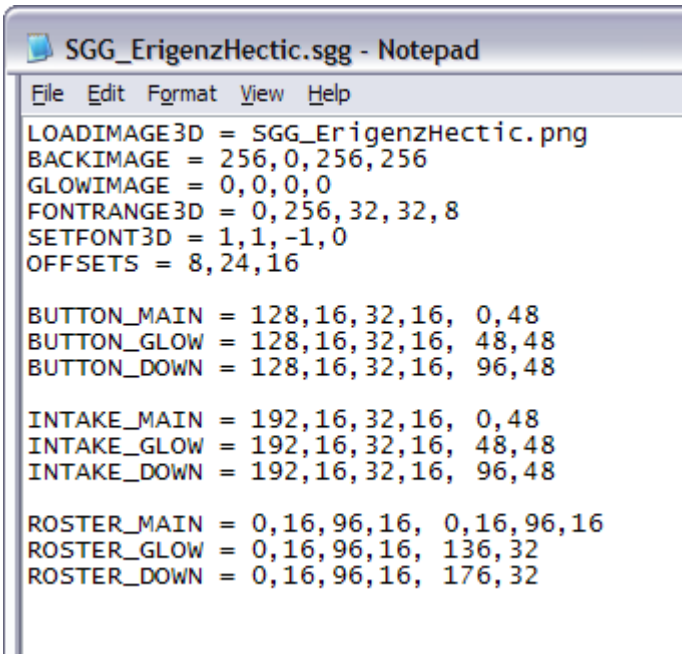
Die SGG *(Simple Game GUI)* ist eine einfach gehaltene GUI für die Draw3D2. Die SGG setzt, anders als die meisten anderen GUIs, nicht auf Fenstersteuerung, sondern ausschliessliche auf die wenigen Teilbereiche die man in Computerspielen benötigt ohne Fenster/WinAPI-Like.

Das wären:

Bekannter GUI-Name	**Gegebener SGG-Name**	**Beschreibung**
Label	OUTLAY	Bezeichner
Button	BUTTON	Schaltfläche (Taster)
n.v.	BUTTON	Schaltfläche (Schalter)
	INTAKE	Eingabefeld
RadioButton	ROSTER	Auswahlfläche
ListBox	ROSTER	Auflistung

Die SGG ist skinbar und benötigt zu jedem Skin eine Textdatei mit einen INI-ähnlichem Aufbau, und eine zugehörige Textur mit Bildelementen und den eigendlichen Skindesign.

So eine SGG-Datei hat als Dateiendung .sgg und sieht in etwa wie folgt aus:



Im ersten Teil werden allgemeine Informationen eingestellt und die Schrift auf der Textur deklariert. In den drei weiteren Teilen werden die Spezifikationen der einzelnen Funktionen spezifiziert.

```
LOADIMAGE3D = SGG_ErigenzHectic.png
```

Die zu ladene Textur.

```
BACKIMAGE = 256,0,256,256
```

Das Hintergundbild, welches sich übr ein ganzes SGG-Hauptelement erstreckt.

```
GLOWIMAGE = 0,0,0,0
```

Das Hintergrundbild, welches sich über dem `BACKIMAGE` erstreckt, wenn sich die Maus über ein SGG-Hauptelement befindet. Es werden beide Bilder übereinander gezeichnet!

```
FONTRANGE3D = 0,256,32,32,8
```

Diese Anweisung wurde 1:1 aus der Draw3D2 übernommen:
- X·Startposition wo die Schriftart auf der Textur beginnt
- Y·Startposition wo die Schriftart auf der Textur beginnt - von **Chr$(0)** aus gesehen
- Die maximale Breite eines jeden Schriftzeichens
- Die maximale Höhe eines jeden Schriftzeichens
- Die Anzahl Reihen verfügbarer Schriftzeichen

```
SETFONT3D = 1,1,-1,0
```

