Rozdział 5

Menu, paski narzędziowe i podpowiedzi

Dobry interfejs aplikacji jest często równie ważny, jak jej wnętrze. Interfejs jest pierwszą rzeczą, którą widzą użytkownicy - jeśli nie udostępnia on szybkiego i intuicyjnego sposobu przeprowadzenia żądanych czynności, oznacza to, że programiście nie udało się napisać dobrej aplikacji. Podobnie jak w przypadku samochodów, których powodzenie jest w równej mierze kwestią marketingu, jak pracy inżynierów, interfejs aplikacji jest równie ważny, jak jej wewnętrzne algorytmy. Zwłaszcza osoby nie parające się programowaniem często sądzą aplikację po jej wyglądzie i łatwości użycia, a nie po algorytmach.

Użytkownicy przyzwyczaili się do dobrze zaprojektowanych menu i pasków narzędziowych, umożliwiających szybki dostęp do często wykorzystywanych funkcji. W rozdziale tym zaprojektujemy prosty interfejs przy pomocy funkcji GTK+. Znajdzie się w nim menu i pasek narzędziowy.

Uruchamianie aplikacji

Należy stworzyć główne okno aplikacji i ustawić jego atrybuty. W oknach najwyższego poziomu zazwyczaj ustawia się takie atrybuty jak rozmiar, tytuł i obramowanie. Istotną sprawą jest tytuł aplikacji. Funkcja gtk_window_new tworzy okno najwyższego poziomu, w którym określamy charakterystyki aplikacji.

```
okno = gtk_window_new (GTK_WINDOW_TOPLEVEL);
```

Po utworzeniu okna możemy używać uchwytu zwróconego przez gtk_window_new, aby ustawić cechy okna. Tytuł pozwala użytkownikowi odróżnić okno od innych okien na ekranie, i powinien być możliwie krótki, ale odpowiednio opisywać okno. Tytuł ustawiamy przy pomocy funkcji gtk_widget_set_title.

```
gtk_widget_set_title (GTK_WINDOW (okno), "Tytuł aplikacji");
```

Jeśli aplikacja jest rodzajem edytora, wówczas tytuł okna powinien zawierać nazwę pliku, aby przypominał użytkownikowi, co właściwie edytuje. Pamiętajmy, że równocześnie może być otwarte wiele okien, a tytuł pomaga użytkownikowi w ich identyfikacji.

Rozmiar okna wynika z rozmiaru kontrolek w nim zawartych. Czasem rozmiar określony przez kontrolki jest zbyt mały. W takim przypadku można ustalić z góry minimalny rozmiar okna, przy pomocy funkcji gtk_widget_set_usize. Jeśli utworzone okno będzie mniejsze, wówczas jego rozmiar będzie powiększony do minimalnej wielkości. Poniższa instrukcja ustawia wymiary okna na 300 x 200 pikseli:

```
gtk_widget_set_usize (okno, 300, 200);
```

Szerokość obramowania okna można ustawić przy pomocy funkcji gtk_container_border_width. Poniższy przykład ustawia szerokość obramowania na zero:

```
gtk_container_border_width (GTK_CONTAINER (okno), 0);
```

W rozdziale 3, „Tworzenie aplikacji GUI”, nauczyliśmy się uwidaczniać okna i oczekiwać na sygnał "delete_event"; należy przeprowadzić te operacje na oknie naszej aplikacji. Sygnał "delete_event" wskazuje, że okno jest zamykane.

Kiedy okno jest gotowe, możemy zacząć dodawać do niego elementy interfejsu użytkownika, aby nasza aplikacja nabrała bardziej profesjonalnego wyglądu.

Menu (trudniejsza metoda)

Menu w GTK+ są kontrolkami i zachowują się tak, jak wszystkie inne kontrolki. Menu, które zazwyczaj przychodzi nam na myśl, składa się z dwóch części: paska menu, umieszczonego w górnej części okna, oraz kilku menu, zwieszających się z tego paska. Pasek menu jest zazwyczaj najwyżej położoną kontrolką wewnątrz okna aplikacji. Układ, w którym pasek menu znajduje się na górze okna, a pod nim umieszczony jest pasek narzędziowy, sugeruje, że menu powinno być wstawione do pionowego pola pakującego. Musimy więc stworzyć pionowe pole pakujące, zanim będziemy mogli przystąpić do tworzenia menu.

```
/* --- tworzymy pionowe pole pakujące w oknie aplikacji --- */
ypole = gtk_vbox_new (FALSE, 0);

/* --- dodajemy pole pakujące do głównego okna --- */
gtk_container_add (GTK_CONTAINER (okno), ypole);
```

```
/* --- uwidaczniamy pole pakujące --- */
gtk_widget_show (ypole);
```

Menu składa się z paska menu (GtkMenuBar), który spoczywa w poprzek całej górnej części okna, oraz poszczególnych menu (GtkMenu), które mogą być opuszczane z paska menu albo innych menu. Każda pojedyncza opcja menu jest elementem GtkMenuItem, indywidualnie tworzonym i dodawanym do GtkMenu.

Pierwszym krokiem jest utworzenie GtkMenuBar i dodanie go do pionowego pola pakującego. Funkcja gtk_menu_bar_new tworzy kontrolkę GtkMenuBar. Oto kod, tworzący pasek menu i umieszczający go w pionowym polu pakującym:

```
/* --- tworzymy pasek menu dla aplikacji --- */
pasekmenu = gtk_menu_bar_new ();

/* --- umieszczamy pasek menu w polu pakującym --- */
gtk_box_pack_start (GTK_BOX (ypole), pasekmenu, FALSE, TRUE, 0);

/* --- uwidaczniamy pasek menu --- */
gtk_widget_show (pasekmenu);
```

Kod ten tworzy pusty pasek menu. Aby pasek mógł się do czegoś przydać, należy dodać do niego elementy menu.

Kiedy omawialiśmy podstawowe kontrolki, stworzyliśmy kilka procedur, które umożliwiały przeprowadzenie kilku operacji w pojedynczej funkcji. Tej samej techniki możemy użyć w przypadku menu. Najpierw należy utworzyć element menu „Plik” (GtkMenuItem) wraz z etykietą.

```
/* --- tworzymy element menu dla opcji "Plik" --- */
menuPlik = gtk_menu_item_new_with_label ("Plik");
```

Po utworzeniu GtkMenuItem musimy dodać go do utworzonego wcześniej paska GtkMenuBar, a następnie uwidocznić.

```
/* --- dodajemy element "Plik" do paska menu --- */
gtk_menu_bar_append (GTK_MENU_BAR (pasekmenu), menuPlik);

/* --- uwidaczniamy kontrolkę --- */
gtk_widget_show (menuPlik);
```

W większości aplikacji, element menu „Plik” zawiera dalsze elementy menu, widoczne tylko wtedy, kiedy menu zostanie rozwinięte. Wybranie opcji „Plik” nie prowadzi do podjęcia żadnej czynności przez aplikację, tylko otwiera podmenu, które zawiera opcje typu „Nowy”, „Otwórz” lub „Zapisz”. Ponieważ element „Plik” w istocie prowadzi do innego menu,

należy związać go z nowym menu, zawierającym inne elementy. Po stworzeniu nowego GtkMenu, które będzie zawierać opcje „Nowy", „Otwórz" i „Zapisz", możemy związać je z elementem „Plik" przy pomocy funkcji gtk_menu_item_set_submenu.

```
/* --- tworzymy nowe menu dla opcji "Nowy", "Otwórz" itp. --- */
menu = gtk_menu_new ();

/* --- umieszczamy podmenu pod elementem "Plik" --- */
gtk_menu_item_set_submenu (GTK_MENU_ITEM (menuPlik), menu);
```

