FLEXIBLE WERKZEUGE ZUR STEIGERUNG DER KOSTENEFFIZIENZ IN DER SIMULATION

I. Sturhan, A. Jaroš, S. Kuhlmann und G. Sachs
Lehrstuhl für Flugmechanik und Flugregelung der Technischen Universität München
Boltzmannstraße 15, 85748 Garching

1. ÜBERSICHT

Ein wichtiges Element der Flugsimulation ist die Schnittstelle zwischen dem Simulator und dem Benutzer. Für den Forschungs-Flugsimulator des Lehrstuhls für Flugmechanik und Flugregelung der Technischen Universität München wurden Werkzeuge entwickelt, mit denen sich Schnittstellen-Anwendungen schnell und flexibel realisieren lassen. Dadurch können Entwicklungs- und Modifikationszeiten am Flugsimulator verringert und die Kosteneffizienz gesteigert werden. Zu dem beschriebenen Zweck wurde die Skriptsprache SCAMSY entwickelt, die eine flexible Softwarelösung für Aufgaben der Flugsimulation darstellt.

2. EINLEITUNG

Ein Flugsimulator ist ein komplexes System mit hohen Leistungsanforderungen einschließlich einer möglichst realistischen Nachbildung der visuellen Informationen für den Piloten im Cockpit. Über eine Schnittstelle ist der Benutzer mit dem Flugsimulator verbunden. Die Schnittstelle bietet Verbindungen zu den funktionellen Elementen im Cockpit mit den Displays und Eingabeelementen, zum Sichtsystem, zum Soundsystem usw. Ein wichtiges Element für den Piloten bilden die Multifunktions-Displays mit vielfältigen Anzeigen, Informationen und Funktionen. Für den Benutzer stehen Bedienmöglichkeiten zur Steuerung des Simulators zur Verfügung. Zur Durchführung seiner Aufgaben ist eine geeignete Schnittstelle einschließlich einer realistischen Nachbildung der Instrumente erforderlich. Hierfür wurde das Simulation Control and Management System SCAMSY entwickelt.

3. FORSCHUNGS-FLUGSIMULATOR

3.1. Eigenschaften des Forschungs-Flugsimulators

Am Lehrstuhl für Flugmechanik und Flugregelung der Technischen Universität München wurde ein Forschungs-Flugsimulator aufgebaut. Die dazu notwendige Hard- und Software wurde am Lehrstuhl entwickelt. Eines der Entwicklungsziele war eine hohe Flexibilität, um für geänderte Anwendungen eine schnelle Anpassung zu ermöglichen. Dies betrifft die Cockpit-Instrumentierung und die Simulator-Bedienmöglichkeiten, die je nach Forschungsprojekt anzupassen und zu modifizieren sind oder für die neue Elemente zu implementieren sind. Für einen Forschungs-Flugsimulator ist es daher wichtig, derartige Änderungen schnell vornehmen zu können, gegebenenfalls auch während des Betriebs. Vor diesem Hintergrund wurde am Lehrstuhl für Flugmechanik und Flugregelung die Skriptsprache SCAMSY mit dem Ziel einer flexiblen Softwarelösung für Simulationsaufgaben entwickelt.

Fortschritte in der Hard- und Softwaretechnik bei gleichzeitig sinkenden Kosten machen es möglich, PC-basierte Computersysteme in der Flugsimulation einzusetzen und auf kostenaufwendigere Speziallösungen und Workstations zu verzichten. Um die sich daraus ergebenden Vorteile nutzen zu können, wurden Standard-PC und Windows XP als Betriebssystem für den Forschungs-Flugsimulator des Lehrstuhls für Flugmechanik und Flugregelung gewählt. Zur Sicherstellung der Echtzeitfähigkeit der Flugsimulation unter Windows XP wurden spezielle Softwarelösungen entwickelt. Das Simulationsmodell des Flugsimulators wurde vollständig in MATLAB/Simulink aufgebaut. Für die Simulation wird ein Simulink-Modell erstellt.

Der Forschungs-Flugsimulator besitzt ein Zwei-Mann-Cockpit, das mit vier großen TFT-Bildschirmen ausgestattet ist, die von drei Computern angesteuert werden (Bild 1). Jeder TFT-Bildschirm weist an den Seiten frei konfigurierbare Tastenrahmen auf. Das Autopiloten-Panel und der Klappenhebel sind mit kleinen seriellen Dot-Matrix-Anzeigen ausgestattet, die ebenfalls von SCAMSY angesteuert werden.

