

## Kleine Objektkunde

Bevor wir uns ans Erstellen machen, erzähle ich Euch ein bisschen über die Beschaffenheit von Sims-Objekten, damit Ihr wisst, wieso wir nachher das tun, was wir tun ;-). Außerdem ist das eine prima Gelegenheit, Euch mit den gängigen Begriffen vertraut zu machen, die uns später noch sehr oft begegnen werden :-).

In Die Sims gibt es vier Blickwinkel, **NW** (North West = Nordwest), **NE** (North East = Nordost), **SE** (South East = Südost) und **SW** (South West = Südwest) sowie drei **Zooms**, also Ansichtsgrößen (groß, mittel, klein).

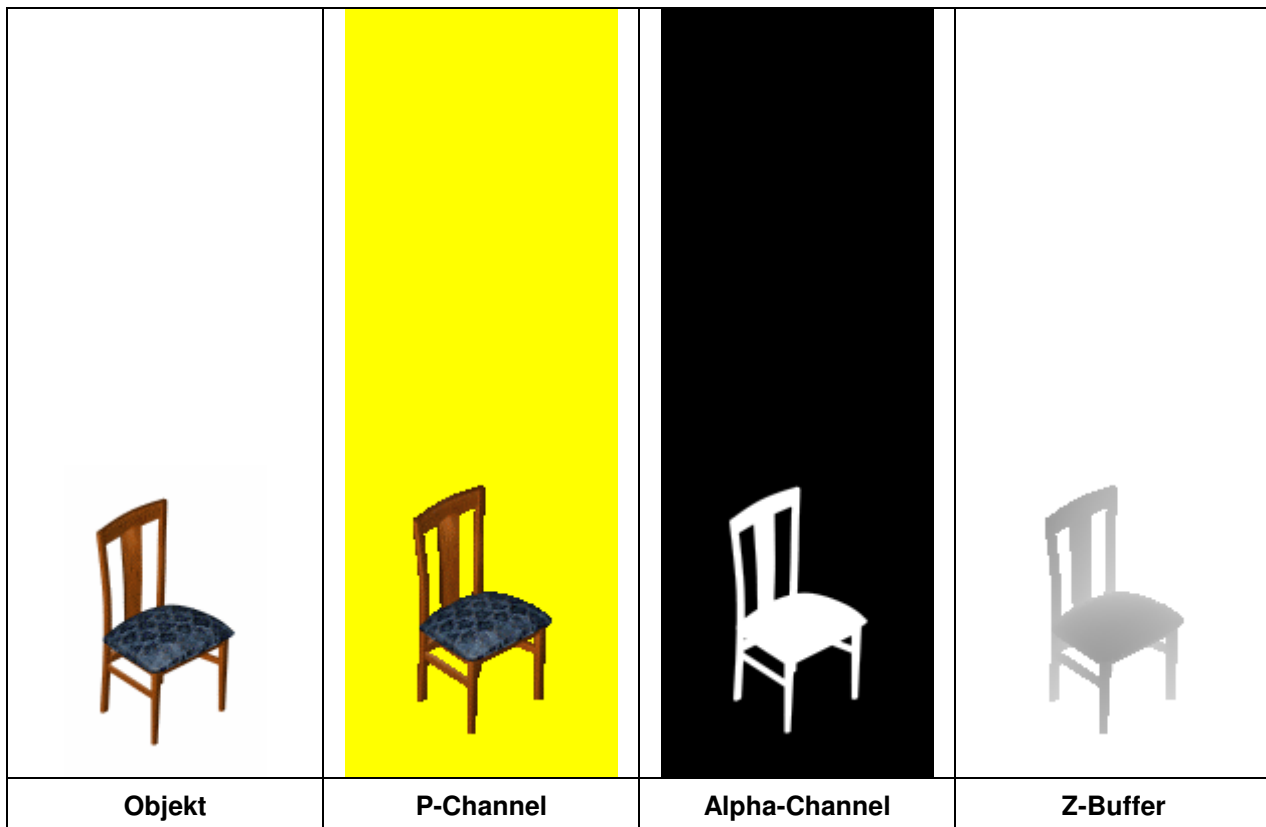
Da Objekte in Wirklichkeit 2-dimensionale Bilder sind, die nur so tun, als wären sie 3-dimensional, bestehen sie aus Einzelbildern, grundsätzlich für jeden Blickwinkel und jeden Zoom eines – also aus 12 Bildern, und das pro Rasterfeld (**Tile**), das sie im Spiel einnehmen (z.B. nimmt ein Stuhl oder Beistelltisch nur 1 Feld ein (**Single Tile Objekt**), ein Sofa oder ein Bett mehrere Felder, hier spricht man dann von einem **Multi Tile Objekt**).

