Open Gl - Delphi – Kurs Teil 6: Alpha – Kanal, Blending, Multitexturring

Einleitung

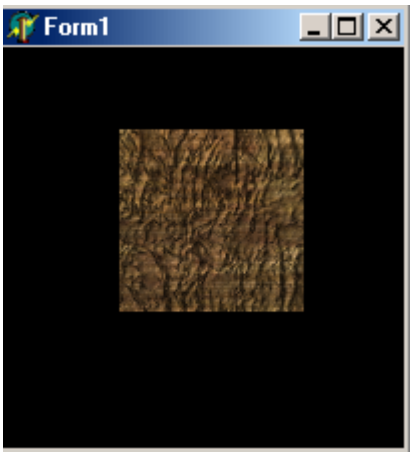
Die Wochen vergehen ja wie im Fluge...
Diesmal wollen wir uns also hauptsächlich mit dem Blending befassen: eine der besten Techniken um für mehr Atmosphäre in euren Open Gl – Szenen zu sorgen...

Die Theorie des Blending

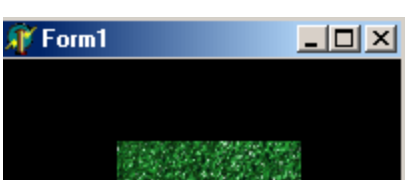
Womit wir also schon gleich beim Thema wären: Was genau ist eigentlich „blending“? Nun ja: an sich ist bedeutet blenden, das man etwas zeichnet, dabei aber die Farben des Dahinterliegenden Objektes mit berücksichtigt...
Wie aber macht das denn Open GL? Nun ja... wohl kaum, indem ich mit bösem Blicke und mit viel Magie vorm Rechner sitze und ihn mit satanischem Gebrabbel dichtlaber, bis er irgendwann es denn auch tut... (auch wenn viele Leute denken, Programmieren ginge so.. *lol*)
Scherz beiseite: in Open GL wird das ganze sehr einfach realisiert: Man hat ja den „Framebuffer“, in welchen man primär reinzeichnet und der denn mit „Swapbuffers“ auf den Screen gebracht wird. Hier setzt das Blending an: alle Objekte, die nicht transparent sein sollen (besser: die nicht geblendet werden sollen), werden zuerst gezeichnet. (Sie sind dann praktisch fertig im Framebuffer drin) Danach wird das Blending aktiviert und die Transparenten Sachen gezeichnet, wobei man natürlich mit angeben muss, wie viel von dem bereits gerendertem übernommen und wie viel Einfluss das (halb) transparente Objekt haben soll.

Die Praxis des Blending

So Leute, lasst uns das ganze in die Praxis umsetzen... wie ihr so ein Objekt darstellen könnt, ist euch sicher klar:



(Wenn nein: schielt einfach mal ins 5. Kapitel ;))
Und so was dürfte genau so wenig Schwierigkeiten bereiten:



(Fällt euch was auf: dasselbe Viereck nur mit einer anderen Textur)
Na ja... ich persönlich finde beide nicht gerade sehr ansprechen... aber so sieht das doch recht gut aus, oder?

Huch, was ist das denn nun? Nun ja: ich habe mit Hilfe des Blendings beide Texturen genommen, sie in einen Topf geschmissen, gut durchgerührt und das ganze dargestellt. Sieht doch aus, wie aus einem Guss, oder?
Ich gebe euch nun erst mal einfach den Code für das ganze... daran kann ich es leichter erläutern, was genau ich gemacht habe.

```
glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT); //Farb und
Tiefenpuffer löschen
glLoadIdentity;

glBindTexture(GL_TEXTURE_2D, tex);   //Erste Textur, für das Urobjekt
glbegin(gl_quads);
gltexcoord2f(0,0);
glvertex3f(-0.5,-0.5,-3);
gltexcoord2f(0,1);
glvertex3f(-0.5, 0.5,-3);
gltexcoord2f(1,1);
glvertex3f( 0.5, 0.5,-3);
gltexcoord2f(1,0);
glvertex3f( 0.5,-0.5,-3);
glend;

glenable(gl_blend);                          //Aktivieren des Blending
gldisable(gl_depth_test);
glblendfunc(gl_src_alpha, gl_one);

glcolor4f(1,1,1,0.5);
glBindTexture(GL_TEXTURE_2D, tex2);     //Zweite Textur für das zweite Objekt
glbegin(gl_quads);
gltexcoord2f(0,0);
glvertex3f(-0.5,-0.5,-3);
gltexcoord2f(0,1);
glvertex3f(-0.5, 0.5,-3);
```

```
gltexcoord2f(1,1);
glvertex3f( 0.5, 0.5,-3);
gltexcoord2f(1,0);
glvertex3f( 0.5,-0.5,-3);
glend;
glenable(gl_depth_test);
gldisable(gl_blend);
SwapBuffers(form1.myDC);                    //Szene ausgeben
end;
```

Ok... denn will ich mal loslegen. Also am Anfang habe ich erst mal was gemacht, was keinen Großartig erstaunen dürfte... ich hab einfach ein Quadrat gezeichnet, auf welches die grüne Textur gespannt ist.

