Animationen

Viele Modelle profitieren sehr davon, wenn sich Teile bei ihnen bewegen. Ob man eine Tür, ein Fenster, ein Drehlicht, vielleicht ein Einziehfahrwerk bei Flugzeugen oder etwas ganz anderes haben möchte, man kommt an Animationen irgendwann nicht mehr vorbei.

Hier ist eine kurze Erklärung, wie man sie schreibt.

1. Einführung, oder "Was ist eine Animation überhaupt?"

Grundsätzlich geht es bei Animationen darum, Teile relativ zu anderen zu verschieben, zu drehen oder zu skalieren. Ein Teil in diesem Sinne kann entweder ein Mesh oder eine Gruppe sein; hier unterscheidet sich die Herangehensweise nur minimal. Um eine Animation für ein Teil zu definieren, muß man dem Spiel zwei Fragen beantworten: "Wann?" und "Wo?". Es geht also darum, dem Spiel zu erklären, wann das zu animierende Teil wo sein soll.

2. Keyframes und Transformationsmatrizen

Um dem Spiel dies zu erklären, gibt es zwei verschiedene prinzipielle Möglichkeiten. Intuitiv am einfachsten ist der Keyframe, mächtiger, aber auch komplexer ist die Transformationsmatrix. Schauen wir uns zunächst einmal einen Keyframe an.

Code

```
{time = 0, rot = { 0,0,0}, transl = {0,0,0} },
```

So sieht der aus, und damit werden die beiden Fragen "Wann?" und "Wo?" beantwortet. "time" legt einen Zeitpunkt ab dem Start der Animation fest, die Einheit hier sind Millisekunden. "rot" definiert die Drehung des Teils um die Z-, Y- und X-Achse (!) in Grad, und "transl" definiert die Verschiebung des Teils relativ zu seiner Ursprungsposition entlang der X-, Y- und Z-Achsen. Warum die Achsen bei der Drehung anders als bei der Verschiebung angeordnet sind, weiß UG wahrscheinlich, ich kann das nicht beantworten.

Eine Transformationsmatrix sieht so aus:

Code

```
transf = {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, },
```

Das sieht etwas unübersichtlicher aus, enthält aber dieselben Informationen wie der Keyframe und dazu noch die Skalierung des Teils um alle Achsen und die Position des Teils. Die Drehung des Teils ist über den Sinus und Cosinus der Drehwinkel angegeben, die Skalierung fließt hier ebenfalls ein - die Berechnung einer Transformationsmatrix ist also am besten einem Computer überlassen und kann mit diesem Werkzeug hier

gut durchgeführt werden. Gut zu wissen ist, daß die letzten vier Stellen der Matrix die X-, Y- , Z- Position und die Skalierung des Teils angeben.

3. Die Bewegung

Ein Keyframe alleine reicht natürlich nicht, damit bewegt sich noch nichts: eine Bewegung ist die Veränderung der Position über die Zeit. Daraus ergibt sich, daß wir mindestens zwei Keyframes oder Matrizen brauchen, um eine Animation zu erzielen.

Wenn wir ein Teil innerhalb von einer Sekunde um einen Meter nach oben schieben wollen, dann würden die Keyframes dazu so aussehen:

Code

```
{time = 0, rot = { 0,0,0}, transl = {0,0,0} },
{time = 1000, rot = { 0,0,0}, transl = {0,0,1} },
```

Zum Zeitpunkt 0 liegt das Teil also ungedreht an seiner Ursprungsposition. Dann setzt es sich sofort in Bewegung, so daß es zum Zeitpunkt 1000ms = 1 Sekunde die Position 0,0,1 erreicht hat, also einen Meter nach oben verschoben ist.

Und daraus ergibt sich alles weitere und man kann extrem komplexe Animationen erstellen, indem man einfach einen Keyframe nach dem nächsten schreibt und damit für immer neue Zeitpunkte die bekannte Frage "Wann?" und "Wo?" beantwortet.

Wenn man das Teil dann in der nächsten Sekunde einen Meter nach links, wieder eine Sekunde später einen Meter nach unten und dann wieder an die Ursprungsposition führen will, dann sieht das zum Beispiel so aus:

Code

```
{time = 0, rot = { 0, 0, 0}, transl = {0, 0, 0} },
{time = 1000, rot = { 0, 0, 0}, transl = {0,-1, 1} },
{time = 2000, rot = { 0, 0, 0}, transl = {0, -1 0} },
{time = 3000, rot = { 0, 0, 0}, transl = {0, 0, 0} },
{time = 4000, rot = { 0, 0, 0}, transl = {0, 0, 0} },
```

Die in "transl=" angegebene Verschiebung bezieht sich immer auf die Ursprungsposition des Teils; man muß also nicht die Verschiebungen aufsummieren oder sonstige komische Exerzitien durchführen.