Diese Anweisung wurde 1:1 aus der Draw3D2 übernommen:
- Size / Gesamtskallierungs-Faktor der Schriftart
- Height / Höhenskallierungs-Faktor der Schrifart
- Padding / Additiver Zeichenabstand
- Italic / Additive Schriftneigung

```
OFFSETS = 8,24,16
```

Hier können noch restliche Sachen gemacht werden:
- X·Abstand der Schriftzeichen zwischen Zeichen und Außenwand (optische Verschönerung)
- Y·Skallierung des Hintergrudes von BUTTON und INTAKE (sollte die Hälfte der gegrabbten höhe aus jeweils BUTTON und INTAKE dessen letzten Parameter haben)
- Additver Y·Abstand beim ROSTER zwischen ersten und letzten Zeile zum Randbereich

```
BUTTON_MAIN = 128,16,32,16, 0,48
```

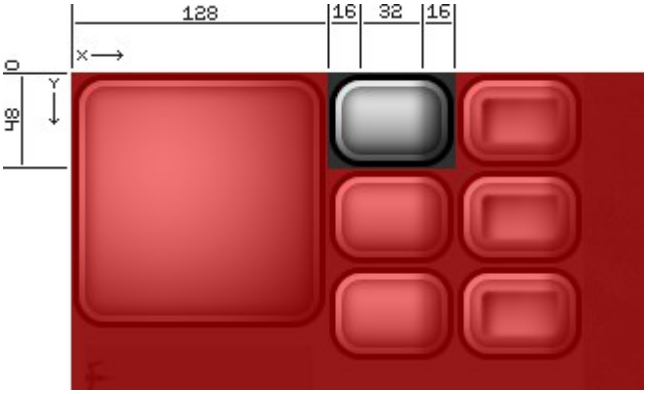
Grabben der einzelnen Bildelemente aus der zu ladenen Textur:
- X·Startbereich, wo die Teilabschnitte gegrabbt werden sollen
- X·Breite des ersten Teilabschnittes (linker Rahmen vom BUTTON)
- X·Breite des zweiten Teilabschnittes (gestreckter Bereich vom BUTTON)
- X·Breite des dritten Teilanschnittes (rechter Rahmen vom BUTTON)
- Y·Startbereich, wo die Teilabschnitte gebrabbt werden sollen
- Y·Höhe des BUTTONs der gegrabbt werden soll

Die X,X,X,X, Y,Y -Skallierung muß sein, damit Fehler im Vorwege vermieden werden. Damit wird auch möglich, dass ein Grafikelement wie BUTTON, INTAKE oder ROSTER mit weniger Vertices gezeichnet werden kann. Außerdem können so keine Artefakte auftretten.

Beispielsweise ein BUTTON:

Besteht aus drei Teilabschnitten:

Diese müssen nun aus der Textur effizient und Fehlersicher gegrabbt werden:



Das sind also die Werte --> `128,16,32,16, 0,48`

```
BUTTON_GLOW = 128,16,32,16, 48,48
```

wird mit der gleichen Methode gegrabbt wie BUTTON. Dieser Teilabschnitt wird über den BUTTON gezeichnet, wenn sich die Maus da drüber befindet.

```
BUTTON_DOWN = 128,16,32,16, 96,48
```

wird mit der gleichen Methode gegrabbt wie BUTTON. Dieser Teilabschnitt wird über den BUTTON gezeichnet, wenn die Maus den BUTTON drückt.

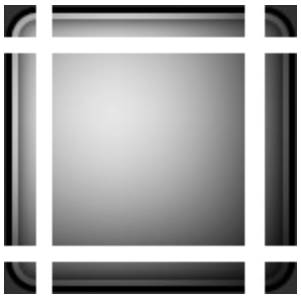
```
INTAKE_MAIN = 192,16,32,16, 0,48
INTAKE_GLOW = 192,16,32,16, 48,48
INTAKE_DOWN = 192,16,32,16, 96,48
```

Ein INTAKE ist ein Eingabefeld und wird auch aus drei Teilabschnitten gezeichnet.
Zum Grabben siehe die Methode aus BUTTON.