Die nächsten 3 Zeilen muss ich aber erklären:

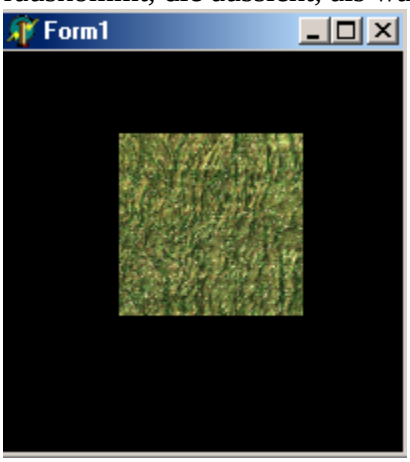
Zunächst habe ich mit „glenable(gl_blend)“ das Open Gl – Blending aktiviert. Damit wusste Open Gl, dass alles bisherige praktisch halb im Framebuffer „überschrieben“ werden kann... besser: Damit habe ich die Grundlage für jegliche Transparenz, usw. erschaffen.
Daraufhin habe ich den Tiefenpuffer deaktiviert... das Motiv dafür ist irgendwie auch logisch: wenn ich den nämlich anlasse, dann wird ja weiterhin geprüft, ob nicht irgendein Objekt davor ist. Das ist beim blenden aber sehr unpraktisch. Wenn ich nämlich ein Objekt an dieselbe Stelle zeichnen möchte, wo bereits eins ist, dann denkt, Open Gl ja, das dort bereits eins sei und zeichnet daher mein neues nicht... wäre in diesem Falle fatal.
Danach müssen wir mit „glblendfunc“ angeben, wie viel von dem neuen und wie viel von dem alten übernommen wird. Der erste Parameter steht dabei für das neue und der zweite für das, was bereits im Framebuffer drin ist. „Gl_one“ bedeutet dann soviel wie, dass man „alles“ von dem einen verwendet... „gl_zero“ bewirkt das Gegenteil. Insgesamt sind diese Angaben möglich:

Angabe	**Bedeutung**
Gl_one	Von der entsprechenden Seite wird jeweils „alles“ verwendet.
Gl_Zero	Von der entsprechenden Seite wird jeweils „nichts“ verwendet.
GL_DST_COLOR	Von der entsprechenden Seite wird jeweils soviel genommen, wie „stark“ die Farbe im Framebuffer ist. Dabei kommt es logischerweise auch zu Verfärbungen des neuen Transparenten Objektes, da alle Farbwerte einzeln betrachtet werden.
GL_SRC_COLOR	Von der entsprechenden Seite wird jeweils soviel genommen, wie „stark“ die Farbe des transparenten Objektes ist. Auch hier werden alle Farbwerte einzeln betrachtet.
GL_ONE_MINUS_DST_COLOR	Die genaue Umkehrung zu GL_DST_COLOR
GL_ONE_MINUS_SRC_COLOR	Die genaue Umkehrung zu GL_SRC_COLOR
GL_SRC_ALPHA	Der Hauptbestandteil des Alpha-Blendings: Die jeweilige Seite wird jeweils so stark beachtet, wie Transparent das transparente Objekt ist... (irgendwie logisch, oder?)
GL_DST_ALPHA	Dasselbe wir Gl_SRC_ALPHA, bloß werden hier die

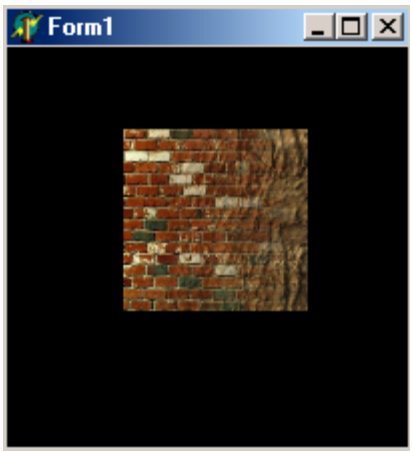
	Objekte im Framebuffer betrachtet
GL_ONE_MINUS_SRC_ALPHA	Na ja: das Gegenteil zu GL_SRC_ALPHA
GL_ONE_MINUS_DST_ALPHA	Muss ich das kommentieren?