Die Anzahl der Einzelbilder kann variieren – bei einer Vase, die von allen Seiten gleich aussieht, ist es z.B. nicht nötig, für jeden Blickwinkel ein eigenes Bild zu erstellen, hier gibt es nur ein Bild pro Zoom. Bei einem Stuhl gibt es pro Zoom nur zwei Bilder, eine Vorder- und eine Rückansicht – die werden im Spiel bei Bedarf einfach gespiegelt.

Andere Objekte dagegen haben nicht nur vier Ansichten und bestehen aus mehreren Feldern, sie haben überdies auch noch mehrere optische Zustände – ein Doppelbett zum Beispiel. Hier haben wir die Zustände „made“ (gemacht), „ready“ (aufgedeckt), „sleep“ (schlafen) und „messy“ (ungemacht) und sind damit schon bei insgesamt 288 Einzelbildern (4 Ansichten x 4 Zustände x 6 Tiles x 3 Zooms), von einem Liebesbett ganz zu schweigen – das Fieseste, was einem passieren kann ;-). Hab ich schon erwähnt, dass noch zusätzliche Bilder für die Kissen dazukommen? \*g\*.

Bevor Ihr jetzt die Krise kriegt: In aller Regel bearbeitet man nur den größten Zoom, insbesondere bei Single Tile Objekten – Betten sind auch nicht gerade die Art von Objekten, mit denen man anfängt. Also keine Bange – wir lassen es ruhig angehen :-).

Aber weiter im Text – die Einzelbilder bestehen aus drei „Schichten“, das sind die sogenannten **Sprites**. Jeder dieser Sprites ist für einen **Channel** (=Kanal) zuständig, den Farbchannel oder **P-Channel** (ich weiß nicht genau, wofür das „P“ steht, ich vermute aber, es kommt von „Picture“ (=Bild), den **Alpha-Channel** und den **Z-Buffer**. Diese Sprites sind es, die wir bei der Objekterstellung bearbeiten, und so sehen sie aus:



## Der P-Channel

Der P-Channel ist für die Farbgebung zuständig. Damit ist nicht nur der Farbton gemeint, hier kann man auch die Schattierung, Struktur und dergl. ändern. Z.B. könnte man nur durch Bearbeitung des P-Channels sowas hier machen:



Hier habe ich den Stuhl neu bezogen, umgestrichen und eine fette Spinne draufgesetzt – nur durch Änderung des P-Channels. Mit anderen Worten, ich habe alles mögliche mit dem Stuhl getrieben, was sich *innerhalb seiner ursprünglichen Form* anstellen ließ.

## Der Alpha-Channel

Der Alpha-Channel gibt dem Objekt seine Form – wenn Ihr den Alpha-Channel oben anschaut, seht Ihr, dass die Fläche, die der Stuhl einnimmt, weiß ist, alles andere ist schwarz. Im Spiel werden alle weißen Bereiche im Alpha-Channel massiv angezeigt (keine Transparenz), alle schwarzen sind unsichtbar (100% Transparenz).

Wenn ich nun den Bereich der Stuhllehne schwarz ausfülle, kommt das hier dabei heraus:



Nicht schön, aber ich denke, Ihr seht, worauf ich hinauswill :-).

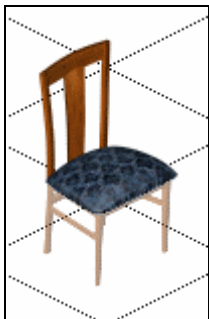
Eine weitere, sehr wichtige Funktion des Alpha-Channels ist die Kantenglättung – seht Euch einmal den Stuhl auf dem P-Channel und den Stuhl im Spiel im Vergleich an:



Wie Ihr seht, ist der Stuhl auf dem P-Channel sehr eckig, im Spiel sind die Konturen jedoch weich. Das kommt daher, dass der Alpha-Channel eben doch nicht nur schwarz und weiß ist – in der Vergrößerung kann man deutlich die verschiedenen Grau-Abstufungen erkennen:



Je dunkler der Grauton, desto durchsichtiger erscheint der Bereich, so entsteht der Eindruck, das Objekt hätte an den Kanten keine Pixel-Treppen, sondern einen weichen Verlauf. Durch die Graustufen lassen sich z.B. auch halbdurchsichtige Fenster, Glastische oder durchscheinende Gardinen erstellen, oder meinetwegen auch transparente Stuhlbeine :o):