Die Nachbildung der Außensicht erfolgt mit einem 3-Kanal-Sichtsystem auf einer gekrümmten Projektionswand (150° $\times$ 45° Sichtwinkelrelationen für Breite bzw. Höhe). Die dafür verwendete Software von der Firma Aerolabs AG wurde in OpenGL programmiert. Die Überblendungen zwischen den Sichtkanälen und die Verzerrungskorrektur für die Rundung der Leinwand werden von dieser Software während der Laufzeit berechnet.

3.2. Anforderungen an das Softwaretool SCAMSY

Ein wichtiger Aspekt des Softwaretools SCAMSY bei der Entwicklung des Flugsimulators betraf Flexibilität und Modifizierbarkeit. Ein Wechsel der Cockpit-Instrumentierung beim Übergang von einem Flugzeug zu einem anderen sowie auch Änderungen in der Simulationskontrolle bei neuen Forschungsprojekten sollten möglichst einfach durchgeführt werden können und wenig Zeit erfordern.

Die schnelle Erlernbarkeit zur Nutzung der Software war ebenfalls ein Ziel bei der Entwicklung von SCAMSY. Dies wurde durch die Erstellung einer einfachen Skriptsprache mit wenigen, leicht zu erlernenden Syntaxbefehlen erreicht. Unter Verwendung von digitalen Fotos von Cockpits können schnell Instrumenten-Nachbildungen generiert werden, die realistisch aussehen und über die erforderliche Funktionsfähigkeiten verfügen. Ein integrierter Debugger unterstützt wirksam den Entwicklungsprozess. Außerdem vereinfacht ein Netzwerk-Modul die Einbindung von Variablen aus den Simulationsprozess. Für die Gra-

fikdarstellung arbeitet SCAMSY mit OpenGL [1], wodurch alle Beschleunigungsoptionen aktueller Grafikkarten, die zu OpenGL kompatibel sind, genutzt werden können.

Werden für ein SCAMSY-Projekt Funktionalitäten benötigt, die über den Standardumfang hinausgehen, können diese in einem eigenen Plugin integriert werden. Diese Plugins, vom Benutzer erstellte Dynamic Link Libraries (DLLs), sind in SCAMSY eigene Objekte, die in dem normalen Skriptcode wie jedes andere Objekt aufgerufen und verwendet werden können.

4. SIMULATION CONTROL AND MANAGEMENT SYSTEM SCAMSY

4.1. Grundkonfiguration

Das System SCAMSY besteht aus drei größeren Schnittstellen, den Konfigurationsdateien, den DLLs und der Netzwerk-Schnittstelle (Bild 2). Die Konfigurationsdateien betreffen Textdateien, die den Skriptcode enthalten und das Layout festlegen, sowie Bitmaps, die als Texturen für die Displaygrafik, Zeichensätze und Schaltflächen dienen. Die Konfigurationsdateien werden vom Hauptprogramm als erste geladen. Während des Ladens wird innerhalb der Benutzerschnittstelle die hierarchische Objektstruktur des SCAMSY-Projekts erzeugt, wodurch gleichzeitig die Abhängigkeiten der einzelnen Objekte voneinander festgelegt werden. Die Objekte werden aus DLLs geladen, die alle Informationen über die Funktionalität und das Aussehen des jeweiligen Objekts beinhalten. DLLs können vom Benutzer auch selbst in C++ erstellt werden. Die Standard Objects Library von SCAMSY enthält bereits eine Vielzahl vorgefertigter Objekte mit den wichtigsten Grundfunktionen.

Das Hauptprogramm nimmt die Eingaben des Benutzers hinsichtlich eines Objektes auf, z.B. mit einem Mausklick auf eines der dargestellten Elemente, und gibt die Befehle an die Funktionen des jeweiligen Objekts weiter. Der Benutzer kann hier auch eigene Funktionen bezüglich eines Objektes spezifizieren, z.B. das Verhalten eines mit der Maus ausgewählten Elementes bei Betätigen einer Taste. Das Hauptprogramm ist auch für das Fenstermanagement und die Prozessprioritäten unter Windows verantwortlich, wenn mehrere Programme gleichzeitig auf demselben Computer laufen.

