Programmieren 1 Übung: GUI-Programmierung

Klaus Kusche

15-Puzzle

Wir wollen ein Programm schreiben, das das 15-Puzzle-Spiel implementiert:
Auf einem Spielfeld mit 4 * 4 Feldern gibt es ein freies Feld.
Dieses ist so lange zu *verschieben*, bis die restlichen Felder in richtiger Reihenfolge sind.
Wir nummerieren die Felder aber nicht durch, sondern teilen ein *Bild* in 4 * 4 Teile.

Grundstruktur:

- Wir brauchen eine *globale* **`wxBitmap`**-*Variable*, die unser *Originalbild* enthält.

  Definiere weiters *Konstanten* für die *Anzahl der Zeilen und Spalten* (4), für die *Größe des Gesamtbildes in Pixeln* (ich setze eine *fixe Größe* von 640 * 640 voraus), *für die Größe eines Teilbildes* in Pixeln, und für den *Filenamen* der Bild-Datei.

- Unsere *Applikationsklasse* enthält nur die übliche *Initialisierungs-Methode*.

- Als nächstes brauchen wir eine *Klasse* für die *Darstellung unseres Bildes* im Fenster.
  Sie ist von **`wxWindow`** abgeleitet und enthält:
  - Einen *Konstruktor* (mit einem Pointer auf das Vater-Fenster als Parameter).
  - Einen *Event-Handler* (mit einem **`wxMouseEvent&`**-Parameter),
    der die Bildteile entsprechend verschiebt, wenn man mit der *Maus* ins Bild *klickt*.
  - Einen *zweiten Event-Handler* (mit einem **`wxPaintEvent&`**-Parameter),
    der sich um die *Repaint-Events* kümmert.
  - Eine *öffentliche Methode* ohne Parameter und Returnwert zum *Mischen* des Bildes.

  Weiters enthält die Klasse ein *Array als Member* (privat), das speichert, *welcher Teil* des Originalbildes gerade in welchem Teilbild des Fensters angezeigt wird.
  Ich habe ein *zweidimensionales* **`int`**-Array (4 * 4) verwendet:
  `0` ist das *leere* Feld, und *positive Zahlen* sind die *Nummer des Original-Teilbildes*, das an dieser Stelle angezeigt werden soll, wobei die Teile des Originalbildes *zeilenweise ab* **`0`** durchnummeriert sind (demgemäß wird das Teilbild in der *linken oberen Ecke* des Originalbildes durch das *leere Feld* ersetzt).

- Unsere *Hauptframe-Klasse* soll unter dem Bild *zwei Buttons* anzeigen:
  Einen zum *Beenden* des Programms und einen zum *zufälligen Mischen* des Bildes.
  Sie hat daher neben dem Konstruktor (ohne Parameter) *zwei Event-Handler* (beide mit einem **`wxCommandEvent&`**-Parameter),
  und wir brauchen *zwei Id-Nummern* für die Buttons.

  Weiters hat unsere Hauptframe-Klasse *ein Member*:
  Einen *Pointer auf das Grafik-Teilfenster* (unsere oben beschriebene Klasse),
  das das eigentliche Bild anzeigt.

Zu den einzelnen Methoden im Detail:

