# Grafische Bedienoberfläche

**4. Praktikumstermin, Y-Gruppe**

*Praktika sind Prüfungsvorleistungen, die zum genannten Termin erbracht werden müssen. Das Programm (C++-Projekt) wird während des Praktikums zum angegebenen Zeitpunkt im Informatik-Labor abgenommen.*

*Zu Beginn dieses Praktikumtermins müssen Sie Basiskenntnisse zur Thematik Klassen, Objekte, Vererbung, grafische Oberflächen und Ereignisbehandlung nachweisen und den prinzipiellen Lösungsweg erläutern können.*

## Ziel

Die Klassen des dritten Praktikums sollen in einem Programm mit grafischer Oberfläche verwendet werden, siehe Abbildung 1: In einem Formular können Daten eingegeben werden, um eine Photovoltaikanalage anzulegen. Danach kann man in weiteren Eingabefeldern eingeben, über welche Zeit welche elektrische Leistung bereitgestellt worden ist. Dies kann mehrfach geschehen; die Energiemengen werden aufaddiert. Das Programm aktualisiert nach jeder solcher Eingabe die Ausgabe der gesamten Energiemenge und die Ausgabe der $CO_2$-Bilanz.

## Aufbau des Formulars

Erzeugen Sie mit Turbo C++ eine VCL-Formularanwendung, keine Konsolenanwendung. Aus der Tool-Palette ziehen Sie die benötigten grafischen Elemente auf das zunächst leere Formular:

- `TEdit` für Eingabefelder
- `TButton` für Knöpfe
- `TLabel` zur Beschriftung und für Ausgabefelder

Im Objektinspektor vergeben Sie unter der Eigenschaft `Name` sinnvolle Namen wie `BtnProduziere` für jedes Element, auf das Sie später per Programm zugreifen wollen. Den Titel des Formulars und den Text der Buttons und Labels stellen Sie mit der jeweiligen Eigenschaft `Caption` ein. Die Eingabefelder leeren Sie, indem Sie die voreingestellten Zeichenketten in der Eigenschaft `Text` entfernen. So lange noch keine Photovoltaikanalage angelegt ist, sollen die Ausgabefelder und der untere Teil (Stromproduktion) noch ausgegraut sein (Eigenschaft `Enabled` auf `false`). Nicht verwirren lassen: Im Design-Modus sehen ausgegraute Knöpfe normal aus.

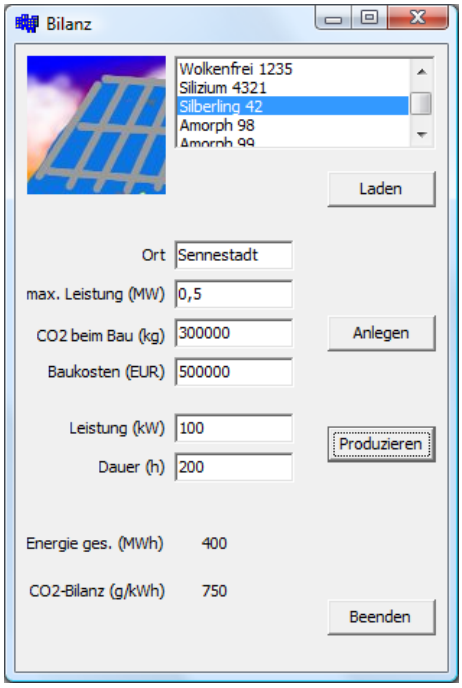


Abbildung 1: Die grafische Oberfläche des zu entwickelnden Programms. Wenn das Grundprogramm läuft, können Sie als freiwillige Erweiterung vorsehen, Voreinstellungen zu laden und dazu Bilder anzuzeigen (oberes Drittel).

Kopieren Sie die .h- und .cpp-Dateien der Klassen von Aufgabe 3 in das Projektverzeichnis. Laden Sie diese .cpp Dateien in der Projektverwaltung zum aktuellen Projekt hinzu. Ergänzen Sie in der .h-Datei Ihrer Formular-Klasse ein privates Datenelement, in dem die aktuelle Photovoltaikanalage gespeichert wird (Header der Photovoltaikanalage-Klasse inkludieren!). Füllen Sie dieses Datenelement im Constructor Ihrer Formular-Klasse mit einem Dummy. (Sauberer, aber komplizierter wäre, einen Zeiger zu verwenden und den auf `NULL` zu lassen, bis der Benutzer eine Photovoltaikanalage angelegt hat.)