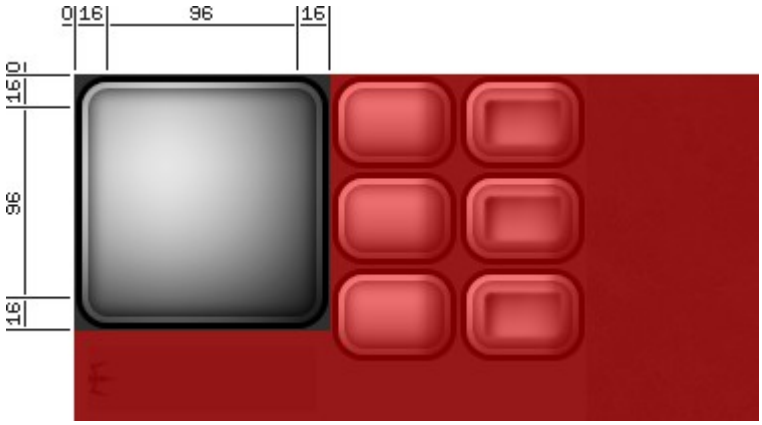
```
ROSTER_MAIN = 0,16,96,16, 0,16,96,16
```

Da ein ROSTER sowohl eine variable Breite als auch eine variable Höhe haben kann, muß ein ROSTER mit Rahmen aus mindestens neun Teilabschnitten bestehen. Diese werden über acht Parameter angegeben.

Ein ROSTER besteht aus neun Teilabschnitten:



Diese werden ähnlich wie beim BUTTON oder INTAKE gegrabbt:



Das sind also die Werte --> `0,16,96,16, 0,16,96,16`

```
ROSTER_GLOW = 0,16,96,16, 136,32
ROSTER_DOWN = 0,16,96,16, 176,32
```

Da ein ROSTER nicht komplett 'aufleuchtet' wenn sich die Maus drüber befindet, oder diese da drauf gedrückt wird, sondern Zeilenweise, so muß auch entsprechend nur dieser Teilabschnitt gegrabbt und vom ROSTER bearbeitet werden.

So ein 'Aufleuchten' besteht aus drei Teilabschnitten und wird so gegrabbt, wie man es vom BUTTON und INTAKE her kennt. Diese Teilabschnitte werden ggf. Dann über dem ROSTER drüber gezeichnet.

Ein markierter ROSTER-Bereich sieht dann zum Beispiel wie folgt aus:

SGG-Befehlsreferenz

CreateSGG (Xpos, Ypos, Xran, Yran, File$, Pivot, Order)

Erstellt ein SGG-Hauptelement. Erst in diesem können weitere Nebenelemente erstellt werden. Untergeordnete Nebenelemente sind ein OUTLAY, BUTTON, INTAKE und ROSTER.

AddingSGG (Handle, Arts, Xpos, Ypos, String$, Align, Xran)

Fügt ein Nebenelement zu einem Hauptelement hinzu. Da es verschiedene Nebenelemente gibt, wird die Art des Nebenelementes beim zweiten Parameter angegeben.

- Das Handle (aus **CreateSGG**) zu welchem Hauptelement es hinzugefpgt werden soll.
- Arts = 1 = OUTLAY = festgelegte Konstante SGGOUTLAY
- Arts = 2 = BUTTON = festgelegte Konstante SGGBUTTON
- Arts = 3 = INTAKE = festgelegte Konstante SGGINTAKE
- Arts = 4 = ROSTER = festgelegte Konstante SGGROSTER

Innerhalb eines Hauptelementes besteht ein eigenes Koordinatensystem, da ein Hauptelement umpositioniert werden kann, müssen ja alle Nebenelemente mit ziehen.

