabedateien an, die eine Ziffernfolge beinhalten. Mit `%0nd` wird die Anzahl der Ziffern auf n Ziffern festgelegt. `f%04d.tga` bedeutet also: `f0000.inc`, `f0001.inc`, ...

- `-o` *Videoname*
  Standard: `film.avi`
  Gibt den Dateinamen des Videos an, das erstellt wird.

- `-help`
  Zeigt die Kurzhilfe des Programmes an.

Die folgenden Parameter gelten nur noch für das Programm `mkavi`:

- `-t` *Bildformat*
  Standard: `tga`
  Gibt das Bildformat der Ausgangsdateien an. Möglich sind: `tga` oder `ppm`.

- `-c` *Codec*
  Standard: `cram16`
  Gibt den zu verwendenden Video-Codec für das AVI an. Möglich sind: `cram16` oder `rgb24`.

3.3. Bekannte Probleme

Probleme gibt es derzeit, wenn die TGA-Bilder verkehrt herum abgespeichert sind, wie einige Windows-Programme dies vorzugsweise machen. In diesem Fall ist dann nachher auch das gesamte Video verkehrt herum.

4. Bedienung von `anitmt-server` und `anitmt-client`

Bitte schauen Sie hierzu im Network-Guide nach.

Teil V.
Wie möchte ich die Szene animieren?

Prinzipiell gibt es zwei Möglichkeiten, eine 3-dimensionale Szene zu animieren:

1. Objekte animieren

Objekte können als eine Einheit auf einer Flugbahn bewegt werden. Zusätzlich kann die Richtung des Objektes in Abhängigkeit von der Flugbahn verändert werden. Ein ganzes Objekt zu animieren ist immer sinnvoll, sobald sich das Objekt entlang einer zusammengesetzten Bahn bewegen soll.

2. Werte animieren

Bei der Interpolation von Werten wird immer eine in der Szenenbeschreibung definierte Variable benutzt. Diese Werte können nach belieben verwendet oder weiter verrechnet werden.

Werte können benutzt werden, wenn bestimme Eigenschaften (wie zum Beispiel Farbe) eines Objektes oder der Szene animiert werden sollen. Auch zum Verändern der Größe mit einer scale-Anweisung oder zum Rotieren von Objekten um nur eine Achse sind einzelne Werte sinnvoll. Die Benutzung der Variablen muß explizit in der Szenenbeschreibung angegeben werden.

Teil VI.
Vorbereiten der Szenenbeschreibung

Eine Szene besteht aus mehreren Komponenten. Diese sind entweder ganze Objekte, die bewegt und gedreht werden können oder einfache Werte. Alle Komponenten einer Szene, die animiert werden sollen, müssen dort benannt werden.

1. POV-Dateien

Im folgenden wird für POV-Ray Szenen, wie man die Komponenten darin animierbar macht.

1.1. Kennzeichnung der Objekte

Um Objekten für eine Animation einen Namen zuzuweisen, wird folgende Syntax verwendet:

```
sphere {                  // My_Sphere        <-- Dies ist der Name
  < 0, 0, 0 >, 0.45
  pigment { OldGold }
  rotate < 0, 0, 0 >
  translate < -4, 1, 1 >
}
```

Hinter dem Objekttyp (sphere) und der geschweiften Klammer wird mit 2 Schrägstrichen ein Kommentar eingeleitet. Das erste darauf folgende Wort wird von AniTMT als Name des Objektes übernommen. (Siehe auch: 4 Regeln für Namensgebung)

1.2. Variablen einfügen

Variablen muss nicht extra ein Name zugewiesen werden. Statt dessen wird der Variablenname verwendet. In der Szene muss, wie folgt, eine `#declare`-Anweisung stehen, die der Variable einen Standardwert zuweist:

```
#declare MyVariable = 12345;
```

Die Anweisung muss unbedingt mit einem Semikolon abgeschlossen werden, damit `anitmt-calc` das Ende der Anweisung erkennt. Wenn diese Varible animiert werden soll, wird `anitmt-calc` für die Einzelbilder alle `#declare`-Anweisungen für die Varible aus der Szenebeschreibung löschen und statt dessen eine eigene einfügen. Aus diesem Grund sollte, wenn man die Variable weiter verwenden will, um beispielsweise ein Schleife abzuarbeiten, nicht der selbe Variablenname verwendet werden. Statt dessen müssen für weitere Verechnungen eine andere Variable benutzt werden, wie in dem folgendem Beispiel gezeigt:

```
#declare MyStartValue = 1;
#declare MyEndValue   = 9;
#declare i = MyStartValue;

#while (i <= MyEndValue)

  box {
    <0,0,0>, <1,1,1>
    pigment { color rgb < i * 0.1, 0, 0 > }
    translate i * y
  }

  #declare i = i + 1;
#end
```

Wenn man versuchen würde, anstelle von „i“ „MyStartValue“ inerhalb der Schleife zu verwenden, würde dieses Beispiel nicht funktionieren, da `anitmt-calc` auch die Zeile

```
#declare MyStartValue = MyStartValue + 1;
```

löschen würde.