Teraz możemy dodać elementy GtkMenuItem do nowego GtkMenu. Aby uprościć procedurę dodawania podmenu, należałoby przenieść cały proces tworzenia podmenu do funkcji.

```
/*
 * UtworzPodmenuPaska
 *
 * Funkcja tworzy podmenu
 *
 * menu - menu wyższego poziomu
 * szNazwa - nazwa elementu podmenu, który ma być dodany do menu
 */
GtkWidget *UtworzPodmenuPaska (GtkWidget *menu, char *szNazwa)
{
    GtkWidget *elmenu
    GtkWidget *podmenu

    /* --- tworzymy element menu --- */
    elmenu = gtk_menu_item_new_with_label (szNazwa);

    /* --- dodajemy je do paska menu --- */
    gtk_menu_bar_append (GTK_MENU_BAR (menu), elmenu);
    gtk_widget_show (elmenu);
    /* --- tworzymy menu i dołączamy je do elementu menu --- */
    podmenu = gtk_menu_new ();
    gtk_menu_item_set_submenu (GTK_MENU_ITEM (elmenu), podmenu);
    /* --- Voila! --- */
    return (podmenu);
}
```

Po dodaniu podmenu, można rozpocząć dodawanie do niego indywidualnych elementów GtkMenuItem. Zaczynamy od utworzenia GtkMenuItem.

```
elmenu = gtk_menu_item_new_with_label ("Nowy");
```

Następnie dodajemy GtkMenuItem do właśnie utworzonego podmenu i czynimy go widocznym:

```
gtk_menu_append (GTK_MENU (podmenu), elmenu);
gtk_widget_show (elmenu);
```

Oczywiście, nasze menu jeszcze nic nie robi. Najpierw musimy przechwycić zdarzenia, sygnalizujące, że użytkownik wybrał jeden z elementów menu. GtkMenuItem posiada sygnał „activate", który wskazuje, że wybrano GtkMenuItem. Ustanawiając funkcję zwrotną dla sygnału „activate" możemy sprawić, że aplikacja zareaguje na wybór opcji z menu.

Można wprowadzić separator pomiędzy dwoma elementami menu GtkMenuItem, wywołując funkcję gtk_menu_item_new (wersja bez etykiety), i dodając otrzymany GtkMenuItem do podmenu. W menu pojawi się wówczas pozioma linia, rozdzielająca grupy poleceń; separatory tego typu są przydatne zwłaszcza w długich menu. Opcja menu „Zakończ" powinna być oddzielona od innych separatorem, aby użytkownik przypadkiem na nią nie trafił, wybierając sąsiednie opcje.

Zaznaczalne elementy menu

Oprócz wykonywania poleceń, elementy menu mogą pokazywać także informacje o stanie programu. Zaznaczalny element menu (GtkCheckMenuItem) działa podobnie jak przycisk wyboru, pozwalając oglądać i ustawiać w menu Boole'owską wartość. Procedura tworzenia GtkCheckMenuItem jest podobna, jak w przypadku tworzenia zwykłego GtkMenuItem, z tym, że obecnie używamy funkcji gtk_check_menu_item_ new_with_label i przechwytujemy sygnał „toggled „. Możemy napisać funkcję, która wykona wszystkie czynności, potrzebne do utworzenia GtkCheckMenuItem:

```
/*
 * UtworzZaznaczalnyElement
 *
 * Tworzy zaznaczalny element menu
 *
 * menu    - menu macierzyste, do którego należy dodać element
 * szNazwa - nazwa przypisana do elementu menu
 * funkcja - procedura obsługi zdarzenia, wywoływana po zmianie
 *           stanu elementu menu
 * dane    - dodatkowe dane przekazywane do funkcji obsługi zdarzenia
```

```
 */
GtkWidget *UtworzZaznaczalnyElement (GtkWidget *menu,
                                     char *szNazwa,
                                     GtkSignalFunc funkcja,
                                     gpointer dane)
{
    GtkWidget *elmenu;

/* --- tworzymy element menu --- */
elmenu = gtk_check_menu_item_new_with_label (szNazwa);

/* --- dodajemy go do menu --- */
gtk_menu_append (GTK_MENU (menu), elmenu);
gtk_widget_show (elmenu);

/* --- nasłuchujemy sygnałów "toggled" --- */
gtk_signal_connect (GTK_OBJECT (elmenu), "toggled",
                    GTK_SIGNAL_FUNC (funkcja), dane);
return (elmenu);
}
```

Stan zaznaczalnego elementu menu można ustawiać przy pomocy funkcji gtk_check_menu_item_set_state.

```
/* --- ustawiamy element na niezaznaczony --- */
gtk_check_menu_item_set_state (GTK_CHECK_MENU_ITEM (elmenu), FALSE);
```

Opcjonalne elementy menu

Elementy menu mogą działać jak przyciski wyboru, ale mogą działać także jak połączone w grupę przyciski opcji (GtkRadioMenuItem). Element menu, który ma zachowywać się jak przycisk opcji, musi być utworzony przy pomocy funkcji gtk_radio_menu_item_new_with_label, przyjmującej nazwę grupy, do której dołączany jest opcjonalny element menu, oraz nazwę tego elementu. Grupa musi zostać ustawiona na NULL przed dodaniem pierwszego elementu menu, a jej nowa wartość musi zostać pobrana po dodaniu każdego elementu. Poniższy kod ilustruje cały proces:

```
GSList *grupa = NULL;

/* --- dodajemy do grupy element "Wysoki" --- */
elmenu = gtk_radio_menu_item_new_with_label (grupa, "Wysoki");
```

```
/* --- uaktualniamy grupę, do której dodaliśmy element --- */
grupa = gtk_radio_menu_item_group (GTK_RADIO_MENU_ITEM (elmenu));
/* --- dodajemy element do menu --- */
gtk_menu_append (GTK_MENU (menu), elmenu);

/* --- uwidaczniamy element menu --- */
gtk_widget_show (elmenu);

/* --- podłączamy procedurę obsługi zdarzenia --- */
gtk_signal_connect (GTK_OBJECT (elmenu), "toggled",
                    GTK_SIGNAL_FUNC (FunkcjaWysoki), NULL);
```

GtkRadioMenuItem wywodzi się od GtkCheckMenuItem, więc sygnały są identyczne, podobnie jak funkcje służące do pobierania i ustawiania stanu elementu. Stan elementu GtkRadioMenuItem można zatem ustawić przy pomocy następującej instrukcji:

```
/* --- ustawiamy element na wyłączony --- */
gtk_check_menu_item_set_state (GTK_CHECK_MENU_ITEM (elmenu),
FALSE);
```

Podpowiedzi

Kolejnym elementem interfejsu, którego możemy użyć w aplikacjach, są podpowiedzi (*tooltips*). Są to niewielkie okienka tekstowe, pojawiające się, kiedy wskaźnik myszy znajdzie się nad kontrolką i znikające, kiedy użytkownik odsunie wskaźnik znad kontrolki. Dostarczają one użytkownikom dodatkowych informacji na temat kontrolki - to znaczy wyjaśniają, do czego służy kontrolka (są przydatne zwłaszcza wtedy, kiedy rysunki na pasku narzędziowym stworzył niezbyt kompetentny grafik). Używa się ich najczęściej w połączeniu z paskami narzędziowymi, ale mogą być także używane wraz z innymi kontrolkami. Podpowiedź tworzymy przy pomocy funkcji gtk_tooltips_new.

```
/* --- tworzymy podpowiedź --- */
podpowiedz = gtk_tooltips_new ();
```

