# Fopra: GTK für Alice

Bericht vom 05.09.02

Robert Grabowski

## Einleitung

Ziel des Fortgeschrittenenpraktikums ist die Erstellung von GTK-Bindings für die Programmiersprache Alice. Dabei soll die Bibliothek GTK+ 2.0 unter Alice möglichst genauso benutzbar sein wie unter C. [1]

GTK+ 2.0 ist keine in sich geschlossene Bibliothek, sondern baut auf den Bibliotheken GLib (C-Hilfsfunktionen zum Speichermanagement), ATK (Eingabehilfen-Toolkit), Pango (Schriftartenunterstützung) und GDK (primitive Grafikfunktionen) auf. Beim Erstellen von grafischen Benutzeroberflächen greift man aber normalerweise höchstens auf GDK zurück.

Außerdem sollen Bindings für das GtkCanvas-Widget generiert werden, welches eine Zeichenfläche zur Verfügung stellt. Alles in allem werden also Schnittstellen für GTK, GDK und GtkCanvas erzeugt. [2]

## Generierung

Die Generierung der Schnittstellen läuft in mehreren Phasen ab.

1. Es wird eine GTK-Headerdatei mit allen Funktionen, Structs, Enums und Typedefs der Bibliothek erzeugt, indem einfach eine Datei mit der Zeile `#include <gtk/gtk.h>` durch den C-Präprozessor geschickt wird.

2. Die Ausgabe wird von einem Hilfsprogramm von diversen GNU-Compiler-spezifischen Schlüsselwörtern, Leerzeilen u.ä. bereinigt.

3. Das eigentliche Programm zur Generierung der Schnittstellen benutzt zunächst das CKit[3], um die bereinigte Datei zu parsen und eine Darstellung in Form eines abstrakten Syntaxbaumes (AST) zu erstellen.

4. Diese Darstellung des CKits wird daraufhin in eine eigene Zwischendarstellung umgewandelt, die auf unnötigen Ballast verzichtet.

5. Ein Funktor erstellt die unsichere Schicht, bestehend aus einer UnsafeX.cc und einer UnsafeX.asig, wobei X für "Gtk", "Gdk" oder "GtkCanvas" steht. (Der jeweilige Namespace wird dem Funktor als Parameter übergeben.)

6. Genauso gibt es einen Funktor für die sichere Schicht, bestehend aus der X.aml und der X-sig.aml.

[1] Unter Alice gibt es jedoch eine andere Namenskonvention, so wird beispielsweise die Funktion "gtk_widget_show_all" den Namen "Gtk.widgetShowAll" haben.

[2] Leider läuft GtkCanvas noch nicht unter der GTK-Version 2.0. Die Portierung des Widgets und die Generierung der Schnittstelle verschiebe ich daher auf später, natürlich auch in der Hoffnung, dass der Autor von GtkCanvas bis dahin selbst eine aktuelle Version herausgebracht hat...

[3] Das CKit ist momentan nur für SML/NJ verfügbar. Der ganze Generator läuft daher noch nicht unter Alice, obwohl das vielleicht wünschenswert wäre.

## Unsafe-Komponenten

In der unsicheren Schicht werden im Wesentlichen die Funktionen der Bibliothek grundsätzlich verfügbar gemacht. Folgende Konvertierungen werden vorgenommen:

### Signatur

- Die C-Datentypen finden eine möglichst deckungsgleiche Alice-Entsprechung:

| C-Typ | Alice-Typ |
|---|---|
| void | unit |
| (ganzzahliger Typ) | int |
| (Fließkomma-Typ) | real |
| gboolean | bool |
| char* | string |
| (enum-Typ) | int |
| t[] | t array |
| t* | 'object (abstrakter Typ) |
| GList | 'object list |
| GSList | 'object list |

- Manche der C-Funktionen schreiben ihre Ausgabe in Variablenparamter (Zeiger). Da es so etwas in Alice nicht gibt, tauchen diese Parameter nicht im Eingabetupel, sondern im Ausgabetupel auf (welche es wiederum in C nicht gibt). Beispiel: Eine C-Funktion `float f(int* x)` bekommt im Alice-Binding den Typ `f : unit -> real * int`.

### Struktur

Die eigentliche Unsafe-Komponente ist eine C++-Datei, welche für jede Bibliotheksfunktion eine Wrapperfunktion enthält. Jede Wrapperfunktion sorgt dafür, dass

- die Eingabeparameter aus der internen Darstellung der VM (word) in die entsprechenden C-Datentypen konvertiert werden,
- die Bibliotheksfunktion aufgerufen wird,
- die Rückgabewerte wieder in words umgewandelt werden.