Für den Nutzer von SCAMSY bilden die Konfigurationsdateien den Kernbereich. Da Programme bei größeren Projekten unübersichtlich werden können, wurde in SCAMSY die Struktur möglichst einfach gehalten. Es gibt in SCAMSY eine Konfigurationsdatei, die während des Programmstarts zur Hauptapplikation geladen wird. In dieser Datei sind die grundlegenden Fenstereinstellungen (Position und Größe des Fensters, Vollbild- oder Fenstermodus), zusätzliche Pfade, in denen die Projektdateien liegen, sowie der Name der Haupt-Layout-Datei und der Netzwerkskonfigurationsdatei festgelegt.

4.2. Layout-Dateien (Bild 3)

Die Haupt-Layoutdatei enthält das Layout des gesamten Projekts. Üblicherweise stehen dort außer den Ladebefehlen für die DLLs nur include-Befehle zu anderen Konfigurationsdateien.

Der include-Befehl lagert dabei lediglich Teile der gesamten Layoutdatei in einzelne Layoutdateien aus. Dies ist gerade bei größeren Projekten hilfreich, da die Projekte so strukturierter und übersichtlicher gegliedert werden können. Außerdem können sinnvoll zusammenhängende Komponenten des Projekts (z.B. die Anzeigen Attitude Indicator, Altitude Indicator usw.) in eigene Konfigurationsdateien geschrieben werden, die in anderen Projekten in einfacher Weise wieder verwendet werden können.

Innerhalb der Konfigurationsdateien werden den Texturen die Bitmaps zugeordnet. Mittels eines Namens ist eine Nutzung für ein Objekt möglich. Texturen können auch mehrfach innerhalb eines Projekts verwendet werden. Zusätzlich können Texturen auch als Font genutzt werden. Um einzelne Buchstaben aus dem Bitmapfont darzustellen, werden die Bitmaps vom Programm über eine Matrix in kleine Zellen aufgeteilt. Die Größe der einzelnen Zelle, d.h. des einzelnen Buchstabens, wird dabei während des Ladens der Textur über eigene Befehle festgelegt.

Der Hauptzweck der Layoutdateien ist die Spezifizierung der Objekte des Projekts. Alle Objekte in einem SCAMSY-Projekt werden in einem hierarchischen Objektbaum gespeichert. Jedes Element des Objektbaums ist ein einzelnes Objekt. Die meisten werden in sichtbarer Weise dargestellt. Jedoch gibt es auch spezielle Elemente, die nicht sichtbar sind, sondern andere Eigenschaften besitzen. Viele Objekte können zusätzlich als sichtbare Oberfläche eine Textur nutzen. In einem komplexen Instrument können diese sichtbaren Objekte entweder vor oder hinter einem anderen Objekt gezeichnet werden.

Ein wichtiger Aspekt für die Nutzung eines hierarchischen Objektbaums betrifft die funktionale Abhängigkeit von Unterobjekten. Sobald ein übergeordnetes Objekt gedreht oder bewegt wird, werden die diesem untergeordneten Objekte mitgedreht oder mitbewegt. Dies ist insbesondere dann nützlich, wenn Teile aus einem Projekt in ein anderes Projekt kopiert werden, da sich dort aufgrund der hierarchischen Struktur alle Objekte in gleicher Weise verhalten wie im vorherigen Projekt.

Eine weitere Möglichkeit in SCAMSY ist die Nutzung von relativen Größen bei Objekten. Hierarchisch im Objektbaum untergeordnete Objekte können Eigenschaften in relativen Werten von übergeordneten Objekten zugewiesen werden. Verändert sich beispielsweise die Breite des obersten Objekts, würde sich, bei relativer Abhängigkeit der Breite vom übergeordneten Objekt, die Breite aller untergeordneten Objekte mit verändern.

Die Interaktion eines SCAMSY-Projekts mit Variablen aus dem Netzwerk des Simulators erfolgt über Links. Diese Links haben die Aufgabe, Objekteigenschaften mit Variablen (z.B. aus dem Simulationsprozess) zu verknüpfen und somit Änderungen während der Laufzeit zu ermöglichen.

4.3. Netzwerkschnittstelle

Die Netzwerkschnittstelle VisIO [2] dient zum Austausch von Daten zwischen verschiedenen Programmen. Grundsätzlich wird der Datenstrom in verschiedene Kanäle unterteilt. Jeder Kanal weist eine eigene Wiederholrate und eine eigene Adresse auf. Er untergliedert sich wiederum in einzelne Einträge (z.B.AIRSPEED als Variablen-

eintrag für die Geschwindigkeit). Die Richtung des Datenstroms wird für jeden Kanal einzeln festgelegt. Die Daten können abhängig von der Konfiguration entweder zu einem bestimmten Host oder, wie es üblicherweise der Fall ist, an das gesamte Netz gesendet werden, wo sie jedem Programm zugänglich sind. Die Netzwerkschnittstelle von SCAMSY nutzt das Netzwerksprotokoll VisIO und wurde so implementiert, dass der Benutzer mit nur wenigen normierten Zeilen in einer einfachen Textdatei die gewünschte Konfiguration generieren kann.