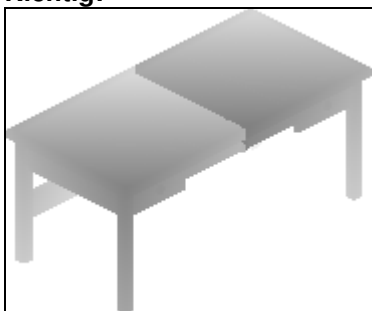
### Der Z-Buffer

Der wichtigste der drei Channel, denn ohne ihn wären die meisten Objekte im Spiel unbrauchbar. Wir erinnern uns: Objekte sind zweidimensional. Dass es einem solchen flachen Ding überhaupt möglich ist, Räumlichkeit vorzugaukeln (wenn ein Sim sich in einen Sessel setzt, bedeckt er die Sitzfläche, aber die Armlehne bedeckt ihn), ist dem Z-Buffer zu verdanken.

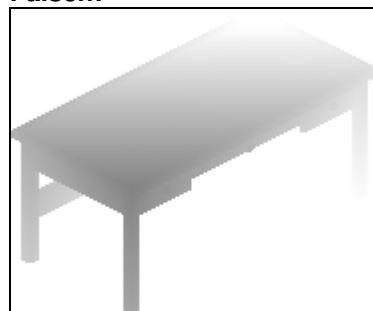
Der Name kommt daher, weil er die Lage des Objektes entlang der Z-Achse bestimmt, das ist die Achse, die in den Bildschirm hinein- bzw. aus dem Bildschirm heraus auf Euch zu verläuft (mit anderen Worten, er bestimmt die Tiefe). Anhand der Grauwerte des Z-Buffers kann das Spiel errechnen, ob ein Teil sichtbar ist oder verdeckt wird – je dunkler das Grau, desto weiter vorne liegt der Bereich. Beim eben genannten Sessel heißt das also, die Sitzfläche, die vom Sim verdeckt werden muss, ist heller, die Armlehne, die ihrerseits den Sim verdeckt, ist dunkler.

Der Einfluss des Z-Buffers ist beschränkt auf das Tile, zu dem der Buffer gehört. Wenn Ihr z.B. einen Tisch habt, der sich über zwei Felder oder Tiles erstreckt, ist der Z-Buffer des hinteren Teils nicht heller als der des vorderen Teils, obwohl man das meinen könnte:

**Richtig:**



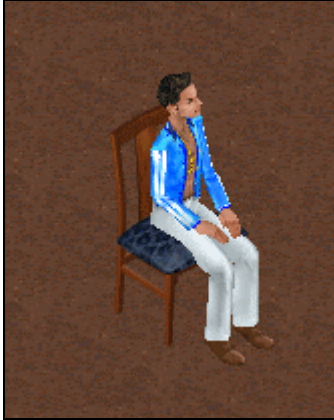
**Falsch:**



An dem falschen Beispiel sieht Ihr schon, dass es so gar nicht machbar wäre – wie sollte der Z-Buffer bei einem 3-teiligen Objekt aussehen? Heller als weiß geht halt net ;-).

Hier noch ein Beispiel für einen Z-Buffer, der danebenging:

**Richtig:**



**Falsch:**



Ihr könnt mir doch bestimmt sagen, was mit dem Z-Buffer nicht stimmt – ist er zu hell oder zu dunkel ;-)?

Zum Schluss dieser kleinen Exkursion noch ein sehr wichtiger Punkt, der anfangs gern übersehen wird und viel Kopfzerbrechen auslöst, warum zum Donnerwetter das doofe Objekt im Spiel nicht anständig angezeigt wird:

## Farbtiefen oder wie speichere ich meine Sprites richtig?

### P-Sprites:

Bitmap (.bmp) in **256 Farben** (8Bit)

Fläche außerhalb des Objekts: Rein Blau (0,0,255\*), Gelb (0,255,0\*) oder Rot (255,0,0\*)

Ich empfehle Gelb, weil es eh Standard ist und man so nichts ändern muss, und außerdem ist es, obwohl grausam genug, auf die Dauer nicht ganz so anstrengend für die Augen wie das reine Blau oder Rot.

### Alpha-Channel:

Bitmap (.bmp) in **Graustufen**

Fläche außerhalb des Objekts: Rein Schwarz (0,0,0\*)

### Z-Buffer:

Bitmap (.bmp) in **Graustufen**

Fläche außerhalb des Objekts: Rein Weiß (255,255,255\*)

### Sprite-Format (Pixel):

Groß: 136 x 384

Mittel: 68 x 192

Klein: 34 x 96

\* RGB-Farbwerte