Po stworzeniu podpowiedzi można skorzystać z funkcji gtk_tooltips_ set_tip, aby dodać je do elementu menu.

```
/* --- dodajemy podpowiedź do elementu menu --- */
gtk_tooltips_set_tip (podpowiedz, elmenu,
    "Kto rano wstaje, temu Pan Bóg daje", NULL);
```

Można zmieniać kolory podpowiedzi przy pomocy funkcji gtk_tooltips_ set_colors. Należy przekazać do niej kolory (GdkColor) tła i tekstu.

```
/* --- zmieniamy kolory --- */
gtk_tooltips_set_colors (podpowiedz, kolor_tla, kolor_tekstu);
```

Można także ustawić opóźnienie w pojawianiu się podpowiedzi przy pomocy funkcji gtk_tooltips_set_delay. Podpowiedzi można wyłączyć przy pomocy funkcji gtk_tooltips_disable i włączyć przy pomocy gtk_tooltips_ enable.

Klawisze skrótu

Wielu ludzi nie lubi wybierania opcji z menu przy pomocy myszy i woli używać skrótów klawiszowych. Dobrzy programiści pamiętają o takich osobach i tworzą skróty, służące do wykonywania najczęściej używanych poleceń. Zanim będzie można używać skrótów, należy je utworzyć i związać z oknem. Funkcja gtk_accel_group_new tworzy nową tablicę GtkAccelGroup, a funkcja gtk_window_add_accelerator_table wiążę tablicę skrótów z oknem.

```
/* --- tworzymy tablicę skrótów --- */
grupa_skrotow = gtk_accel_group_new ();

/* --- dodajemy tablicę skrótów do okna aplikacji --- */
gtk_accel_group_attach (grupa_skrotow, GTK_OBJECT (glowne_okno));
```

Chociaż tablica skrótów jest już utworzona, to klawisze skrótu nie są jeszcze odwzorowane na elementy menu. Funkcja gtk_widget_install_ accelerator odwzorowuje klawisz na element menu. Funkcja ta przyjmuje GtkMenuItem, grupę skrótów, sygnał (zazwyczaj „activate") oraz klawisze, które będą generować sygnał.

Poniższy przykład odwzorowuje sekwencję klawiszy CONTROL+C (GDK_CONTROL_MASK, 'C') na GtkMenuItem i uwidacznia klawisz skrótu w menu (GTK_ACCEL_VISIBLE). Po wciśnięciu sekwencji CONTROL+C zostanie wysłany sygnał „activate".

```
/* --- menu to jest aktywowane przez klawisze Control+C --- */
gtk_widget_install_accelerator (elmenu,
                                "activate",
                                grupa_skrotow,
                                'C',
                                GDK_CONTROL_MASK,
                                GTK_ACCEL_VISIBLE);
```

Konsolidacja kodu

Można skonsolidować powyższe operacje w jednej funkcji, tworzącej GtkMenuItem wraz ze skrótami i podpowiedziami. Funkcja ta zastępuje 10 do 15 linii kodu dla każdego dodawanego elementu menu i nieco ułatwia pisanie programu.

Funkcja przyjmuje parametry dla podpowiedzi i klawisza skrótu, ale jeśli ich nie otrzyma, po prostu ich nie ustawi. Jeśli nie podamy nazwy dla etykiety menu, wówczas zostanie utworzony separator. Cała funkcja wygląda następująco:

```
/*
 * UtworzElementMenu
 *
 * Tworzy element, umieszcza go w menu i zwraca element.
 *
 * menu - menu - pojemnik
 * szNazwa - Nazwa menu - NULL dla separatora
 * szSkrot - Klawisz skrótu - "^C" dla Control-C
 * szPodp - Podpowiedzi
 * funkcja - funkcja zwrotna
 * dane - dane dla funkcji zwrotnej
 *
 * zwraca nowy element menu
 */
GtkWidget *UtworzElementMenu (GtkWidget *menu,
                              char *szNazwa,
                              char *szSkrot,
                              char *szPodp,
                              GtkSignalFunc funkcja,
                              gpointer dane)
{
    GtkWidget *elmenu;
    /* --- Jeśli określono nazwę, tworzymy element i przypisujemy
    *     mu procedurę obsługi sygnału
    */
    if (szNazwa && strlen (szNazwa)) {
        elmenu = gtk_menu_item_new_with_label (szNazwa);
        gtk_signal_connect (GTK_OBJECT (elmenu), "activate",
                GTK_SIGNAL_FUNC(funkcja), dane);
    } else {
        /* --- Tworzymy separator --- */
```

```
        elmenu = gtk_menu_item_new ();
    }

    /* --- Dodajemy element do menu i uwidaczniamy go. --- */
    gtk_menu_append (GTK_MENU (menu), elmenu);
    gtk_widget_show (elmenu);

    if (grupa_skrotow == NULL) {
        grupa_skrotow = gtk_accel_group_new ();
        gtk_accel_group_attach (grupa_skrotow,
                GTK_OBJECT (glowne_okno));
    }

    /* --- Jeśli określono skrót klawiszowy --- */
    if (szSkrot && szSkrot[0] == '^') {
        gtk_widget_add_accelerator (elmenu,
                                    "activate",
                                    grupa_skrotow,
                                    szSkrot[1],
                                    GDK_CONTROL_MASK,
                                    GTK_ACCEL_VISIBLE);
    }

    /* --- Jeśli określono podpowiedź --- */
    if (szPodp && strlen (szPodp)) {
        gtk_tooltips_set_tip (podpowiedzi, elmenu, szPodp, NULL);
    }

    return (elmenu);
}
```

Kiedy mamy już podmenu, możemy wykorzystać powyższą funkcję, aby utworzyć elementy menu. Możemy utworzyć elementy „Nowy" i „Otwórz" wraz z podpowiedziami i klawiszami skrótu w następujący sposób:

```
/* --- tworzymy "Nowy" --- */
elmenu = UtworzElementMenu (menu, "Nowy", "^N",
                            "Tworzy nowy element ",
                            GTK_SIGNAL_FUNC (FunkcjaNowy), "nowy");
elmenu = UtworzElementMenu (menu, "Otwórz", "^O",
                            "Otwiera istniejący element",
                            GTK_SIGNAL_FUNC (FunkcjaOtworz), "otworz");
```

Użycie pojedynczej funkcji ułatwia czytanie kodu. Łatwiej też jest zmienić kod w późniejszym czasie, ponieważ cała procedura tworzenia menu znajduje się w jednym miejscu. Funkcję można rozszerzyć tak, aby odczytywała wszystkie potrzebne jej informacje ze struktury, co pozostawiamy jako ćwiczenie dla Czytelników.