Teil VII.
Animation Description Language

1. Aufbau und Syntax

Die ADL-Datei ist hierachisch in verschiedene Ebenen eingeteilt: Szene, Komponente (Objekt/Variable), Funktion, Subfunktionen

```
povscene MeineSzene {       // POV-Ray-Szene
  filename "myscene.pov";     // Szenendatei
  my_obj {                    // Name des Objektes
    move {                      // Bewegung auf Flugbahn
      straight {                  // Gerade zum Ursprung
        startpos    <-2,0,0>;       // Startposition
        startdir    x;              // Startrichtung
        length      3;              // Laenge in LE
        startspeed  2;              // Geschwindigkeit in LEs
      }
      circle {                    // 180 Grad Kurve
        normal      y;              // X-Z Ebene
        radius      2;              // Radius in LE
        angle       180;            // Winkel in Grad
      }
      circle {                    // 130 Grad Kurve schief im Raum
        center      <1,1,-1>;       // Rotationszentrum
        angle       130;            // Winkel in Grad
      }
      straight {}                   // Gerade in die Unendlichkeit
    }
  }
}
```

Grundsätzlich werden zwei Angabearten unterschieden. Man kann Blöcke öffnen, deren Inhalt in geschweifte Klammern eingeschlossen wird, und man kann Eigenschaften eines Blockes definieren.

Blöcke werden durch einen Bezeichner (z. B. `povscene`) eingeleitet. Wenn später auf diesen Block verweisen können soll, muß man ein Name hinzufügen (im Beispiel `MeineSzene`), mit dem man diesen wieder ansprechen kann.

Eigenschaften bestehen aus einem Bezeichner (z. B. `filename`) und einem durch ein Leerzeichen abgetrennten Wert (z. B. `"myscene.pov"`) mit abschließendem Semikolon. Werte können sowohl Skalare als auch Vektoren, Zeichenketten oder komplette mathematische Ausdrücke sein. In jedem Block sind nur bestimmte Eigenschaften möglich.

Auf oberster Ebene werden Szeneblöcke geöffnet, bei POV-Ray durch den Bezeichner `povscene`. Alle Szenen benötigen die Angabe `filename`, die einen Dateinamen enthalten soll. Innerhalb der Szenen werden die Komponentenblöcke mit dem zuvor vergebene Namen aus der Szenenbeschreibung eingeleitet (siehe: VI Vorbereiten der Szenenbeschreibung)

In den Komponenten können Funktionsblöcke aufgemacht werden. Die Funktion `change` animiert zum Beispiel Skalare, während die Funktion `move` Objekte durch Aneinanderreihung von Bahnelementen bewegt. Durch diese Funktionen wird der Typ einer Komponente bestimmt. Die Elemente einer Funktion werden durch Subfunktionsblöcke definiert. In ihnen werden die eigentlichen Eigenschaften festgelegt, die das Verhalten der Animation bestimmen.

2. Eigenschaften und Werte

Das eigentliche Verhalten der Animation wird durch die Eigenschaften der Subfunktionen gesteuert. Normalerweise werden den Eigenschaften sofort ein fester Wert zugewiesen:

```
startpos    < 5, 2, 3.141 >;
```

Es jedoch auch möglich, Werte zu verrechnen und Konstanten zu verwenden:

```
startpos    (5 * x) + (2 * <0,1,0>) + <0,0,pi>;
```

Im folgenden werden die Möglichkeiten zur Verrechnung von Ausdrücken aufgelistet.

2.1. Operatoren

Mögliche mathematische Operatoren sind:

1. Rangstufe: `( )`
2. Rangstufe: `+ -` (Als Vorzeichen)
3. Rangstufe: `* /`
4. Rangstufe: `+ -`

Beachten Sie, dass die Operatoren `*`, `+`, `-` auch mit Vektoren funktionieren, und dort entsprechende Bedeutungen haben. Als Multiplikation (`*`) zweier Vektoren wird das Skalarprodukt (`vdot(A,B)`) verwendet (siehe: 2.4 Vektor-Funktionen). Auch bei Zeichenketten ist das Pluszeichen (`+`) zum Aneinanderhängen erlaubt (vergl.: 2.5 Funktionen für Zeichenketten, `concat`).

Des weiteren gibt es Vergleichsoperatoren, mit denen mathematische Ausdrücke vergli-

chen werden können:

`A < B`	`A` kleiner `B`.
`A <= B`	`A` kleinergleich `B`.
`A = B`	`A` gleich `B`.
`A != B`	`A` ungleich `B`.
`A >= B`	`A` größergleich `B`.
`A > B`	`A` größer `B`.

Für logische Vergleiche gibt es:

`A & B`	`A` und `B` muss erfüllt sein.
`A | B`	`A` oder `B` muss erfüllt sein.

Außerdem gibt es noch den konditionalen Vergleich:

`(C ? A : B)`	Wenn Bedingung `C` erfüllt ist, dann liefere `A` zurück, sonst `B`.

2.2. Fest definierte Konstanten

Diese fest definierten Konstanten können anstelle von Werten zur Berechnung verwendet werden.

```
true  = 1;
yes   = 1;
on    = 1;
false = 0;
no    = 0;
off   = 0;
pi    = 3.14159265358979323846;
e     = 2.718281828459045235360;
x     = <1,0,0>;
y     = <0,1,0>;
z     = <0,0,1>;
```

2.3. Mathematische Funktionen

Diese Funktionen stehen zur Verrechnung von einfachen Werten zur Verfügung. `A` und `B` stehen für einfache Werte.