4.4. Verknüpfungen

Jedes Objekt kann eine Vielzahl von Eigenschaften besitzen, z.B. Größe, Drehwinkel, Position auf dem Bildschirm, oder auch in komplexerer Weise mit einem Textinhalt oder Textoptionen versehen sein. Alle derartigen Eigenschaften können mit den Eingabeparametern aus dem Netzwerk verknüpft werden. Dies stellt eine Verknüpfung zwischen Netzwerksvariablen und Objekteigenschaft dar, die bei jedem Zeichenzyklus neu berechnet wird. Beispiele sind Verknüpfungen in den Farben, Positionen oder Größen sowie die Zuordnung von Drehwinkel und Variablenwert in einem Zeigerinstrument. Ein weiteres Beispiel ist in Bild 4 gezeigt.

4.5. Debugger

Ein standardmäßiges Plugin von SCAMSY ist der Debugger. Dieses Werkzeug kann alle Netzwerksvariablen und Objekteigenschaften der einzelnen Objekte visualisieren und die zugehörigen Werte anzeigen. Zusätzlich wird es dem Benutzer ermöglicht, jeden Parameter, sowohl hinsichtlich Objekteigenschaft als auch Netzwerkvariablen, einzeln zu verändern und die Auswirkungen auf das gesamte System direkt zu beobachten. Ein Beispiel für den Debugger zeigt Bild 5.

4.6. Skripte

Der Betrieb einer Anzeige kann nicht nur von einer Variablen, sondern zusätzlich auch von einfachen Operationen oder Bedingungen abhängig sein. Außerdem können weitere Zusammenhänge wie die Anbindung an eine externe Software vorliegen. In SCAMSY kann dies über eine zusätzliche Skriptdatei behandelt werden. Zeitabhängige Funktionen oder Bedingungen vom Type if/else lassen sich hier verwenden. Weiter besteht die Möglichkeit, über Skriptbefehle auf externe Hardware zuzugreifen. So ist z.B. eine Anbindung an ein serielles LCD-Display realisiert, das in einem Ultraleichtflugzeug-Simulator verwendet wurde [3], vgl. hierzu auch Bild 6.

4.7. Plugins

Die in SCAMSY enthaltene Standard Objects Library ist für den Nutzer das Kernstück bei der Arbeit mit der Software. Die Standard Objects Library enthält alle wichtigen Basisobjekte, wie Linie, Rechteck und Dreieck, sowie komplexere Elemente, wie Knopf, Text und Eingabefeld. Die Objekte aus der Standard Objects Library sind mit derselben Benutzerschnittstelle verbunden, die für die Entwicklung und Einbindung von Elementen verwendet wird, die der Benutzer generiert. Ein Beispiel hierfür ist ein Navigationsdisplay, für das Informationen aus einer Datenbank ausgelesen und dann dargestellt werden.

4.7.1. Standardobjekte

Das wichtigste Objekt ist ein als Box bezeichnetes Element (Bild 7). Eine Box in SCAMSY ist ein rechteckiges Feld, das gedreht, verschoben und in der Größe verändert werden kann. Weitere Merkmale sind Farbe und Transparenz. Besonders wichtig ist die Möglichkeit, der Box eine Textur zuweisen zu können. Hierfür wird ein Bitmap verwendet, das mit einer Bildbearbeitungssoftware erstellt werden kann. Das Bitmap wird in SCAMSY als Textur der Box zugeordnet und auf der rechteckigen Fläche abgebildet. Dies ist besonders nützlich, wenn Instrumente aus fotographischen Vorlagen realistisch nachzubilden sind.

Ein weiteres wichtiges Element ist das Label für ein Textfeld (Bild 7). Das Label zeigt entweder einen Wert an oder einen ihm zugewiesenen Textstring. Als zusätzliche Eigenschaft besitzt das Textfeld auch noch die Textausrichtung. Eine Abwandlung des Textfelds ist das Eingabefeld, das Texteingaben der Tastatur in eine Variable speichern kann.