Fabryki elementów (łatwiejsza metoda)

Menu można tworzyć w łatwy sposób przy pomocy „fabryk" elementów (GtkItemFactoryEntry). Fabryka elementów korzysta z wstępnie zdefiniowanej struktury, która opisuje tworzone menu. Poszczególne elementy struktury składają się z pięciu części: ścieżki menu, skrótu, funkcji zwrotnej, dodatkowych parametrów dla funkcji zwrotnej oraz znaczników, określających, czym jest dany element. Istnieją następujące znaczniki:

- <Title>. Tworzy element tytułowy.
- <Item>. Tworzy zwykły element menu.
- <CheckItem>. Tworzy zaznaczalny element menu.
- <ToggleItem>. Tworzy przełączalny element menu.
- <RadioItem>. Tworzy opcjonalny element menu.
- <Separator>. Tworzy separator, rozdzielający elementy menu.
- <Branch>. Tworzy element, który będzie przechowywał podmenu.
- <LastBranch>. Tworzy dosunięty do prawej strony element, który będzie przechowywał podmenu.

Użycie NULL, 0 albo "" w miejscu znacznika spowoduje utworzenie zwykłego elementu menu.

Na przykład podmenu „Nowy" w menu „Plik" mogłoby być zdefiniowane w następujący sposób:

```
"/Plik/Nowy", "<control>N", PlikNowy, NULL, 0
```

Tekst „Plik/Nowy" jest analizowany w celu zidentyfikowania ścieżki. W tym przypadku pierwszy ukośnik wskazuje menu główne albo pasek menu. „Plik" jest elementem na pasku menu, a „Nowy" jest elementem menu „Plik". Zapis "<control>N" oznacza, że klawiszem skrótu, który wywołuje to menu jest CTRL+N. Klawisze skrótu mogą wykorzystywać sekwencję <shift> ("<shift>b"), która oznacza, że wciśnięto klawisz SHIFT, albo sekwencję <alt> ("<alt>z"), która oznacza, że wciśnięto klawisz ALT. Funkcja zwrotna jest wywoływana wtedy, kiedy użytkownik wybierze element menu przy pomocy myszy albo naciśnie klawisz skrótu. Dodat-

kowy parametr jest przekazywany do wywoływanej funkcji zwrotnej; ustawienie go na NULL spowoduje przekazanie wartości NULL. Funkcje zwrotne dla fabryki elementów różnią się nieco od standardowych, jeśli chodzi o przekazywane parametry. Otrzymują one w wywołaniu zdefiniowane w tablicy dane, sygnał funkcji zwrotnej i kontrolkę - w takiej kolejności.

Kodowanie fabryk elementów

Kiedy tablica GtkItemFactoryEntry zostanie już utworzona, kod wykorzystujący tę tablicę jest dość krótki. Pierwszym krokiem jest utworzenie GtkItemFactory przy pomocy funkcji gtk_item_factory_new. W wywołaniu funkcji podaje się typ menu (GTK_TYPE_MENU_BAR), nazwę oraz grupę skrótów klawiszowych, utworzonych przy pomocy funkcji gtk_accel_group_new. Elementy menu z tablicy GtkItemFactoryEntry dodaje się przy pomocy funkcji gtk_item_factory_create_items, przekazując jej informacje na temat tablicy. Poniższy przykład korzysta z fabryki elementów, aby stworzyć menu dla okna. Program zawiera funkcję zwrotną, która wyświetla wykonywane czynności, oraz element menu „Zamknij", który umożliwia wyjście z aplikacji.

```
/*
 * menu.c
 *
 * Menu wykorzystujące fabrykę elementów
 */

#include <gtk/gtk.h>

static void ZamknijApl (gpointer dane_funkcji,
                        guint sygnal_funkcji,
                        GtkWidget *kontrolka);
static void PokazMenu (gpointer dane_funkcji,
                        guint sygnal_funkcji,
                        GtkWidget *kontrolka);

/*
 * Struktura tworzonego menu
 */
static GtkItemFactoryEntry elem_menu[] = {
  {"/_Plik",                NULL,              0,              0,   "<Branch>"
},
  {"/Plik/tearoff1",        NULL,              PokazMenu, 0,   "<Tearoff>"
},
```

```
  {"/Plik/_Nowy",                "<control>N",       PokazMenu, 0 },
  {"/Plik/_Otwórz",              "<control>O",       PokazMenu, 0 },
  {"/Plik/_Zapisz",              "<control>S",       PokazMenu, 0 },
  {"/Plik/Zapisz _jako...",      NULL,               PokazMenu, 0 },
  {"/Plik/sep1",                 NULL,               PokazMenu, 0,
"<Separator>"},
  {"/Plik/_Zamknij",             "<control>Q",       ZamknijApl, 0 },

  {"/_Edycja",                   NULL,               0,          0,    "<Branch>"
},
  {"/_Edycja/Wytnij",            "<control>X",       0,          0, 0},
  {"/_Edycja/_Kopiuj",           "<control>C",       0,          0, 0},
  {"/_Edycja/_Wklej",            "<control>V",       0,          0, 0},
  {"/_Edycja/_Czcionka",         NULL,               0,          0,    "<Branch>"
},
  {"/_Edycja/Czcionka/Pogrubiona",                   NULL,          PokazMenu,
  0,                             "<RadioItem>" },
  {"/_Edycja/Czcionka/_Kursywa",                     NULL,          PokazMenu,
  0,                             "<RadioItem>" },
  {"/_Edycja/Czcionka/_Podkreślenie", NULL, PokazMenu,          0,
  "<RadioItem>" },
  {"/_Edycja/_Kolor",            NULL, 0,            0,          "<Branch>" },
  {"/_Edycja/Kolor/_Czerwony",                       NULL,          PokazMenu,
  0,                             "<CheckItem>" },
  {"/_Edycja/Kolor/_Niebieski",                      NULL,          PokazMenu,
  0,                             "<CheckItem>" },
  {"/_Edycja/Kolor/_Zielony",    NULL, PokazMenu,          0,
  "<CheckItem>" },

  {"/_Pomoc",                    NULL, 0,            0,
  "<LastBranch>" },
  {"/Pomoc/_O programie",        NULL, PokazMenu,          0 },
};
/*
 * Zamykamy fabrykę...
 *
 * Funkcja zamyka aplikację po wywołaniu jej z menu.
 */
static void ZamknijApl (gpointer dane_funkcji,
                        guint sygnal_funkcji,
                        GtkWidget *kontrolka)
{
    /* --- Wyświetla komunikat o wybranym elemencie menu --- */
    g_message ("Fabryka: aktywowano \"%s\"",
```

```
        gtk_item_factory_path_from_widget (kontrolka));

    /* --- Zamyka aplikację --- */
    gtk_main_quit ();
}

/*
 * PokazMenu
 *
 * Wyświetla element menu
 */
static void PokazMenu (gpointer  dane_funkcji,
                       guint     sygnal_funkcji,
                       GtkWidget *kontrolka)
{
    g_message ("Fabryka: aktywowano \"%s\", czynność %d",
        gtk_item_factory_path_from_widget (kontrolka),
        (int) sygnal_funkcji);
}

/*
 * --- Liczba elementów w menu
 */
static int l_elem_menu = sizeof (elem_menu) / sizeof (elem_menu[0]);

/*
 * KoniecApl
 *
 * Zamykamy GTK kiedy użytkownik zamyka okno aplikacji
 */
static gint KoniecApl (GtkWidget *kontrolka, gpointer dane)
{
    gtk_main_quit ();
    return (TRUE);
}

/*
 *
 */
static void UtworzFabrykeElementow ()
{
    GtkWidget *okno = NULL;
```

```
GtkWidget *pole1;
GtkWidget *pole2;
GtkWidget *separator;
GtkWidget *etykieta;
GtkWidget *przycisk;
GtkAccelGroup *grupa_skrotow;
GtkItemFactory *fabryka_elem;

/* --- tworzymy okno --- */
okno = gtk_window_new (GTK_WINDOW_TOPLEVEL);

/* --- ustawiamy minimalny rozmiar --- */
gtk_widget_set_usize (okno, 200, 200);

/* --- Podłączamy sygnały w celu usunięcia okna --- */
gtk_signal_connect (GTK_OBJECT (okno), "destroy",
                    GTK_SIGNAL_FUNC (gtk_widget_destroyed),
                    &okno);
gtk_signal_connect (GTK_OBJECT (okno), "delete_event",
                    GTK_SIGNAL_FUNC (KoniecApl),
                    NULL);

/* --- Tworzymy nową grupę skrótów --- */
grupa_skrotow = gtk_accel_group_new ();

/* --- Tworzymy nową fabrykę elementów --- */
fabryka_elem = gtk_item_factory_new (GTK_TYPE_MENU_BAR,
                                     "<bla>",
                                     grupa_skrotow);
/* --- tworzymy elementy fabryki na podstawie danych --- */
gtk_item_factory_create_items (fabryka_elem,
                               l_elem_menu,
                               elem_menu,
                               NULL);

/* --- Dołączamy grupę skrótów do okna aplikacji --- */
gtk_accel_group_attach (grupa_skrotow, GTK_OBJECT (okno));
/* --- Ustawiamy tytuł okna --- */
gtk_window_set_title (GTK_WINDOW (okno), "Fabryka elementów");
/* --- Bez obramowania --- */
gtk_container_border_width (GTK_CONTAINER (okno), 0);
/* --- Pionowe pole pakujące. --- */
pole1 = gtk_vbox_new (FALSE, 0);
```

```
    gtk_container_add (GTK_CONTAINER (okno), pole1);
    /* --- Umieszczamy menu w pionowym polu pakującym --- */
    gtk_box_pack_start (GTK_BOX (pole1),
        gtk_item_factory_get_widget (fabryka_elem, "<bla>"),
        FALSE, FALSE, 0);
    /* --- Uwidaczniamy wszystkie kontrolki --- */
    gtk_widget_show_all (okno);
}

int main (int argc, char *argv[])
{
    gtk_init (&argc, &argv);
    UtworzFabrykeElementow ();
    gtk_main ();
    return (0);
}
```