`abs(A)`: Absoluter Wert von `A`. Wenn `A` negativ ist, wird `-A` zurückgegeben, ansonsten `A`.

`acos(A)`: Arcus-Kosinus von `A`. Gibt den Winkel in Bogenmaß zurück, dessen Kosinus `A` ist.

`asin(A)`: Arcus-Sinus von `A`. Gibt den Winkel in Bogenmaß zurück, dessen Sinus `A` ist.

`atan2(A,B)`: Arcus-Tangens von `(A/B)`. Gibt den Winkel in Bogenmaß zurück, dessen Tangens `A` ist. Gibt auch den entsprechenden Wert zurück, wenn `B` null ist. Benutzen Sie `atan2(A,1)` um die normale `atan(A)`-Funktion zu berechnen.

`ceil(A)`: Aufrunden von `A`. Gibt den kleinsten ganzzahligen Wert größer oder gleich `A` zurück. Rundet zur nächst höheren ganzen Zahl auf.

`cos(A)`: Kosinus von `A`. Gibt den Kosinus von dem Winkel `A` zurück, der in Bogenmaß gemessen wird.

`degrees(A)`: Wandelt Bogenmaß in Grad um. Gibt den Winkel in Grad zurück, dessen Wert in Bogenmaß `A` ist. Die Formel lautet `degrees=A/pi*180`.

`div(A,B)`: Ganzzahldivision. Gibt den ganzzahligen Wert von `A/B` zurück.

`exp(A)`: natürliche Potenz. Gibt das Ergebnis von `e` hoch `B` zurück. `e` ist die Basis des natürlichen Logaritmus und ungefähr `2.71828182846`.

`floor(A)`: Abrunden von `A`. Gibt den größten ganzzahligen Wert kleiner oder gleich `A` zurück. Rundet zur nächten niedrigeren ganzen Zahl ab.

`int(A)`: Ganzzahliger Teil von `A`. Gibt den abgeschnittenen ganzzahligen Wert von `A` zurück. Rundet Richtung null.

`log(A)`: Natürlicher Logaritmus von `A`. Gibt den natürlichen Logaritmus zur Basis `e` von `A` zurück. `e` ist ungefähr `2.71828182846`.

`max(A,B)`: Maximum von `A` und `B`. Gibt `A` zurück, wenn `A` größer ist wie `B`, ansonsten `B`.

`min(A,B)`: Minimum von `A` und `B`. Gibt `A` zurück, wenn `A` kleiner ist wie `B`, ansonsten `B`.

`mod(A,B)`: Ergebnis von `A` modulo `B`. Gibt den Rest, der Ganzzahldivision von `A/B`. zurück. Die Formel lautet `mod=((A/B)-int(A/B))*B`.

`pow(A,B)`: Potenzieren. Gibt das Ergebnis von `A` hoch `B` zurück.

`radians(A)`: Wandelt Grad in Bogenmaß um. Gibt den Winkel in Bogenmaß zurück, dessen Wert in Grad `A` ist. Die Formel lautet `degrees=A*pi/180`.

`sin(A)`: Sinus von `A`. Gibt den Sinus von dem Winkel `A`, der in Bogenmaß gemessen wird.

`sqrt(A)`: Quadrat-Wurzel von `A`. Gibt den Wert zurück, dessen Quadrat `A` ist.

`tan(A)`: Tangens von `A`. Gibt den Tangens von dem Winkel `A` zurück, der in Bogenmaß gemessen wird.

2.4. Vektor-Funktionen

Diese Funktionen stehen zur Verrechnung von Vekoren zur Verfügung. `A` und `B` stehen dabei für 3-dimensionale Vektoren und F für einen einfachen Wert.

`vaxis_rotate(A,B,F)`:	Rotiere Punkt `A` um die Achse durch den Ursprung mit dem Vektor `B` um dem Winkel `F`. Das Ergebnis sind die Koordinaten des gedrehten Punktes
`vcross(A,B)`:	Kreuzprodukt von `A` und `B`. Das Ergebnis ist ein Vektor, der senkrecht zu den zwei orginalen Vektoren ist und die Länge ist proportional zum Winkel zwischen den beiden.
`vdot(A,B)`:	Skalarprodukt von `A` und `B`. Liefert einen Wert zurück, der das Skalarprodukt von `A` und `B` darstellt. Die Formel lautet `(A.x*B.x) + (A.y*B.y) + (A.z*B.z)`.
`vlength(A)`:	Länge von `A`. Gibt die Länge des Vektors `A` zurück. Kann benutzt werden, um den Abstand zwischen zwei Punkten zu bestimmen `Dist=vlength(B-A)`. Die Formel lautet `vlength=sqrt(vdot(A,A))`.
`vnormalize(A)`:	Einheitsvektor von `A`. Gibt einen Vektor mit der Länge eins zurück, der die selbe Richtung wie `A` hat. Die Formel lautet `vnormalize=A/vlength(A)`.
`vrotate(A,B)`:	Rotiert den Punkt `A` um den Ursprung entprechend `B`. Der Punkt `A` wird erst um die x-Achse entsprechend des Wertes `B.x`, dann um die y-Achse entsprechend `B.y` und die z-Achse mit `B.z`. Alle Werte werden in Grad gemessen. Das Ergebnis ist der Ortsvektor des neuen Punktes.