Für komplexere Aufgaben wird häufig auch eine Maske benutzt. Sie wird verwendet, wenn Teile eines Objekts nicht sichtbar sein sollen. Bild 8 zeigt hierzu ein Beispiel, in dem Masken bei der Darstellung des künstlichen Horizonts oder der beweglichen Anzeigebänder für Geschwindigkeit und Höhe benutzt werden. Die Maske verhindert die Anzeige von Unterobjekten außerhalb der Fläche des betreffenden Objekts. Eine Abwandlung der Maske ist die Bitmapmaske, für die als Maskenfläche ein Bitmap verwendet wird. Mittels spezieller Farbcodierung lassen sich die Bereiche spezifizieren, in denen Unterobjekte dargestellt werden und in denen sie nicht sichtbar sind. Damit lassen sich auch sehr komplex geformte Abdeckflächen erzeugen.

4.7.2. Selbsterstellte Objekte

Falls die Standard Objects Library nicht für eine Aufgabe ausreicht, können vom Benutzer Objekte in SCAMSY generiert werden. Derartige Objekte lassen sich in C++ schreiben und werden als ein Plugin über die Benutzer-Schnittstelle in SCAMSY geladen. Diese wird in zwei unterschiedliche Klassen unterteilt.

Die erste Klasse wird als Object Class bezeichnet, die dieselben Callbacks nutzt wie jedes Objekt der Standard Objects Library. Derartige vom Benutzer erstellten Objekte sind in SCAMSY dann wie jedes andere Objekt verfügbar. Hierbei kann der Benutzer dem Objekt interaktiv über Maus- oder Tastatureingaben Befehle mitteilen. Im Hinblick auf ihre Darstellung auf dem Display wird festgelegt, ob dies vor oder nach der Zeichnung der ihnen übergeordneten Objekten erfolgt. Für Zeichenoperationen wird die Grafikbibliothek OpenGL benutzt.

Die zweite Klasse betrifft die Bibliothek, die im Hintergrund für Berechnungen hinsichtlich der jeweils zugeteilten Objekte dient. Hierbei können auch Aufgaben durchgeführt werden, die keine Verbindung mit SCAMSY besitzen, wie z.B. das Laden einer Datenbank während des Startens des Programms. Auf diese Klasse hat der Benutzer im Gegensatz zur Object Class keine weitere interaktive Zugriffsmöglichkeit.

4.8. Weitere Anwendungen

Für die Erstellung von SCAMSY-Projekten ist ein Windows-Editor in Entwicklung [4]. Mit diesem Editor wird es möglich, ein SCAMSY-Projekt mit Mausklicks sowie mit dem Bewegen und Positionieren von Objekten mit der Maus (drag and drop) zu erstellen. Des Weiteren ist für diesen Windows-Editor eine Objektdatenbank geplant. Die Objektdatenbank soll eine Vielzahl von Standardinstrumenten und Simulationskontroll-Objekten enthalten. Mit diesen Werkzeugen kann ein SCAMSY-Benutzer ein vollständiges Instrumentenpanel mit Mausklicks und unter Verwendung von Instrumenten aus der Datenbank erstellen.

5. ZUSAMMENFASSUNG

Das Softwaretool SCAMSY wurde für die Generierung und Ansteuerung von Simulations-Instrumentierungen entwickelt. Damit lassen sich realistische Nachbildungen von Instrumenten und komplexe Simulationskontrollen mit vergleichsweise geringem Aufwand erstellen. Mit dem Softwaretool SCAMSY ist es möglich, die visuellen Mensch-Maschine-Schnittstellen eines Flugsimulators auf kosteneffiziente Weise in kurzer Zeit zu generieren. Die einfache Struktur der Skriptsprache erlaubt eine schnelle Einarbeitung.

6. REFERENZEN

[1] M. Woo, Davis, Mary Beth Sheridan: The OpenGL Programming Guide. Addison-Wesley, Reading, MA, 1997.
[2] Heise, M.: Development and Implementation of a Communication Interface for a Research Flight Simulator. Semesterarbeit, Lehrstuhl für Flugmechanik und Flugregelung, Technische Universität München, 2004.
[3] F. Holzapfel, A. Jaros, G. Mumelter, G. Sachs, S. Müller, M. Heise, M. Betsch: Kosteneffizienter Trainingssimulator für Leicht- und Ultraleichtflugzeuge. DGLR Jahrestagung 2004.
[4] F. Schauer: Grundlagen eines Windowseditors für SCAMSY. Semesterarbeit, Lehrstuhl für Flugmechanik und Flugregelung, Technische Universität München, 2005.