- X·Position des Nebenelement innerhalb des Hauptelementes
- Y·Position des Nebenelement innerhalb des Hauptelementes
- String gibt den Capture an, welches als Text für das Nebenelement gilt
- Align ist die Textausrichtung (0=Links, 1=Zentriert, 2=Rechts)
- Xran ist die Breite eines Nebenelementes (bei einem OUTLAY ist es Angle!)

Hinweis:

Wird beim String ein ''|'' eingegeben, so gilt dieses als Trennzeichen. Bei einem BUTTON wird dieser zu einem Schaltwechsler (zum Beispiel ''On|Off'' macht, dass nach jedem klick, sich der Inhalt abwechselt). Bei einem ROSTER sind es dann die einzelne Zeilen. Es können in beiden Fällen beliebig viele Einträge gesetzt werden.

AppendSGG (Handle, String$)

Mit diesem Befehl kann man nachträglich neue Einträge hinzufügen. Auch hier kann man gleich mehrere Einträge durch ein ''|''-Trennzeichen tätigen. Dieser Befehl wird benötigt, wenn man zum Beispiel Dateinamen einliest und nacheinander eintragen lässen. Oder wenn der User neue Einträge per Laufzeit einsetzt.

DrawSGG (Handle, Xpos, Ypos)

Zeichnet ein zuvor erstelltes Hauptelement mit all den Nebenelementen. Befindet sich die Maus über ein Hauptelement, so gibt diese Funktion eine ''1'' zurück. Wurden Änderungen gemacht, so wird eine ''2'' zurück gegeben.

Mit den Positionsangaben kann man zusätzlich noch ein SGG-Hauptelement umpositionieren.

SortSGG (Handle)

Sortiert alle Einträge Alphabetisch aufsteigend eines Nebenelementes.

SGG-Befehlsreferenz

SetSelect (Handle, Arts, Wert)

Mit diesem Befehl kann man vorher festgelegte Eigenschaften eines Nebenelementes ändern:

- Wert = 1 = SGGXPOS = Die X·Position des Nebenelementes
- Wert = 2 = SGGYPOS = Die Y·Position des Nebenelementes
- Wert = 3 = SGGALGN = Die Textausrichtung vom String
- Wert = 4 = SGGXRAN = Die Breite des Nebenelementes bzw. Angle beim OUTLAY
- Wert = 5 = SGGSLCT = Die Auswahl vom BUTTON oder ROSTER bei mehreren Einträgen
- Wert = 6 = SGGVIEW = Die Sichtbarkeitsgröße (Anzahl Zeilen) beim ROSTER
- Wert = 7 = SGGLPOS = Die Listen-Scroll-Position beim ROSTER

AddSelect (Handle, Arts, Wert)

Dieser Befehl ist SetSelect gleich, nur das hier relativeingaben gemacht werden. Das ist praktisch, wenn man zum Beispiel mit dem Mausrad innerhalb eines ROSTER scrollen will, gibt man eben nur die relativen Änderungen an. Ein auslaufen ausserhalb der Liste wird automatisch unterbunden.

GetSelect (Handle, Arts)

Gibt den Wert zurück, der beim erstellen eines Nebenelementes gegeben wurde, oder durch SetSelect/AddSelect geändert wurde. Außerdem gibt es hier noch ein achten Wert, der die Anzahl Einträge zurück gibt.

- Wert = 8 = SGGSUMM = Anzahl Einträge vom BUTTON oder ROSTER

GetString$ (Handle)

Dieser Befehl gibt die augenblicklich markierten Teil eines Nebenelementes dessen String zurück. Wenn also bei einem BUTTON oder ROSTER dessen dritte Zeile markiert ist, dann wird dieser String zurück gegeben. Beim OUTLAY oder INTAKE eben das, was gerade drin steht.

SetString (Handle, String$)

Mit diesem Befehl kann man die Zeile eines Nebenelementes ändern.