Fabryki elementów a menu kodowane ręcznie

Fabryki elementów są doskonałym sposobem na szybkie stworzenie menu, ale posiadają pewne ograniczenia, jeśli chodzi o dodawane przez nie elementy. Nie można ich użyć, jeśli chcemy dodać do menu kontrolkę, na przykład piksmapę. Jednak we większości zastosowań są optymalną metodą dodawania menu do aplikacji. Ręczne kodowanie menu jest trudniejsze, ale bardziej elastyczne.

Piksmapy

Do wyświetlenia rysunków na przyciskach i innych kontrolkach potrzebne są piksmapy. Są to kontrolki rysunków, umieszczane często jako ikony na przyciskach lub innych kontrolkach i obrazujące działanie danej kontrolki. Paski narzędziowe w aplikacjach składają się często z całego rzędu przycisków graficznych, ponieważ zajmują one mniej miejsca, niż przyciski opisane tekstem. Jednym ze sposobów utworzenia piksmapy jest użycie edytora graficznego (na przykład GIMP-a) i zapisanie rysunku w formacie xpm. Oto przykładowy plik xpm{xe "xpm, pliki"}, utworzony w GIMP-ie (komentarze dodano ręcznie):

```
static const gchar *xpm_otworz[] = {
"16 16 4 1",            /* bitmapa 16 x 16, 4 kolory, 1 znak / kolor */
"  c None",             /* kolor 1 - ' ' = kolor przezroczysty */
```

```
"B c #000000000000",   /* kolor 2 - 'B' = czarny */
"Y c #FFFFFFFF0000", /* kolor 3 - 'Y' = żółty */
"y c #999999990000",   /* kolor 4 - 'y' = ciemnożółty */
"              ",              /* tutaj zaczynają się dane rysunku */
"         BBB    ",
"  BBBBB  B    BB ",
"  BYYYB       BB ",
" BYYYYYBBBBB    ",
" BYYYYYYYYYB    ",
" BYYYYYYYYYB    ",
" BYYYYYYYYYB    ",
" BYYBBBBBBBBBB ",
" BYYByyyyyyyyB   ",
" BYByyyyyyyyB    ",
" BYByyyyyyyyB    ",
" BByyyyyyyyB     ",
" BByyyyyyyyB     ",
" BBBBBBBBBB    ",
"              ",   /* tutaj kończą się dane rysunku */
};
```

Plik xpm składa się z trzech sekcji: nagłówka, tablicy kolorów i danych bitmapy. Nagłówek jest pierwszym rzędem w tablicy. Wygląda on mniej więcej w ten sposób: "16 16 4 1", gdzie pierwsze dwie liczby to wysokość i szerokość rysunku, trzeci numer to liczba kolorów, zdefiniowanych w tablicy kolorów, a czwarty numer to liczba znaków na kolor. W naszym przykładzie "16 16 4 1" wskazuje, że rysunek ma wysokość 16 pikseli i taką samą szerokość. Trzecia liczba określa liczbę łańcuchów za nagłówkiem, które definiują używane kolory. W przykładzie znajdują się więc cztery linie z danymi kolorów:

```
"  c None",
"B c #000000000000",
"Y c #FFFFFFFF0000",
"y c #999999990000",
```

Każda linia z danymi kolorów definiuje pojedynczy kolor w rysunku. Nagłówek określa, ile znaków (najczęściej jeden) koduje pojedynczy kolor.

Każda linia z danymi kolorów zawiera definiowany znak, literę c, oraz szesnastkową wartość albo tekst None, który określa brak koloru (kolor przezroczysty). Wartość koloru określa się w formacie RGB.

W poprzednim przykładzie, spacje oznaczają brak koloru, znak B oznacza kolor czarny (#000000000000), znak Y kolor żółty (#FFFFFFFF0000), a znak y - ciemnożółty (#999999990000). Wartości szesnastkowe, które reprezentują kolory, są 8-bajtowymi liczbami. Można także użyć czterobajtowych wartości kolorów; ciemnożółty byłby wówczas reprezentowany przez #999900 zamiast #999999990000.

Dane rysunku następują bezpośrednio za danymi kolorów. Przy pomocy zdefiniowanych wcześniej kolorów można utworzyć ikonę „Otwórz", która wygląda jak częściowo otwarty, żółty folder plików. Brak koloru oznaczany jest najczęściej przez spacje, aby łatwiej można było rozpoznać zarysy rysunku w kodzie.