Zugegeben, das war nun etwas verwirrend... man braucht etwas, um das wirklich zu verstehen...
Auf jeden Falle dürfte nun unsere Zeile „glblendfunc(gl_src_alpha, gl_one);“ einigermaßen klar sein: es soll alles im Framebuffer beachtet werden (gl_one) und das neue Objekt je nach seiner Transparenz drübergezeichnet werden. Damit haben wir ja den klassischen Transparenzfall, der für solche Aufgaben auch zu empfehlen ist.

Kommen wir also zur nächsten Zeile:
“glcolor4f(1,1,1,0.5);” Das ist nun unser eigentliches Kernstück. Wie ihr seht, handelt es sich um eine Normale Color - Zuweisung, die aber nun einen Parameter mehr verlangt „4f“. dieser vierte Parameter ist der Alpha-Wert oder auch Transparenz genannt. Wenn man mit glcolor3f eine Farbe zuweist, wird dieser immer auf „1“ gesetzt, was bedeutet, das dieses Objekt nicht transparent ist. (Ist logisch, das „0“ dementsprechend für volltransparent steht) Wenn wir also nun den Transparenzwert „0.5“ mit „glblendfunc(gl_src_alpha, gl_one);“ mixen bedeutet das im Klartext, das wir beide Teile gleich stark gewichten, so das unsere nette Mischtextur rauskommt, die aussieht, als wäre sie aus einem Guss:



So... damit wir noch mal ein wenig Übung sammeln, frage ich euch, wie man denn so was erreichen kann:



Nun ja: ich würde sagen, das es recht einfach ist: ich habe hier wie im ersten Beispiel zunächst die Felstextur auf ein Quadrat gezeichnet und nun einfach eine Steintextur drüber

gekleistert... der Trick bei der Sache ist, dass ich ja verschiedenen Ecken unterschiedliche Alpha oder Colorwerte zuweisen kann. Damit ist der Aufbau dieser Szene recht einfach:

```
glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT); //Farb und
Tiefenpuffer löschen
glLoadIdentity;

glBindTexture(GL_TEXTURE_2D, tex); //Felstextur
glbegin(gl_quads);
gltexcoord2f(0,0);
glvertex3f(-0.5,-0.5,-3);
gltexcoord2f(0,1);
glvertex3f(-0.5, 0.5,-3);
gltexcoord2f(1,1);
glvertex3f( 0.5, 0.5,-3);
gltexcoord2f(1,0);
glvertex3f( 0.5,-0.5,-3);
glend;

glenable(gl_blend);
gldisable(gl_depth_test);
glblendfunc(gl_src_alpha, gl_one_minus_src_alpha);


glBindTexture(GL_TEXTURE_2D, tex2);  //Steintextur
glbegin(gl_quads);
glcolor4f(1,1,1,1);
gltexcoord2f(0,0);
glvertex3f(-0.5,-0.5,-3);

glcolor4f(1,1,1,1);
gltexcoord2f(0,1);
glvertex3f(-0.5, 0.5,-3);

glcolor4f(1,1,1,0);
gltexcoord2f(1,1);
glvertex3f( 0.5, 0.5,-3);

glcolor4f(1,1,1,0);
gltexcoord2f(1,0);
glvertex3f( 0.5,-0.5,-3);
glend;

glenable(gl_depth_test);
gldisable(gl_blend);
SwapBuffers(form1.myDC);
```

So... damit hätten wir nun zweit Texturen ineinander überlaufen lassen (ihr könnt es ja mal mit mehreren versuchen)...

Sicherlich ist euch nun klar, das man mit dem Blenden eine menge Effekte machen kann... so werde ich euch in einem der Zukünftigen Teilen erklären, wie man z.B. mit Hilfe des Blendings einige einfache Beleuchtungseffekte kreiert (ich werde das ganz simpel halten) ...

Jojo… damit wären wir zunächst mit dem Blenden bzw. Alpha-Blenden durch. Ich hoffe, es hat euch Spaß gemacht und wir sehen uns in zwei Wochen wieder. Ich bin mir noch nicht ganz sicher, was ich im nächsten Teil genau bringe, aber so wie es aussieht, wird es ein Part, welcher verschiedene kleine Dinge in Open GL, wie z.b. Kameraführung, usw. enthält. (kann aber auch ein Part mit praktischen Übungen werden... bin mir noch nicht sicher...)

Mr_T