## Ereignisbehandlungsroutinen

Die Knöpfe `Anlegen`, `Produzieren` und `Beenden` benötigen Ereignisbehandlungsroutinen. Durch einen Doppelklick auf eine Komponente im Formular (Ansicht: Design) wird automatisch der Rumpf der dafür üblichen Ereignisbehandlungsroutine erzeugt. Um mehr Kontrolle darüber zu haben, können Sie auch für die aktuell ausgewählte Komponente im Objektinspektor unter dem Reiter „Ereignisse“ in der rechten Spalte doppelklicken oder einen sinnvollen Namen für die Routine eingeben und die Return-Taste drücken.

Mit der Eigenschaft `Text` von Eingabefeldern lesen Sie deren Inhalt als Zeichenkette in Borlands eigener Klasse `AnsiString` aus. Um diese an einen `std::string` zu übergeben, rufen Sie mit einer Operation wie `MeinEingabefeld->Text.c_str()` den zugrundeliegenden C-String ab. Um Gleitkommazahlen aus Eingabefeldern einzulesen, rufen Sie die Elementfunktion `ToDouble` der `AnsiString`-Zeichenkette auf; Ganzzahlen liefert die Elementfunktion `ToInt`. Um Gleitkommazahlen in Oberflächenelemente auszugeben, können Sie sie mit `FloatToStr()` in eine `AnsiString`-Zeichenkette verwandeln.

Die `OnClick`-Ereignisbehandlungsroutine des Knopfs `Anlegen` liest die Werte aus den mittleren Eingabefeldern, ruft den Konstruktor von `cPhotovoltaikanlage` damit auf, schaltet den unteren Teil des Dialogs aktiv und leert die Ausgabefelder für Energie und $CO_2$-Bilanz, denn die könnten sich noch auf eine andere, vorher betrachtete Photovoltaikanalagen beziehen.

Die `OnClick`-Ereignisbehandlungsroutine des Knopfs `Produzieren` liest die Dauer und die Leistung ein, ruft die entsprechende Elementfunktion `speiseEin` von `cPhotovoltaikanalage` auf und schreibt die Gesamtenergie und die $CO_2$-Bilanz zur Ausgabe in Labels. Erweitern Sie dazu `cPhotovoltaikanlage` um eine Elementfunktion zur Abfrage der Energie.

Die `OnClick`-Ereignisbehandlungsroutine des Knopfs `Beenden` ruft die Elementfunktion `Close` des Formulars auf.

## Optionale Erweiterungen

Aus einer Liste (`TListBox`) soll man durch Klick auf den Button `Laden` einen vorgefertigten Anlagentyp wählen können. Um beim Entwurf Einträge in die Liste zu setzen, öffnen Sie mit dem Objektinspektor die Eigenschaft `Items` und fügen Zeile für Zeile (Phantasie-)Namen von Anlagen ein.

Außerdem soll zu dem aktuell in der Liste gewählten Typ ein Bild angezeigt werden. Das Bild kann einer Komponente vom Typ `TImage` erscheinen. Bereiten Sie dazu auf der Festplatte geeignete Bilddateien vor, zum Beispiel im JPEG-Format `*.jpg`.

Die `OnClick`-Ereignisbehandlungsroutine des Knopfs `Laden` wertet die Eigenschaft `ItemIndex` der Liste in einer `switch`-Anweisung aus und füllt darin die mittleren Eingabefelder mit geeigneten Werten.

Die `OnClick`-Ereignisbehandlungsroutine der Liste wertet ebenfalls die Eigenschaft `ItemIndex` der Liste in einer `switch`-Anweisung aus. Sie lädt darin mit einer Anweisung wie `MeinImage->Picture->LoadFromFile("xyz.jpg")` das passende Bild. Damit das JPEG-Format verarbeitet wird, ist `<jpeg.hpp>` zu inkludieren.