Bild 1 Forschungs-Flugsimulator mit vier TFT Bildschirmen und seriellen LCD-Displays auf dem Autopilotenpanel mit SCAMSY-Ansteuerung

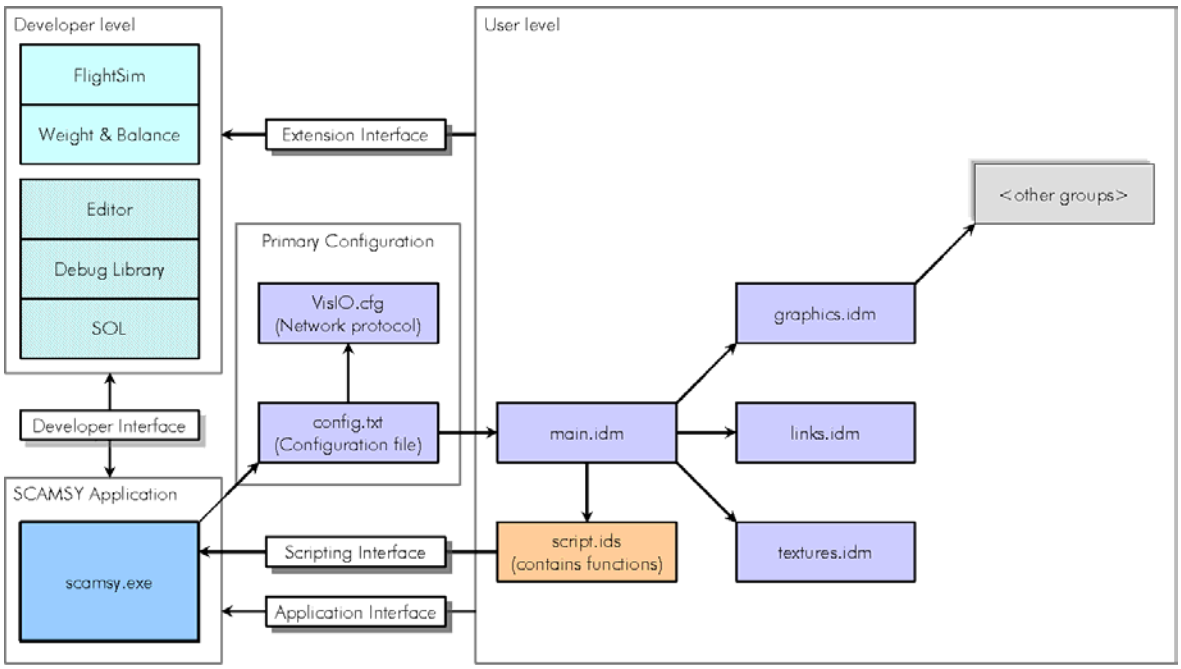

Bild 2 Struktur von SCAMSY

```
// Attitude Indicator
// CT Ultralight Aircraft Simulator
// 2004 by Andreas Jaros

// Inner Part
// Static Part
BoxMask( AI_BG_MASK, 0, 0, 100%, 100%)
{
    Box( AI_ROLL, 0, 0, 100%, 100%)
    {
        // Rolling Background
        Texture ( 0, BMP_AI_ROLL);

        // Pitching Scale
        Box( AI_PITCH, 0, 0, 100%, 100%)
        {
            Texture ( 0, BMP_AI_PITCH);
        }

        // Bank Angle Indicator
        Box( AI_RING, 0, 0, 100%, 100%)
        {
            Texture ( 0, BMP_AI_RING);
        }

    }
    // Aircraft Symbol
    Box( AI_AC, 0, 0, 100%, 100%)
    {
        Texture ( 0, BMP_AI_AC);
    }
    Box( AI_BG, 0, 0, 100%, 100%)
    {
        Texture ( 0, BMP_AI);
    }
}
```

```
Container (LEFT_TOP, -220, 150, 50%, 50%)
{
    Container (PFD, 0, 0, 400, 400)
    {
        #include "AI.idm";
    }

    Container (AT, 242, 0, 40, 400)
    {
        #include "AT.idm";
    }

    Container (ST, -242, 0, 80, 400)
    {
        #include "ST.idm";
    }
}

Container (RIGHT_TOP, 280, 150, 50%, 50%)
{
    Container (EP, 0, 0, 400, 400)
    {
        #include "EP.idm";
    }
}

Container (RIGHT_BOTTOM, 280, -150, 50%, 50%)
{
    Container (SF, 0, 0, 400, 400)
    {
        #include "SF.idm";
    }
}
```