2.5. Funktionen für Zeichenketten

Diese Funktionen stehen zur Verrechnung von Zeichenketten zur Verfügung. `S1`, `S2` und `S3` stehen dabei für eine Zeichenkette und `A`, `L` und `P` für normale Werte.

`asc(S1)`:	ASCII-Wert von `S1`. Gibt einen ganzzahligen Wert von 0 bis 255 der dem ASCII-Wert des ersten Zeichens von `S1` entspricht. Zum Beispiel `asc("ABC")` ist 65, weil das der Wert des Zeichen `"A"` ist.
`chr(A)`:	Zeichen mit ASCII-Wert `A` ist. Gibt ein Zeichen zurück, dessen ASCII-Wert `A` ist. `A` muss zwischen 0 und 255 liegen.
`concat(S1,S2,[S3...])`:	Hängt die Zeichenketten `S1`, `S2`, `S3`, ... aneinander. Diese Funktion muss mindestens zwei Parameter oder mehr haben.

`str(A,L,P)`: Wandelt den Wert `A` in eine Zeichenkette um. Gibt den Wert `A` als eine formatierte Zeichenkette zurück. Der Wert `L` entspricht der Minimallänge der Zeichenkette und gibt die Formatierung an, wenn das Ergebnis kürzer wie die Minimallänge ist. Wenn `L` positiv ist, wird der Platz vor der Zahl mit Leerzeichen aufgefüllt. Wenn `L` negativ ist, dann wird der Platz mit Nullen aufgefüllt. Wenn die Zeichenkette länger als das Minimum sein muss, wird sie so lang gemacht, wie es nötig ist, um die Zahl darzustellen. Der Wert `P` legt die Zahl der Nachkommastellen fest. Wenn `P` negativ ist, wird ein kompiler-spezifischer Standard Wert benutzt. Hier sind einige Beispiele:

```
str(123.456,0,3)   "123.456"
str(123.456,4,3)   "123.456"
str(123.456,9,3)   "  123.456"
str(123.456,-9,3)  "00123.456"
str(123.456,0,2)   "123.46"
str(123.456,0,0)   "123"
str(123.456,5,0)   "  123"
str(123.000,7,2)   " 123.00"
str(123.456,0,-1)  "123.456000" (Plattform spezifisch)
```

`strcmp(S1,S2)`: Vergleiche Zeichenkette `S1` mit `S2`. Gibt den Wert null zurück, wenn die beiden Zeichenketten gleich sind, einen positiver Wert, wenn `S1` vor `S2` beim Vergleich der ASCII-Werte kommt und, wenn es umgekehrt ist, einen negativen Wert

`strlen(S1)`: Länge der Zeichenkette `S1`. Gibt die Anzahl der Zeichen in der Zeichenkette `S1` zurück.

`strlwr(S1)`: Kleinbuchstaben von `S1`. Gibt eine neue Zeichenkette zurück, in der alle Großbuchstaben in Kleinbuchstaben umgewandelt wurden.

`substr(S1,P,L)`: Teilstück der Zeichenkette `S1`. Gibt ein Teilstück der Zeichenkette `S1` zurück, beginnend an der Position `P` mit der Länge von `L` Zeichen. Zum Beispiel `substr("ABCDEFGHI",4,2)` liefert die Zeichenkette `"EF"`. Wenn `P+L > strlen(S1)` ist, tritt ein Fehler auf.

`strupr(S1)`: Großbuchstaben von `S1`. Gibt eine neue Zeichenkette zurück in der alle Kleinbuchstaben in Großbuchstaben umgewandelt wurden.

`val(S1)`:	Wandelt die Zeichenkette `S1` in einen Wert um. Gibt einen Wert zurück, der durch den Text in `S1` dargestellt wird. Zum Beispiel `val("123.45")` liefert 123.45 als einen Wert.

3. Funktionen

3.1. Allgemeines

Um Komponenten der Szene animieren zu können, braucht man verschiedene Funktionen (nicht verwechseln mit den Funktionen im mathematischen Sinn). Eine Funktion bestimmt, wie eine Komponente animiert wird. Für ein Objekt gibt es zum Beispiel die Funktion `move`. Diese Funktion, die nur bei Objekten zulässig ist, besagt, dass das Objekt sich entlang einer Flugbahn bewegen sollen, die aus mehreren Teilstücken definiert ist.

Da bei AniTMT die Definition einer Animation möglichst komfortabel sein soll, gibt es für alle Funktion, also sowohl für Werte- und Vektorinterpolationen als auch für Flugbahnen folgende Möglichkeiten:

3.1.1. Bestimmung über beliebige Eigenschaften

Jede Subfunktion besitzt verschiedene Eigenschaften, wie zum Beispiel `starttime`, `endtime`, `startpos`, `endpos`, `duration`, ...