4. Integration ins Spiel

Eine Animation wird in TPF2 in der .mdl einem Teil umgehängt. Das sieht dann mit der obigen einfachen Animation so aus:

Code

```
{

                                    },
                                },
                            },
                        },

                                transf = { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 5,5, 5, 1,
                    },
```

Alles anzeigen

Schauen wir uns das einmal an. Die beiden Schweifklammern, die die Definition eines Teils in der .mdl umschließen, kennen wir schon - auch hier wird der Block mit einer öffnenden Schweifklammer begonnen. "animations = {" öffnet dann den Block, in dem die Animationen des Teils festgelegt werden. "forever ={" ist der Name der Animation im Spiel, hier gibt es verschiedene festgelegte Auslöser. Zu diesen kommen wir später, wichtig ist hier nur, daß hier dem Spiel gesagt wird, wann die Animation gestartet werden soll. "params = {" und "keyframes = {" sagt dem Spiel, daß jetzt Keyframes kommen, das muß da genau so stehen.

Und dann finden wir schon die beiden Keyframes wieder, mit denen das Teil in einer Sekunde um einen Meter nach oben geschoben wird. Der Block "keyframes = {" wird danach mit einer Schweifklammer geschlossen, mit "origin=..." dem Spiel gesagt, daß es tatsächlich an der Ursprungsposition des Teils beginnen soll, und der "params="-Block auch geschlossen. "type = "KEYFRAME"" erklärt dem Spiel, daß es soeben Keyframes gelesen hat - es gibt hier auch noch andere Methoden, zu denen komme ich aber später. Und dann machen wir auch den Block mit dem Auslöser wieder zu und zum Schluß auch den "Animations = {..."-Block. Die drei Punkte "materials", "mesh" und "transf" kennen wir schon, die sind auch bei nicht animierten Teilen genau so vorhanden.

Zum Schluß kommt noch die schließende Schweifklammer für den ganzen Block, und das war es auch schon.

5. Die Auslöser

Eben habe ich es schon angedeutet: zu einer Animation gehört auch, daß man dem Spiel sagt, wann sie stattfinden soll. Ein Blinklicht sollte immer blinken, das Fahrwerk sollte aber nach dem Einfahren drin bleiben und nicht gleich wieder rauskommen. Oben steht im Beispiel "forever", es gibt aber auch noch viele andere.

Auslöser	bewirkt Animation, ...
forever	immer, wenn das Teil im Spiel ist: die Animation läuft durch und beginnt dann wieder von vorne
open	wenn eine Tür geöffnet wird

Auslöser	bewirkt Animation, ...
close	wenn eine Tür geschlossen wird
drive	wenn das Modell fährt. Die Geschwindigkeit der Animation ist auch abhängig von der Fahrtgeschwindigkeit.
close_doors	wenn alle Türen des Modells geschlossen werden
open_doors	wenn alle Türen des Modells geöffnet werden
open_doors_left oder _right	wenn alle auf der linken oder rechten Seite liegenden Türen geöffnet werden
close_doors_left oder _right	wenn alle auf der linken oder rechten Seite liegenden Türen geschlossen werden
close_wheels	wenn das Fahrwerk eines Flugzeugs einfahren soll
open_wheels	wenn das Fahrwerk eines Flugzeugs ausfahren soll

Diese Auslöser schreibt man dann dort hin, wo im obigen Beispiel "forever" steht.

6. Die Animation mit einer .ani-Datei

Lange Animationen bewirken oft, daß die .mdl sehr unübersichtlich wird. Man kann daher auch die Animation in eine Datei auslagern, wenn man das lieber möchte.

Der Block in der .mdl sieht dann etwas anders aus:

Code

```
            },
        },
    },
```

Wir sehen hier eine Animation, die zum Einfahren des Fahrwerks dient, denn sie ist mit dem Auslöser "close_wheels" verknüpft. Unter "params = " steht hier kein Keyframe, sondern ein Verweis auf eine Datei unter "id = ", die im Verzeichnis res\models\animation\ gesucht wird. Hier heißt sie "animation.ani"; sie kann nach Belieben benannt werden - der Name muß nur im Spiel einmalig sein.

Schauen wir uns jetzt einmal eine solche .ani an.

Code

```
function data()
return {
                        times = {

                { 1, 0, 0, 0, 0, 0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 1, },
                { 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, },
    },
}
end
```

Wir erkennen hier die oben erklärten Transformationsmatrizen wieder. Unter "times=" steht eine Liste von Zeitpunkten, wieder in Millisekunden, womit die Frage "Wann?" beantwortet ist. Für die Frage "Wo?" arbeitet das Spiel einfach der Reihe nach die Matrizen ab: die erste zum ersten Zeitpunkt, die zweite zum zweiten und so weiter. Wenn wir uns jetzt daran erinnern, daß die vier letzten Stellen in der Matrize die XYZ-Koordinaten und die Skalierung beinhalten, zeigt sich, daß diese .ani genau dasselbe macht wie die vorherigen beiden Keyframes: sie schiebt das Teil innerhalb einer Sekunde um einen Meter nach oben.