Bild 3 Beispiele für eine Layoutdatei

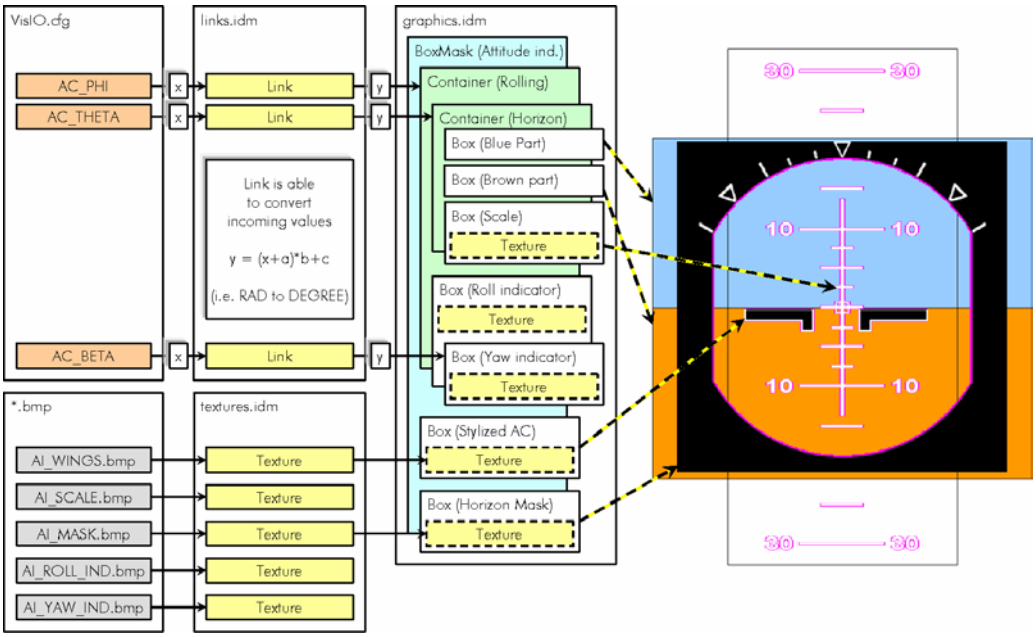

Bild 4 Beispiel eines mit Links animierten Layouts

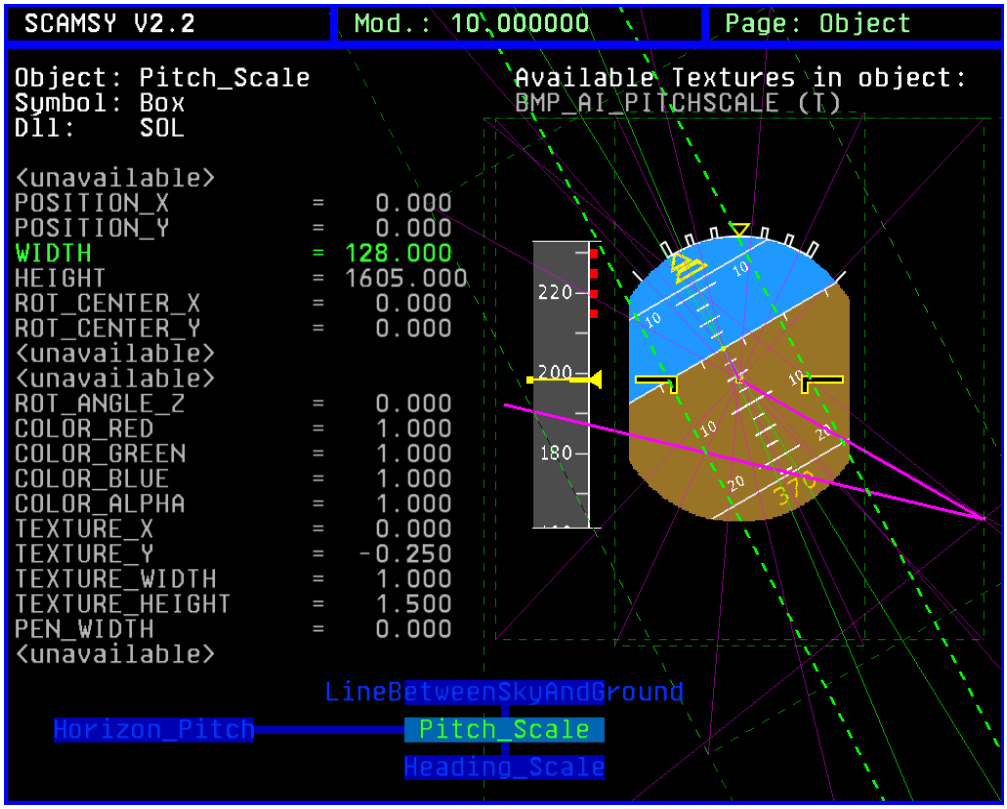

Bild 5 Teile eines künstlichen Horizonts im SCAMSY Debugger

Bild 6 Instrumtenpanel eines Ultraleichtflugzeug-Simulators mit fotorealistischen