Zdefiniowanie rysunku to dopiero początek. Aby go wykorzystać, należy przekształcić go na kontrolkę - będziemy mogli wówczas traktować rysunek tak, jak każdą inną kontrolkę. Aby utworzyć kontrolkę piksmapy, potrzebujemy struktury GdkPixmap, którą należy przekazać do funkcji gtk_pixmap_new wraz z maską. Na szczęście możemy wywołać funkcję GDK gdk_pixmap_create_from_xpm_d, która przyjmuje strukturę danych xpm i zwraca maskę oraz strukturę GdkPixmap.

```
/*
 * UtworzKontrolkeZXpm
 *
 * Konwertuje xpm na kontrolkę piksmapy
 * Wykorzystuje globalny wskaźnik glowne_okno
 */
GtkWidget *UtworzKontrolkeZXpm (GtkWidget *okno, gchar **dane_xpm)
{
    GdkBitmap *maska;
    GdkPixmap *dane_mapy;
    GtkWidget *kontrolka_piksmapy;

    /* --- przekształcamy łańcuchy na GdkPixmap --- */
    dane_mapy = gdk_pixmap_create_from_xpm_d (
                                    glowne_okno->window,
                                    &maska,
                                    NULL,
                                    (gchar **) dane_xpm);
    /* --- przekształcamy piksmapę na kontrolkę piksmapy --- */
    kontrolka_piksmapy = gtk_pixmap_new (dane_mapy, maska);

    /* --- uwidaczniamy kontrolkę --- */
    gtk_widget_show (kontrolka_piksmapy);
```

```
    /* --- zwracamy kontrolkę --- */
    return (kontrolka_piksmapy);
}
```

Kod ten tworzy kontrolkę piksmapy, która będzie widoczna po dodaniu jej do pojemnika. Można więc stworzyć przycisk i dodać do niego piksmapę - otrzymamy wówczas przycisk z rysunkiem. Technika ta umożliwia umieszczanie ikon na przyciskach paska narzędziowego.

Paski narzędziowe

Paski narzędziowe udostępniają użytkownikom skróty do najczęściej używanych poleceń. Zawierają zazwyczaj przyciski z ikonami, które reprezentują polecenia, ale mogą też zawierać inne kontrolki. Kontrolkę paska narzędziowego tworzy się przy pomocy funkcji gtk_toolbar_new, która przyjmuje dwa parametry, określające styl paska narzędziowego. Pierwszy parametr określa, czy ikony są rozłożone poziomo (GTK_ORIENTATION_HORIZONTAL), czy pionowo (GTK_ORIENTATION_VERTICAL), natomiast drugi parametr określa, czy przyciski wyświetlają ikony (GTK_TOOLBAR_ICONS), tekst (GTK_TOOLBAR_TEXT), czy też ikony i tekst jednocześnie (GTK_TOOLBAR_BOTH). Pasek narzędziowy w poniższym przykładzie jest ułożony poziomo; ikony leżą w poprzek głównego okna. Dodamy pasek narzędziowy do pionowego pola pakującego, które utworzyliśmy wcześniej.

```
/* --- tworzymy poziomy pasek narzędziowy z ikonami --- */
pasek_narz = gtk_toolbar_new (GTK_ORIENTATION_HORIZONTAL,
                              GTK_TOOLBAR_ICONS);

/* --- dodajemy pasek narzędziowy do pionowego pola pakującego --- */
/* --- aplikacji, tuż pod paskiem menu --- */
gtk_box_pack_start (GTK_BOX (ypole), pasek_narz, FALSE, TRUE, 0);

/* --- uwidaczniamy pasek narzędziowy --- */
gtk_widget_show (pasek_narz);
```

Mamy teraz pasek narzędziowy, ale jeszcze bez przycisków. Dodawanie kontrolek do paska i jego formatowanie wymaga użycia jednej z kilku funkcji. Trzema najczęściej używanymi są gtk_toolbar_append_item, gtk_toolbar_append_element i gtk_toolbar_append_space.

Dodawanie przycisku do paska narzędziowego

Funkcja gtk_toolbar_append_item dodaje przycisk do paska narzędziowego. Przyjmuje ona parametry określające piksmapę przycisku, podpowiedź, funkcję zwrotną, i kilka innych danych. Prototyp funkcji wygląda następująco:

```
GtkWidget *gtk_toolbar_append_item (
        GtkToolbar *pasek_narz,
        const char *tekst,
const char *tekst_podpowiedzi,
        const char *prywatny_tekst_podpowiedzi,
        GtkWidget  *kontrolka,
        GtkSignalFunc *funkcja_zwrotna,
        gpointer   dane_uzytkownika);
```

pasek_narz to pasek narzędziowy, do którego dodawany jest przycisk. Parametr tekst służy do opisu przycisków tekstowych. Nasz pasek został zdefiniowany jako graficzny, więc pole to zostanie zignorowane. Parametr tekst_podpowiedzi określa podpowiedź, która pojawia się, kiedy użytkownik przesunie wskaźnik myszy ponad przycisk. Parametr prywatny_tekst_podpowiedzi możemy na razie zignorować. kontrolka jest zazwyczaj piksmapą, którą należy umieścić na przycisku. funkcja_zwrotna określa funkcję, wywoływaną po naciśnięciu przycisku, a dane_ uzytkownika to dodatkowe dane, przekazywane do funkcji zwrotnej.

Możemy skorzystać z napisanej wcześniej funkcji UtworzKontrolkeZXpm, aby utworzyć kontrolkę z danych xpm i umieścić rysunek na przycisku. Przekazanie piksmapy jako parametru funkcji spowoduje dodanie jej do przycisku.

```
/* --- tworzymy przycisk „Nowy” na pasku narzędziowym --- */
gtk_toolbar_append_item (GTK_TOOLBAR (pasek_narz),
        NULL, "Nowe okno", NULL,
        UtworzKontrolkeZXpm (ypole, (gchar **) xpm_nowy),
        (GtkSignalFunc) PrzyciskZdarzenie,
        NULL);
```

Przycisk będzie wyświetlał przekazaną piksmapę jako ikonę. Kiedy użytkownik naciśnie przycisk na pasku narzędziowym, wówczas zostanie wywołana funkcja PrzyciskZdarzenie, w której można umieścić kod odpowiedzialny za przetworzenie zdarzenia. Zazwyczaj wywołuje się tę samą funkcję, którą wywołałoby polecenie z menu („Nowe okno”). Miłą cechą funkcji gtk_toolbar_append_item jest ustawianie procedury obsługi zdarzenia i podpowiedzi, bez potrzeby wywoływania innych funkcji.

Kiedy tworzyliśmy menu, poszczególne operacje trzeba było przeprowadzić ręcznie, więc napisaliśmy odpowiedzialną za to funkcję. Tutaj wystarczy pojedyncza funkcja, co ułatwia programowanie pasków narzędziowych.

Dodawanie innych elementów do paska narzędziowego

Funkcja gtk_toolbar_append_element przypomina funkcję gtk_toolbar_ append_item, ale jest elastyczniejsza i dlatego bardziej skomplikowana. Oprócz dodawania zwykłych przycisków (GTK_TOOLBAR_ CHILD_BUTTON) umożliwia dodawanie przełączników (GTK_TOOLBAR_CHILD_TOGGLEBUTTON), przycisków opcji (GTK_TOOLBAR_CHILD_RADIOBUTTON), kontrolek (GTK_TOOLBAR_CHILD_WIDGET) oraz tworzenie przerw (GTK_TOOLBAR_CHILD_ SPACE) pomiędzy logicznymi grupami przycisków. Funkcja ta przyjmuje dwa dodatkowe parametry: typ i kontrolkę. Parametr typ określa, jaki rodzaj kontrolki jest wstawiany do paska narzędziowego. Jeśli ustawiony jest na GTK_TOOLBAR_CHILD_WIDGET, wówczas wstawiana jest kontrolka, określona przez parametr kontrolka.

```
GtkWidget *gtk_toolbar_append_element (
        GtkToolbarChildType typ,
        GtkWidget  *kontrolka,
        GtkToolbar *pasek_narz,
        const char *tekst,
        const char *tekst_podpowiedzi,
        const char *prywatny_tekst_podpowiedzi,
        GtkWidget  *ikona,
        GtkSignalFunc *funkcja_zwrotna,
        gpointer   dane_uzytkownika);
```

Przy pomocy funkcji gtk_toolbar_append_element możemy dodać na przykład przełącznik:

```
/* --- tworzymy przycisk służący do pogrubiania czcionki --- */
pasek_pogrub = gtk_toolbar_append_element (GTK_TOOLBAR (pasek_narz),
        GTK_TOOLBAR_CHILD_TOGGLEBUTTON,
        NULL,
        NULL, "Pogrubienie", NULL,
        UtworzKontrolkeZXpm (ypole, (gchar **) xpm_pogrub),
        (GtkSignalFunc) KliknietoPogrub,
        NULL);
```

Można także najpierw utworzyć GtkToggleButton, przekazać go do funkcji jako parametr kontrolka, i zmienić typ na GTK_TOOLBAR_CHILD_WIDGET. Instrukcja ta informuje GTK+ że przekazujemy własną kontrolkę, więc nie należy tworzyć nowej, jak miało to miejsce w przypadku typu GTK_ TOOLBAR_CHILD_TOGGLEBUTTON.