Alternativ kann man die .ani auch mit etwas veränderten Keyframes bestücken. Dann sieht dieselbe Animation so aus:

Code

```
local vec3 = require "vec3"
local transf = require "transf"
function data()
return {
                        times = {

transf.rotZYXTransl(transf.degToRad(0, 0, 0), vec3.new(0, 0, 0)),
transf.rotZYXTransl(transf.degToRad(0, 0, 0), vec3.new(0, 0, 1)),
    },
}
end
```

Alles anzeigen

Wenn man diesen Weg gehen möchte, dann sind die beiden Zeilen "local ..." ganz zu Anfang entscheidend: sie erklären dem Spiel, wo sie die beiden Funktionen "vec3" und "transf" suchen müssen. Ohne die Zeilen gibt es einen Crash. Und in den beiden Zeilen, die mit transf.rotZYXtransl beginnen, finden wir dieselben Daten wieder, die auch in einem klassischen Keyframe drin stehen: erst die Rotation um die Z-, Y- und X-Achsen und dann die Verschiebung in Metern entlang der X-, Y- und Z-Achsen.

7. Drehungen

Wenn man ein Teil zum Drehen bekommen möchte, dann bietet es sich an, die Animation sinnvoll zu zerteilen. Was nicht funktioniert, ist, dem Spiel einfach die Keyframes

Code

```
{time = 0, rot = { 0,0,0}, transl = {0,0,0} },
{time = 2000, rot = { 0,180,0}, transl = {0,0,0} },
{time = 4000, rot = { 0,0,0}, transl = {0,0,0} },
```

hinzuschreiben: damit ist nur festgelegt, daß das Teil zum Zeitpunkt 2000 um 180 Grad um die Y-Achse gedreht und bei 4000 wieder in der Ursprungsposition stehen soll. Aber in welche Richtung das passieren soll, ist damit nicht definiert. Wenn man das Teil innerhalb von 4 Sekunden einmal im Kreis um die Y-Achse drehen möchte, dann muß man die Animation in Schritten schreiben. Das sähe so aus:

Code

```
{time     =     0,     rot     =     {     0,0,0},     transl     =     {0,0,0}     },
{time     =     1000,     rot     =     {     0,90,0},     transl     =     {0,0,0}     },
{time     =     2000,     rot     =     {     0,180,0},     transl     =     {0,0,0}     },
{time     =     3000,     rot     =     {     0,270,0},     transl     =     {0,0,0}     },
{time = 4000, rot = { 0,360,0}, transl = {0,0,0} },
```

So gibt es eine eindeutige Drehrichtung und das Teil bewegt sich flüssig innerhalb von 4 Sekunden im Kreis.

In TPF2 ist hier noch etwas zu beachten. Diese Keyframes werden vom Spiel relativ brutal in Matrizen umgerechnet, was zu Problemen mit der Skalierung führt: das Teil hat jeweils zum im Keyframe festgelegten Zeitpunkt seine originale Skalierung, während es sich aber zwischen ihnen bewegt, wird es verzerrt. Das sieht nicht gut aus, ist aber einfach zu minimieren. Man zerlegt die Rotation einfach in noch mehr kleine Segmente: nicht wie oben alle 1000 Millisekunden 90 Grad, sondern zum Beispiel alle 100ms 9 Grad. Das erhöht die Zahl der Keyframes zwar massiv, sieht aber im Spiel wesentlich besser aus.

8. Nützliche Kleinigkeiten

Manchmal möchte man einem Teil mehr als eine Animation mitgeben. Zum Beispiel könnte man einen Propeller haben wollen, der wie beim Schienenzeppelin während der Fahrt lastabhängig, bei Stillstand aber im Leerlauf dreht - auch das ist kein Problem. In dem Fall würde man im Block "animations=" eine Animation "forever=" und eine weitere "drive = " definieren. Das Spiel benutzt dann beide; wenn das Fahrzeug stillsteht, dann läuft nur "forever" und "drive" kommt dann dazu, wenn es losfährt.

Es gibt Situationen, in denen man ein Teil kurz stillegen möchte. Zum Beispiel die Fahrwerksklappen eines Flugzeugs gehen auf, bleiben dann offen, während die Räder in die gewünschte Position fahren, und schließen sich dann wieder. Das würde in Keyframes so aussehen:

Code

```
{time     =     0,     rot     =     {     0,0,0},     transl     =     {0,0,0}     },
{time     =     1000,     rot     =     {     0,0,90},     transl     =     {0,0,0}     },
{time     =     5000,     rot     =     {     0,0,90},     transl     =     {0,0,0}     },
{time = 6000, rot = { 0,0,0}, transl = {0,0,0} },
```

Dann fährt die Tür von 0-1000ms auf und bleibt bis 5000ms offen. Zwischen 1000 und 5000 würde dann das Fahrwerksbein ein-oder ausfahren, und von 5000 bis 6000ms geht die Tür wieder zu.

Hier muß man einfach ein wenig herumprobieren.

Viel Erfolg!