Wenn nun eine Eigenschaft benötigt wird, so sind alle Möglichkeiten bekannt, sie aus den anderen Eigenschaften zu berechnen. Bei einer geraden Bahn wird beispielsweise die Endposition (endpos) aus Startposition (startpos), Startrichtung (startdir) und Länge (length) berechnet. Falls diese Eigenschaften ebenfalls nicht angegeben wurden, bestehen wiederum mehrere Möglichkeiten, auch diese Werte zu ermitteln.

Es genügt also, einen Abschnitt eindeutig zu bestimmen, so dass alle anderen Werte ausgerechnet werden können.

3.1.2. Wertebestimmung über Nachbarelemente

Die Ermittlung der benötigten Eigenschaften erfolgt in mehreren Stufen. Zuerst werden in jedem Abschnitt die Unbekannten, wie gerade beschrieben, ermittelt. Wenn Subfunktionen nicht sämtliche ihrer Eigenschaften berechnen konnten, werden entsprechende Werte von Nachbarelementen übernommen.

Um zum Beispiel bei einer Flugbahn einen zeitlich zusammenhängenden Vorgang sicherzustellen, sollten die Start- und Endzeiten benachbarter Elemente übereinstimmen. Um Sprünge zu vermeiden, benötigt man gleiche Positionen; um Knicke zu verhindern, gleiche Richtungen; um die Animation flüssig zu gestalten, gleiche Geschwindigkeiten. Entsprechendes gilt natürlich auch für Werte und Vektorinterpolationen. Auf jeder Stufe wird eine Eigenschaft in obiger Reihenfolge zur Übergabe freigegeben.

Damit ist es möglich, eine Animation mit relativen Veränderungen, wie der Dauer, zu definieren und nur an einer Stelle absolute Werte, wie zum Beispiel die Startzeit, anzugeben.

3.1.3. Vergabe von Standardwerten

Da wir sowohl mächtige als auch komfortable Funktionen anbieten wollen, werden bei Abschnitten, die noch nicht eindeutig definiert sind, Standardwerte gesetzt. Damit wird beispielsweise statt einer beschleunigten Bewegung eine gleichförmige Bewegung angenommen. Einige Anfangswerte des ersten Abschnitts werden standardmäßig gesetzt. Auch die Endzeit des letzten Abschnittes kann mit Hilfe der Endzeit der gesamten Animation ermittelt werden.

3.1.4. Verweise auf Werte anderer Elemente

Desweiteren ist es möglich, Beziehungen zwischen mehreren Bewegungen herzustellen.

Beispielsweise kann man eine Rakete durch Verweise auf die Eigenschaften eines Flugzeuges so ausrichten, dass sie dieses zum richtigen Zeitpunkt trifft, ohne die exakte Position und Zeit zu bestimmen.

Generell ist es ja auch möglich, die Werte weiter zu verrechnen, um beispielsweise die Rakete das Flugzeug verfehlen zu lassen, indem sie die Bahn erst einige Sekunden später kreuzt.

3.1.5. Separators

Manchmal möchte man einen Positions- oder Geschwindigkeits-Sprung zwischen benachbarten Elementen haben. Jedoch kann es sein, dass man nicht zwei unterschiedliche Werte an der Trennstelle angeben möchte, beispielsweise weil sich einer der Werte aus anderen Eigenschaften berechnen soll. Jedoch kommt es dann vor, dass der Wert aus dem einen Abschnitt, in den anderen Abschnitt übergeben wird. Dann erhält man natürliche kein Sprung an dieser Stelle. Statt dessen taucht der Sprung vielleicht wo ganz anders auf, oder es entsteht sogar eine Überdefinition.

Um dies zu verhindern, gibt es den `separator`. Ein Separator dient zum Kennzeichnen der Trennstelle. In der Syntax wird er wie jede andere Subfunktion gehandhabt. Er stellt jedoch bei der Animation kein eigener Abschnitt dar. In einem Separator wird nur festgelegt, welche Werte von Abschnitt vor dem Separator in den Abschnitt danach und umgekehrt passieren dürfen und welche nicht. Genaueres zur Syntax des Separators erfahren Sie im Abschnitt 3.2.3 Spezielle Subfunktionen.

3.2. Liste aller Funktionen

3.2.1. Werteinterpolation (change)

Diese Funktion dient zum Animieren einer Variablen, die der Anwender beliebig im Programm weiter verwenden kann, um zum Beispiel Texturen oder Objekte wie Türen

(Öffnungswinkel) und ähnliches zu verändern.

Jede Subfunktion hat Standardeigenschaften, die bei allen möglich sind, und weitere spezielle. Die folgenden Eigenschaften sind für alle Subfunktionen gleich:

`starttime`	Startzeitpunkt
`endtime`	Endzeitpunkt
`duration`	Dauer
`startframe`	Startbild
`endframe`	Endbild
`frames`	Anzahl Bilder
`startvalue`	Startwert
`endvalue`	Endwert
`difference`	Unterschied zwischen Start- und Endwert

Für die Werteinterpolationen gibt es 2 Subfunktionen:

- `linear`
- `accelerated`

linear Die Funktion linear entspricht einer konstanten Zunahme eines Wertes. Als weitere Eigenschaften für linear gibt es:

`slope`	Zunahme in Einheiten pro Sekunde

accelerated Die Funktion accelerated entspricht einer konstant beschleunigten Zunahme eines Wertes. Als weitere Eigenschaften für linear gibt es:

`startslope`	Zunahme in Einheiten pro Sekunde am Anfang
`endslope`	Zunahme in Einheiten pro Sekunde am Ende
`acceleration`	„Beschleunigung“

3.2.2. Bewegung von Objekten mittels einer Flugbahn (move)

Für alle Objekte wird eine „Vorne“-Richtung und eine „Oben“-Richtung bestimmt, die ihre ursprüngliche Lage in der Szene festlegt. Diese werden zusammen mit dem Drehzentrum `center` im Objekt wie folgt angegeben:

```
povscene{
  my_object{
    center      <1,1,1>;
    front       <1,0,1>;
    up          y;
```

```
    move{...}
  }
}
```

Der Standardwert für das Drehzentrum is `<0,0,0>`, der für `front` ist `x` und der für `up` ist z.

Um die Lage eines Körpers im dreidimensionalen Raum eindeutig zu bestimmen, verwenden wir ein System, das eine Blickrichtung durch zwei senkrecht aufeinander stehende Winkel definiert, und die Rotation des Objektes um die Blickrichtung angibt.

Ein Flugzeug, zum Beispiel, könnte man als erstes um seine Blickrichtung oder Längsachse drehen. Zusätzlich ist eine Drehung um die Querachse möglich oder auch eine Ausrichtung des Cockpits nach oben oder unten. Beide Achsen werden durch die „Oben"-Richtung festgelegt, nachdem diese in Abhängigkeit zur Flugbahn gedreht wurde. Zuletzt wird es dann an die richtige Position geschoben. Dadurch werden alle erdenkbaren Lagen abgedeckt.

Die Funktion `move` ist an dieses Konzept angelehnt. Mit einzelnen Abschnitten, wie Gerade oder Kreisbogen, wird eine Flugbahn oder Position bestimmt. Das Objekt schaut mit seiner Blick- oder Vornerichtung zunächst immer tangential zur Flugbahn. Wenn das Objekt durch eine Kurve fliegt, wird die Obenrichtung entsprechend dieser gedreht.

Wenn man jedoch ein Objekt haben möchte, dessen Lage immer gleich ist (zum Beispiel ein Ufo) gibt es die Möglichkeit, die automatische Rotation in Abhängigkeit zur Bahn auszuschalten. Dazu kann man die Eigenschaft der Funktion move wie folgt festlegen:

```
autorotate false;
```

Alle Subfunktionen unterstützen beschleunigte Bewegungen und Rotation um die drei Achsen. Deswegen besitzen alle Abschnitte einer Flugbahn folgende Eigenschaften:
Allgemein:

`starttime`	Startzeitpunkt
`endtime`	Endzeitpunkt
`duration`	Dauer
`startframe`	Startbild
`endframe`	Endbild
`frames`	Anzahl Bilder
`startup`	„Oben"-Vector am Anfang
`endup`	„Oben"-Vector am Ende
`up_roll`	Diese Zahl gibt die Rotation des „Oben"-Vectors um die Bahn in Grad an

Bewegung:

`startpos`	Startposition
`endpos`	Endosition
`startdir`	Startrichtung
`enddir`	Endrichtung
`startspeed`	Startgeschwindigkeit
`endspeed`	Endgeschwindigkeit
`acceleration`	Beschleunigung

Rotation um die Blickrichtung:

`startrot_roll`	Startwert in Grad
`endrot_roll`	Endwert in Grad
`diffrot_roll`	Unterschied in Grad
`startrotspeed_roll`	Rotationsgeschwindigkeit am Anfang in Grad pro Sekunde
`endrotspeed_roll`	Rotationsgeschwindigkeit am Ende in Grad pro Sekunde
`rotacceleration_roll`	Beschleunigung der Rotation

Rotation in der Horizontalen:

`startrotspeed_horizontal`	Startwert in Grad
`endrotspeed_horizontal`	Endwert in Grad
`diffrot_horizontal`	Unterschied in Grad
`rotacceleration_horizontal`	Rotationsgeschwindigkeit am Anfang in Grad pro Sekunde
`startrot_horizontal`	Rotationsgeschwindigkeit am Ende in Grad pro Sekunde
`endrot_horizontal`	Beschleunigung der Rotation

Rotation in der Vertikalen:

`startrot_vertical`	Startwert in Grad
`endrot_vertical`	Endwert in Grad
`diffrot_vertical`	Unterschied in Grad
`startrotspeed_vertical`	Rotationsgeschwindigkeit am Anfang in Grad pro Sekunde
`endrotspeed_vertical`	Rotationsgeschwindigkeit am Ende in Grad pro Sekunde
`rotacceleration_vertical`	Beschleunigung der Rotation

Natürlich muß für eine Flugbahn meist nur ein sehr kleiner Teil dieser Eigenschaften angegeben werden. Wenn zum Beispiel keine Rotationen stattfinden sollen, muss keine dieser Eigenschaften angegeben werden. Die drei Rotationswinkel werden dann automatisch auf 0 gesetzt und das Objekt dreht sich nur mit der Flugbahn. (Siehe auch: 3.1.3 Vergabe von Standardwerten)
Für die Bewegung von Objekten gibt es 3 Subfunktionen:

- `straight`
- `circle`
- `bezier`

straight Die Subfunktion `straight` bewegt das Objekt auf einer Geraden. Als einzige weitere Angabe kommt die Streckenlänge hinzu:

`length`	Streckenlänge

circle Die Subfunktion `circle` bewegt das Objekt auf einer Kurve im Raum. Mögliche weitere Angaben sind:

`length`	Streckenlänge
`normal`	Normalenvektor der Ebene in der die Rotation stattfindet
`radius`	Radius des Kreises
`center`	Mittelpunkt des Kreises
`angle`	Winkel, um den gedreht werden soll

bezier Die Subfunktion `circle` bewegt das Objekt auf einer Freiform-Kurve im Raum. Diese wird durch 4 Punkte definiert. Der erste und der letzte Punkt geben Start und End-Punkt der Kurve an. Die beiden andern Punkte sind Kontrollpunkte. Sie bilden mit den Start- und Endpunkten die Tangenten am Anfang und Ende. Das heißt, dass vom Startpunkt die Kurve als erstes in Richtung des ersten Kontrollpunkt verläuft. Sie krümmt sich dann erst in Richtung des zweiten Kontrollpunktes um dann aus dessen Richtung am Ende anzukommen. Mögliche weitere Angaben sind:

`length`	Streckenlänge
`control_point1`	Kontrollpunkt Nr. 1
`control_point2`	Kontrollpunkt Nr. 2
`dist2point1`	Abstand des 1. Kontrollpunktes zum Startpunkt
`dist2point2`	Abstand des 2. Kontrollpunktes zum Endpunkt

Die Flugbahn einer Bezierkurve wird nur näherungsweise berechnet, indem sie in unzählige Einzelstücke zwischen Punkten zerlegt wird. Immer, wenn der Winkel zwischen zwei

Teilstücken zu groSS ist oder die Länge der Teile zu lange, dann werden die Teilstücke nochmals unterteilt. Diese Werte besitzen Standardvorgaben, die in der Regel gute Ergebnisse liefern sollten. Jedoch kann es sein, dass bei sehr großen oder sehr kleinen Kurven Probleme auftreten. Dann könnten andere Einstellungen weiterhelfen:

`max_angle`	Maximalwinkel zwischen 3 Punkten
`max_len`	Maximallänge zwischen 2 Punkten
`min_len`	Minimallänge zwischen 2 Punkten (wichtig bei spitzen Kurven)
`initial_point_anz`	Anfangszahl an Punkten

3.2.3. Spezielle Subfunktionen

Separator Der Separator dient zum Verhindern, dass ein Werte von zwei benachbarten Abschnitten in den andern übernommen wird. Dies wird mit den folgenden Eigenschaften festgelegt. Jede Eigenschaft kann auf `true` oder `false` gesetzt werden. Somit legt sie für einen oder eine Gruppe von Werten fest, ob diese von dem einen Abschnitt in den anderen passieren dürften und umgekehrt.

`pass_by_default`	Standard für alle Werte
`pass_speeds`	Alle Geschwindigkeitswerte (`speed` und alle `rotspeeds`)
`pass_state`	Alle statischen Werte
`passpos`	Postion
`passdir`	Richtung
`passup`	„Oben“-Richtung
`passspeed`	Geschwindigkeit
`passrot_roll`	Rotationswinkel um die Blickrichtung
`passrotspeed_roll`	Rotationsgeschwindigkeit um die Blickrichtung
`passrot_vertical`	Rotationswinkel in der Vertikalen
`passrotspeed_vertical`	Rotationsgeschwindigkeit in der Vertikalen
`passrotspeed_horizontal`	Rotationswinkel in der Horizontalen
`passrot_horizontal`	Rotationsgeschwindigkeit in der Horizontalen

Die Gruppen `pass_by_default`, `pass_speeds` und `pass_state` beziehen sich immer auf mehrere Werte und legen für diese das Verhalten fest, wenn diese nicht extra angegeben werden. Somit ist es möglich, Positiv- oder Negativlisten zu bilden. Zum Beispiel möchte man grundsätzlich alles passieren lassen (`pass_by_default true;`) nur die Geschwindigkeit nicht (`passspeed false;`). Oder andersherum kann man sagen, kein Wert ausser der Position soll übernommen werden (`pass_by_default false; passpos true;`).

4. Regeln für Namensgebung

In den Szenenbeschreibungen, wie zum Beispiel den POV-Ray-Dateien, konnen den Objekten mit Kommentaren Namen zugewiesen werden. Ebenso kann in der ADL-Datei den einzelnen Blöcken Namen geben, um Verweise auf Eigenschaften von anderen Komponenten möglich zu machen. Für einen solchen Namen gelten folgende Regeln:

- Er darf nur aus Groß- und Kleinbuchstaben (a-z), Ziffern und Unterstrich bestehen.
- Er muss mit einem Buchstaben beginnen.
- Er darf höchstens 100 Zeichen lang sein.
- Es darf kein Name verwendet werden, der bereits in AniTMT als Schlüsselwort, als Funktion, oder ähnliches verwendet wird.
- Groß und Kleinschreibung wird beachtet und unterschieden.