Krótszą wersją tej funkcji jest gtk_toolbar_append_widget. Wymaga ona wstępnego ustawienia większej ilości parametrów, ale ma też większe możliwości. Funkcja jest zdefiniowana następująco:

```
void gtk_toolbar_append_widget (GtkToolbar **pasek_narz,
                GtkWidget  *kontrolka,
                const gchar *tekst_podpowiedzi,
                const gchar *prywatny_tekst_podpowiedzi);
```

Ponieważ funkcja ta nie ustawia funkcji zwrotnych i ikon, trzeba dodać je programowo. Metoda ta zwiększa ilość kodu, ale zapewnia większą elastyczność.

Dodawanie odstępów do paska narzędziowego

Funkcja gtk_toolbar_append_space dołącza odstęp na końcu paska narzędziowego. Jedynym potrzebnym parametrem jest pasek, do którego należy dodać odstęp.

```
/* --- dodajemy odstęp pomiędzy kontrolkami --- */
gtk_toolbar_append_space (pasek_narz);
```

Funkcja gtk_toolbar_append_space wstawia odstęp za ostatnim przyciskiem. Umieszczenie odstępów w pasku narzędziowym umożliwia połączenie przycisków w logiczne grupy. Zgrupowanie przycisków (jedna grupa dla przycisków zmieniających czcionkę, inna dla operacji na plikach) ułatwia użytkownikowi wybór przycisku, wykonującego określone polecenie.

Oprócz funkcji dołączających kontrolki na końcu paska narzędziowego, istnieją też funkcje wstawiające je na początek paska (gtk_toolbar_ prepend_item, gtk_toolbar_prepend_element, gtk_toolbar_prepend_space i gtk_t oolbar_prepend_widget). Parametry dla tych funkcji są takie same, jak w przypadku wersji dołączających kontrolki.

Czasem zachodzi konieczność wstawienia kontrolki w środek grupy znajdującej się na pasku - zazwyczaj podczas modyfikowania istniejącego paska. Przyciski/kontrolki można wstawiać w dowolnym punkcie paska przy pomocy funkcji _insert_. Przypominają one poprzednie dwie grupy

funkcji, ale posiadają dodatkowy parametr, który określa punkt wstawiania przycisku/kontrolki. Jest to liczba całkowita - indeks, wskazujący pozycję na pasku narzędziowym. Ustawienie go na zero wskazuje, że kontrolkę należy dodać jako pierwszy element, a zapis pasek_narz->num_children, że jako ostatni (pasek_narz-> num_children jest aktualną liczbą elementów w pasku narzędziowym).

Tworzenie interfejsu użytkownika dla aplikacji

Posiadając wszystkie niezbędne informacje, możemy stworzyć prosty interfejs, który można łatwo zmodyfikować dla dowolnego rodzaju aplikacji. Tworząc aplikację spróbujmy konsolidować kod we wszystkich miejscach, gdzie ma to sens. Jeśli w celu wykonania jakiejś czynności wielokrotnie wywołujemy te same trzy czy cztery funkcje GTK+, warto jest zawrzeć je w pojedynczej funkcji. W rozdziale opisującym menu stworzyliśmy właśnie taką funkcję, w której skonsolidowaliśmy wiele czynności potrzebnych do utworzenia menu.

Jesteśmy teraz przygotowani na zbudowanie od podstaw interfejsu użytkownika; podrozdział ten pomoże nam wykonać to zadanie szybko i sprawnie. Potrzebne nam będą dwa menu. W menu „Plik" znajdą się opcje „Nowy", „Otwórz", „Zapisz", „Zapisz jako...", separator i „Zakończ". W menu „Edycja" znajdą się opcje „Wytnij", „Kopiuj", „Wklej" oraz podmenu „Czcionka", z opcjami „Pogrubienie", „Kursywa" i „Podkreślenie".

Tworzenie okna i menu aplikacji

Główny program jest niewielki. Jego jedyne interesujące aspekty to utworzenie podpowiedzi i wywołanie funkcji UtworzGlowneOkno, która wykonuje całą dalszą pracę.

```
/ *
 * main
 *
 * --- Tutaj zaczyna się program
 */
int main(int argc, char *argv[])
{
   /* --- Inicjujemy GTK --- */
   gtk_init (&argc, &argv);

   /* --- Inicjujemy podpowiedzi --- */
```

```
    podpowiedzi = gtk_tooltips_new ();

/* --- Tworzymy aplikację --- */
    UtworzGlowneOkno ();

/* --- Oddajemy sterowanie do GTK --- */
    gtk_main();

return 0;
}
```

Funkcja UtworzGlowneOkno zajmuje się utworzeniem większości kontrolek, dokonując tego w kilku krokach. Pierwszym jest utworzenie głównego okna i ustawienie jego atrybutów. Atrybuty takie jak tytuł i rozmiar ustawiane są przed wyświetleniem okna. Inicjujemy tablicę skrótów, którą następnie przypisujemy do głównego okna, aby w menu można było korzystać z klawiszy skrótu, oraz dodajemy procedurę obsługi zdarzenia, która będzie oczekiwać na sygnał "delete_event". Po utworzeniu okna dodajemy do niego pionowe pole pakujące, w którym menu będzie umieszczone na górze, a pozostałe elementy znajdą się pod menu. Po dodaniu pionowego pola pakującego uwidaczniamy główne okno.

```
/*
 * UtworzGlowneOkno
 *
 * Tworzy główne okno i związane z nim menu oraz pasek narzędziowy
 */
static void UtworzGlowneOkno ()
{
    GtkWidget *ypole;
    GtkWidget *pasekmenu;
    GtkWidget *menu;
    GtkWidget *elmenu;
    GtkWidget *menuczcionka;
    GtkWidget *pasek_narz;
    GtkWidget *przycisk;

    /* --- tworzymy główne okno i ustawiamy jego rozmiary --- */
    glowne_okno = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_widget_set_usize(glowne_okno, 360, 260);

    /* --- ustawiamy tytuł i szerokość obramowania --- */
    gtk_window_set_title (GTK_WINDOW (glowne_okno), "Test menu");
    gtk_container_border_width (GTK_CONTAINER (glowne_okno), 0);
```

```
/* --- tworzymy tablicę skrótów klawiszowych --- */
tablica_skrotow = gtk_accelerator_table_new();
gtk_window_add_accelerator_table(GTK_WINDOW(glowne_okno),
                                        tablica_skrotow);

/* --- Główne okno musi oczekiwać na sygnał delete_event --- */
gtk_signal_connect (GTK_OBJECT (glowne_okno), "delete_event",
                        GTK_SIGNAL_FUNC (EndProgram), NULL);

/* --- pole pakujące dla menu i paska narzędziowego --- */
ypole = gtk_vbox_new (FALSE, 0);

/* --- umieszczamy pole w oknie --- */
gtk_container_add (GTK_CONTAINER (glowne_okno), ypole);
gtk_widget_show (ypole);
gtk_widget_show (glowne_okno);

 /* --- Pasek menu --- */
pasekmenu = gtk_menu_bar_new ();
gtk_box_pack_start (GTK_BOX (ypole), pasekmenu, FALSE, TRUE, 0);
gtk_widget_show (pasekmenu);
```