- **Init-Methode der Applikationsklasse:**
  - Rufe **`srand`** auf, damit das Mischen der Teilbilder wirklich zufällig ist.
  - Rufe **`wxImage::AddHandler`** mit dem Handler für deinen Bild-Dateityp auf (jpg, png oder was immer).
  - Lade die Bilddatei in die globale Bitmap. Gib eine Fehlermeldung (auf **`cerr`**) aus und returniere **`false`**, wenn das fehlschlägt.
  - Öffne wie üblich das Hauptfenster.
- **Konstruktor des Hauptfensters:**
  - Erzeuge ein *neues Grafik-Teilfenster* und speichere den Pointer darauf im Member.
  - Erzeuge wie üblich *zwei Buttons*.
  - Erzeuge *zwei Sizer*, ordne die Buttons und das Teilfenster damit an, und kopple den äußeren Sizer mit dem Hauptfenster.
  - Verbinde die Buttons mit ihren Event-Handlern.
- **Close-Button-Event des Hauptfensters:**
  - Rufe wie üblich die **`Close`**-Methode auf.
- **Mischen-Button-Event des Hauptfensters:**
  - Rufe die *Misch-Methode des Teilfensters* auf.
- **Konstruktor des Grafik-Teilfensters:**
  - Gib als Parameter beim Aufruf des Vaterklassen-Konstruktors u.a. **`-1`** als Fenster-Id und die *Bildgröße als Fenstergröße* an (sowie **`wxBORDER_SIMPLE`**, damit das Bild einen dünnen Rand bekommt).
  - Initialisiere das Array, das die Position der Teilbilder angibt: Das erste Element links oben bekommt **`0`**, und die restlichen Elemente werden *in ursprünglicher Reihenfolge* nummeriert.
  - Verbinde die Events mit ihren Event-Handlern: Der Paint-Event heißt **`wxEVT_PAINT`** und der Mausklick-Event **`wxEVT_LEFT_UP`**, beide ohne bestimmte Id.
- **Paint-Event des Grafik-Teilfensters:**
  - Lege ein *lokales* **`wxPaintDC`**-Objekt für das eigene Objekt bzw. Fenster an.
  - Lege ein *lokales* **`wxMemoryDC`**-Objekt an (das ist eine *nicht sichtbare Zeichenfläche im Speicher*) und initialisiere es mit dem *Original-Bild* aus der globalen Bitmap (Methode **`SelectObject`**).
  - *Lösche* den Bild-Inhalt der Fenster-Zeichenfläche. Sie wird dadurch komplett mit der Hintergrund-Farbe gefüllt.
  - Geh das *Array mit den Teilbild-Nummern* durch, überspringe dabei das leere Feld. *Verschiebe* jedesmal mit der Methode **`Blit`** Pixel in der Größe eines Teilbildes von der richtigen Stelle des Originalbildes im **`wxMemoryDC`**-Objekt an die richtige Stelle in Fenster-Zeichenfläche.
  - Zeichne im Fenster weiters dünne vertikale und horizontale *Linien* zwischen den Teilbildern.
- **Mausklick-Event des Grafik-Teilfensters:**
  - Ermittle aus den Event-Daten (dem *Parameter* des Event-Handlers) die *relativen Koordinaten des Mausklicks* im Zeichenfenster und rechne sie in die *Zeile und Spalte des angeklickten Teilbildes* um.

- Das angeklickte Teilbild soll *verschoben* werden.
  Prüfe daher im Array mit den Teilbild-Nummern die *vier Nachbarn*
  des angeklickten Teilbildes (*Achtung auf den Rand!*),
  ob einer davon das *leere Feld* ist.
- Wenn du *einen leeren Nachbarn gefunden* hast:
  Trage in *diesem Nachbarn* die *eigene Teilbild-Nummer* ein
  und setze dann die *eigene Teilbild-Nummer* auf **`0`** (leer).
- Wenn *kein Nachbar leer* ist, wird der Klick *ignoriert*:
  Kehre ohne weitere Aktionen zurück.
- Verwende dann **`Refresh`**, um die neue Anordnung der Teilbilder *anzuzeigen*.

- **Misch-Methode des Bild-Teilfensters:**
  - *Suche* im Array der Teilbild-Nummern *die Position des leeren Feldes*.
  - *Wiederhole* folgende Schritte (z.B. 500 Mal) für die *jeweils aktuelle Position*:
    - Ermittle einen *zufälligen Nachbarn* dieses Feldes (*Achtung auf den Rand!*).
    - Speichere *im eigenen Feld* die *Teilbild-Nummer dieses Nachbarn*
      und *im Nachbarn* die Teilbild-Nummer **`0`** (*leer*).
    - Gehe im nächsten Schritt von der *Position dieses Nachbarn* aus.
  - Verwende **`Refresh`**, um die neue Anordnung der Teilbilder *anzuzeigen*.

  Achtung: Es ist ***nicht*** zulässig, die Array-Elemente einfach *"irgendwie"* zu mischen:
  Nicht jede Anordnung der Teilbilder kann durch Verschieben erreicht werden!