## Safe-Komponenten (noch nicht implementiert)

Die sichere Schicht enthält im wesentlichen Wrapperfunktionen, die die entsprechenden Funktionen aus der unsicheren Schicht aufrufen. Folgende Aspekte werden dabei abgedeckt:

### Typsicherheit

Es wird ein Datentyp `object` deklariert, der neben dem eigentlichen Pointer (`'object` aus dem Unsafe-Layer) auch Typinformationen enthält. Die Wrapperfunktionen fügen bei jedem ausgehenden `object` selbst die richtige Typinformation hinzu. Umgekehrt wird bei `objects` als Eingabeparameter überprüft, ob die Klasse des Objekts der von der Funktion verlangten Klasse entspricht bzw. eine Unterklasse ist.

### Enum-Konstanten

In dieser Schicht werden die Konstanten für die Enum-Datentypen zur allgemeinen Verwendung deklariert.

### Main loop

Die Komponente startet beim Laden einen Thread, welcher die GTK-Hauptschleife (gtk_main) simuliert. gtk_main kann nicht direkt verwendet werden, da diese Schleife alle Alice-Threads blockieren würde.

Der Generator kann auf einfache Weise dazu gebracht werden, für bestimmte Funktionen keine oder eine benutzerdefinierte Schnittstelle zu generieren. Dies ist nützlich gerade für die besonderen Eingriffe, die notwendig sind für das

## Event-Handling (noch nicht implementiert)

Leider können Alice-Callback-Funktionen nicht direkt bei GTK registriert werden. Beim Registrieren einer Funktion geschieht daher folgendes:

- Die sichere Schicht speichert die Funktion zusammen mit einer ID in einer Tabelle ab und ruft eine Registrierfunktion aus der unsicheren Schicht auf.
- Diese Registrierfunktion registriert einen eigenen, in C geschriebenen Event-Handler bei GTK. Dieser Eventhandler bekommt beim Aufruf die ID der Funktion.

Tritt das entsprechende Event auf, passiert folgendes:

- Der Event-Handler in der unsicheren Schicht schreibt die ID, die er bekommen hat, in einen Strom, und gibt die Meldung "Signal verarbeitet" an GTK zurück.
- In der sicheren Schicht läuft ein Thread, der auf Eingaben aus dem Strom wartet. Empfängt er eine ID auf dem Strom, schlägt er in der Tabelle die zugehörige Callback-Funktion nach und ruft diese auf.

Das Verfahren birgt Risiken. Gibt es beispielsweise ein "delete"-Event für ein Fenster, so wird unter Umständen Event-Handler aufgerufen, welcher sofort ein "Signal verarbeitet" meldet. GTK entfernt daraufhin das Fenster aus dem Speicher, obwohl die eigentliche Callback-Funktion für das Event noch gar nicht aufgerufen wurde. Deshalb muss das Fenster-Objekt vom Event-Handler referenziert werden, und nach dem Abarbeiten der Callback-Funktion dereferenziert werden. Dadurch wird das Löschen des Objekts verzögert.

## Weitere Aufgaben

- Momentan wird die Schnittstelle nur für GTK für Unix (Linux) generiert. Natürlich sollte die Schnittstelle auch die Windows-Version von GTK benutzen können.
- Schnittstellen für das GtkCanvas.
- Einigen Gtk-Funktionen muss man ggf. einen NULL-Pointer übergeben. Deshalb muss in die Schnittstelle eine zusätzliche Funktion eingefügt werden, die einen NULL-Pointer zurückliefert.
- Zum Lesen und Schreiben der Attribute von GTK-Objekten gibt es in der GTK-Bibliothek entsprechende Get-/Set-Funktionen, die dann natürlich auch in der Schnittstelle verfügbar sind. In der GDK-Bibliothek gibt es aber einige structs (GdkColor, GdkPoint, ...), deren Felder unter C direkt gelesen und geschrieben werden müssen. Da die structs unter Alice für den Benutzer nicht direkt verfügbar sind, müssen hier zusätzliche Funktionen erzeugt werden.
- In GTK+ 2.0 wurden viele Datentypen-Verwaltungsaufgaben in die GLib ausgelagert. Möglicherweise benötigt man für die GUI-Erstellung doch Dinge aus der GLib.
- Statt Konstanten zu deklarieren, könnten Enums in Alice auch als datatypes mit nullstelligen Konstruktoren dargestellt werden. Dadurch wird in dieser Hinsicht absolute Typsicherheit hergestellt, allerdings müssten die Wrapperfunktionen dann den jeweiligen Konstruktor immer in den entsprechenden int-Wert umwandeln und umgekehrt.