SCAMSY-Instrumenten auf einem mit einer Maske versehenen TFT-Bildschirm (links) und mit seriellem LCD-Display (rechts)

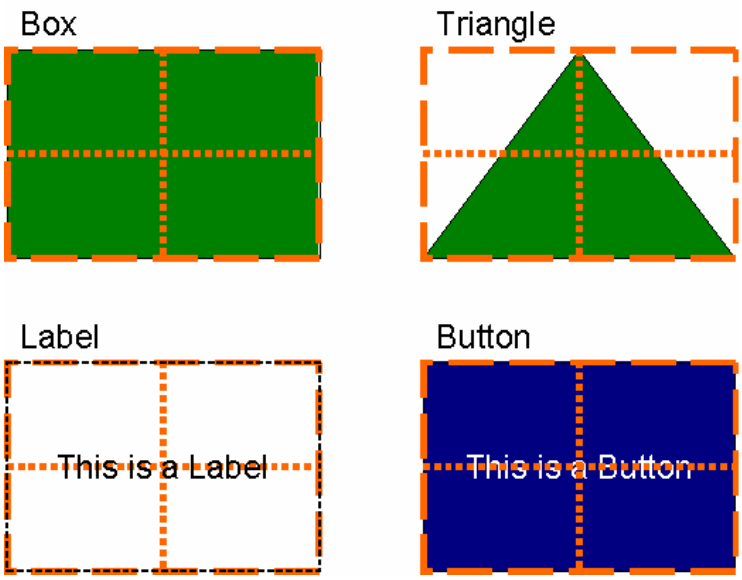

Bild 7 Objekte der Standard Object Library von SCAMSY

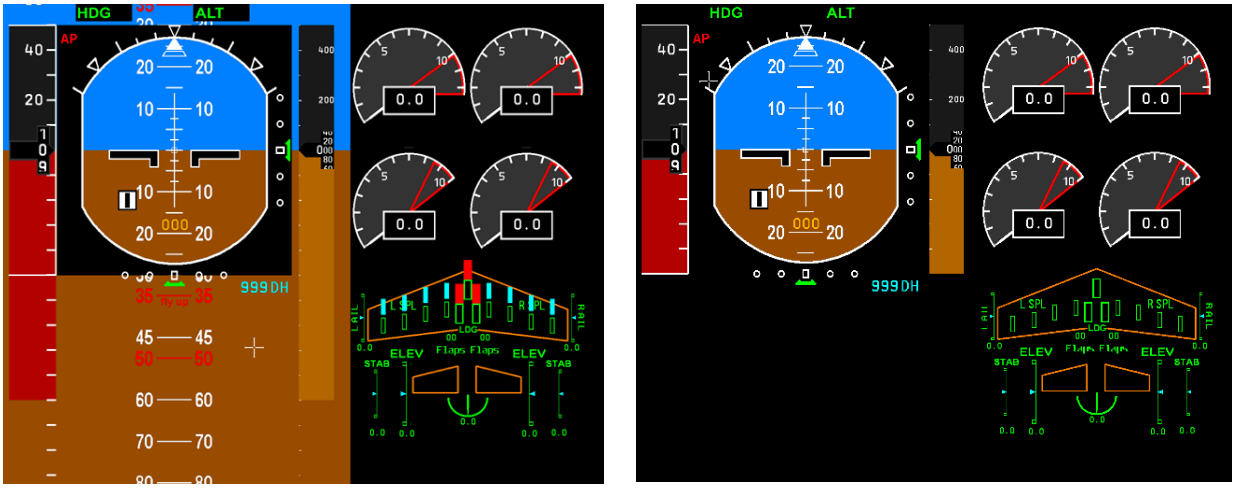

Bild 8 Effekt der Maske in SCAMSY (links: ohne Maske, rechts: mit Maske)