Teil VIII.
Die INI-Datei

Die Einstellungen zum Film werden in einer INI-Datei festgelegt. Der Berechnungsvorgang wird gestartet, in dem diese an `anitmt-calc` übergeben wird.

1. Syntax

Eine INI-Datei besteht aus Abschnitten, Einträgen, und eventuell auch aus Kommentaren. Die Groß- und Kleinschreibung ist zu beachten. Abschnitte werden durch dessen Namen in eckigen Klammern (Bsp: `[Abschnitt]`) eingeleitet. Sie reichen bis zum nächsten Abschnitt oder bis zum Ende der Datei. Einträge werden immer in einer eigenen Zeile definiert. Sie beginnen mit dem Namen des Eintrages, gefolgt von einem Gleichheitszeichen (`=`) und einem Wert. Kommentare werden durch ein Semikolon (`;`) am Zeilenanfang gekennzeichnet.

2. Aufbau

Die INI-Datei für `anitmt-calc` besteht aus zwei Abschnitten: `[files]` und `[options]`. In beiden Abschnitten gibt es Definitionen, die unbedingt vorhanden sein müssen und andere, die vorhanden sein können.

Der Abschnitt `[files]` gibt Auskunft über die beteiligten Dateien und beinhaltet folgende Einträge:

- `adl=`*Dateiname*
  Dieser Eintrag gibt an, welche ADL-Dateie für die Animation verwendet wird. Wenn mehrere ADL-Dateien verwendet werden sollen, führen Sie diesen Eintrag mehrfach mit den entsprechenden Dateien auf. Jedoch muss mindestens eine ADL-Datei definiert sein.

- `copy=`*Dateiname*
  Dieser Eintrag gibt an, welche zusätzlichen Dateien mitkopiert werden sollen. Da `anitmt-calc` automatisch die beteiligten Include-Dateien und Bild-Dateien aus der Szenenbeschreibung versucht herauszulesen, ist dies nur selten nötig. Geben Sie auch hier für jede Datei, diese mit einem eigenen `copy`-Befehl an.

Der Abschnitt `[options]` enthält Einstellungen zur Animation und beinhaltet folgende Einträge:

- `width=`*Breitenangabe*
  Dieser Eintrag gibt an, welche Breite in Pixel der Film haben soll.

- `height=`*Höhenangabe*
  Dieser Eintrag gibt an, welche Höhe in Pixel der Film haben soll.

- `ani_dir=`*Verzeichnisspfad*
  Dieser Eintrag gibt an, in welchem Verzeichnis `anitmt-calc` die vorbereiteten Szenen für den Film ablegen soll. Dies kann jedoch nicht das Ausgangsverzeichnis sein, da es sonst zu Namenskonflikten kommt.

- `duration=`*Filmdauer*
  Dieser Eintrag gibt an, wie lange der Film in Sekunden dauert.

- `fps=`*Bilder_pro_Sekunde*
  Dieser Eintrag gibt an, welche Anzahl Bilder pro Sekunde der Film haben soll.

- `frames=`*Anzahl_Bilder*
  Dieser Eintrag gibt an, wieviele Bilder der Film hat.

- `starttime=`*Startzeitpunkt*
  Standard: `0`
  Dieser Eintrag gibt an, zu welchem Zeitpunkt die Animation starten soll.

- `endtime=`*Endzeitpunkt*
  Dieser Eintrag gibt an, zu welchem Zeitpunkt die Animation beendet werden soll.

- `raytracer=`*Raytracer*
  Standard: `povray3.1`
  Dieser Eintrag gibt an, welcher Raytracer für das Berechnen der Bilder verwendet werden soll. Möglich sind hier `povray3.1` und `povray3.0`. Dies ist insbesondere im Zusammenhang mit `anitmt-server` und `anitmt-client` von Bedeutung, da diese mehrere Raytracer konfiguriert werden können.

- `params=`*Parmeter*
  Dieser Eintrag gibt an, welche zusätzlichen Parameter an den Raytracer übergeben werden sollen. Die Einstellungen für Anti-Aliasing können zum Beispiel damit gesetzt werden.

  Für bestimmte Versionen von POV-Ray ist es wichtig, dass hier der Parameter `params= -p` angegeben wird (`-p` = pause_when_done). Ansonsten hält POV-Ray nach jedem Bild an und wartet auf einen Tastendruck.

  Meist ist es auch ratsam, die Darstellung von POV-Ray mit `-d` abzuschalten. Insbesondere, wenn der Film mit `anitmt-server` über ein Netzwerk verteilt werden soll. Bei manchen POV-Ray-Versionen ist dies auch Standard.

- `name=`*Projektname*
  Dieser Eintrag gibt an, welcher Name das Projekt haben soll. Dies ist insbesondere im Zusammenhang mit `anitmt-server` und `anitmt-client` von Bedeutung,

Von diesen Einstellungen müssen auf jeden Fall `width`, `height` und `ani_dir` angegeben werden. Des weiteren muss aus `duration`, `starttime`, `endtime`, `fps` und `frames` Angaben so ausgewählt werden, dass sich die Dauer des Filmes und seine Bilderanzahl eindeutig bestimmen läßt.