Do paska menu dodajemy podmenu „Plik", a następnie elementy, które będą znajdować się w menu „Plik". Są to (w kolejności) „Nowy", „Otwórz", „Zapisz", „Zapisz jako...", separator i „Zakończ". Do każdego elementu przypisujemy funkcję obsługi zdarzenia, podpowiedź, a we większości przypadków także klawisz skrótu. Dla większości elementów funkcją zwrotną jest FunkcjaWypisz, która wyświetla na konsoli komunikat, stwierdzający, który element menu został wybrany. Gdyby nasz przykład był rzeczywistą aplikacją, wówczas funkcja zwrotna wykonywałaby jakąś czynność. Ostatnim parametrem funkcji UtworzElementMenu są dane, które przekazywane są do funkcji zwrotnej, dzięki czemu większość elementów menu może używać tej samej funkcji obsługi zdarzenia.

```
/* -----------------
 * --- Menu Plik ---
 * ----------------- */
menu = UtworzPodmenuPaska (pasekmenu, "Plik");

/* --- Element menu _Nowy_ --- */
elmenu = UtworzElementMenu (menu, "Nowy", "^N",
            "Tworzy nowy element",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "nowy");
```

```
/* --- Element menu _Otwórz_ --- */
elmenu = UtworzElementMenu (menu, "Otwórz", "^O",
            "Otwiera istniejący element",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "otwórz");
/* --- Element menu _Zapisz_ --- */
elmenu = UtworzElementMenu (menu, "Zapisz", "^Z",
            "Zapisuje bieżący element",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "zapisz");

/* --- Element menu _Zapisz jako_ --- */
elmenu = UtworzElementMenu (menu, "Zapisz jako...", "",
            "Zapisuje bieżący element pod nową nazwą",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "zapisz jako");

/* --- separator --- */
elmenu = UtworzElementMenu (menu, NULL, NULL,
            NULL, NULL, NULL);
/* --- _Zakończ_ --- */
elmenu = UtworzElementMenu (menu, "Zakończ", "",
            "Czy może istnieć bardziej wymowna opcja?",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "zakończ");
```

Menu „Edycja" jest mniejsze i posiada następujące elementy: „Wytnij", „Kopiuj", „Wklej" i „Czcionka". Element „Czcionka" to w istocie podmenu, więc w celu jego stworzenia używamy innej funkcji, niż w przypadku pozostałych elementów.

```
/* ------------------
 * --- Menu Edycja---
 * ------------------ */
menu = UtworzPodmenuPaska (pasekmenu, "Edycja");

/* --- Element menu _Wytnij_ --- */
elmenu = UtworzElementMenu (menu, "Wytnij", "^X",
            "Usuwa element i umieszcza go w schowku",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "wytnij");

/* --- Element menu _Kopiuj_ --- */
elmenu = UtworzElementMenu (menu, "Kopiuj", "^C",
            "Umieszcza kopię elementu w schowku",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "kopiuj");

/* --- Element menu _Wklej_ --- */
```

```
elmenu = UtworzElementMenu (menu, "Wklej", "^V",
            "Wkleja element",
            GTK_SIGNAL_FUNC (FunkcjaWypisz), "wklej");
```

Wreszcie dodajemy elementy menu „Czcionka": „Pogrubienie", „Kursywa" i „Podkreślenie". Będą to zaznaczalne elementy menu, aby mogły pojawiać się w menu w postaci zaznaczonej lub nie zaznaczonej. Pasek narzędziowy tworzymy po utworzeniu wszystkich elementów menu. W poprzednim przykładzie nie zapamiętywaliśmy tworzonych na pasku przycisków, obecnie jednak musimy to uczynić. Powód wyjaśnimy niebawem.

```
/* ------------------------------
 * --- Podmenu Czcionka ---
 * ------------------------------ */
menuczcionka = UtworzPodmenu (menu, "Czcionka");

menuPogrub = UtworzMenuZazn (menuczcionka,
  "Pogrubienie", GTK_SIGNAL_FUNC (ZaznaczMenu), "pogrubienie");

menuKurs = UtworzMenuZazn (menuczcionka,
  "Kursywa", GTK_SIGNAL_FUNC (ZaznaczMenu), "kursywa");

menuPodkr = UtworzMenuZazn (menuczcionka,
  "Podkreślenie", GTK_SIGNAL_FUNC (ZaznaczMenu), "podkreślenie");

/* --- Tworzymy pasek narzędziowy --- */
UtworzPasek(glowne_ypole);
```

Zauważmy, że nie mogliśmy skorzystać w tym przykładzie z fabryki elementów, ponieważ w menu znajdują się pola wyboru. Gdyby ich nie było, prawdopodobnie łatwiej byłoby użyć fabryki elementów.

Tworzenie paska narzędziowego

Funkcja UtworzPasek tworzy pasek ze wszystkimi ikonami na przyciskach. Dodatkowo umieszcza na nim rozwijaną listę z typami czcionek, aby nadać mu bardziej profesjonalny wygląd. Najpierw trzeba utworzyć pasek i dodać go do pionowego pola pakującego.

```
/*
 * UtworzPasek
 *
 * Tworzy pasek narzędziowy
```

```
 */
void UtworzPasek (GtkWidget *ypole)
{
  GtkWidget *kontrolka;

    /* --- tworzymy pasek i dodajemy go do okna --- */
    pasek_narz = gtk_toolbar_new (GTK_ORIENTATION_HORIZONTAL,
    ➥GTK_TOOLBAR_ICONS);
    gtk_box_pack_start (GTK_BOX (ypole), pasek_narz, FALSE, TRUE, 0);
    gtk_widget_show (pasek_narz);

    /* --- Tworzymy przycisk "nowy" --- */
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek_narz),
                             NULL, "Nowe okno", NULL,
                              UtworzKontrolkeZXpm (ypole, (gchar **)
xpm_nowy),
                            (GtkSignalFunc) KliknietoPrzycisk,
                              NULL);

    /* --- Tworzymy przycisk "otwórz" --- */
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek_narz),
                              "Okno dialogowe Otwórz", "Okno dialogowe
Otwórz", "",
                              UtworzKontrolkeZXpm (ypole, (gchar **)
xpm_otworz),
                             (GtkSignalFunc) KliknietoPrzycisk,
                             NULL);

    /* --- Mały odstęp --- */
    gtk_toolbar_append_space (GTK_TOOLBAR (pasek_narz));

    /* --- Przycisk _Wytnij_ --- */
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek_narz),
                             "Wytnij", "Wytnij", "",
                               UtworzKontrolkeZXpm (ypole, (gchar **)
xpm_wytnij),
                             (GtkSignalFunc) KliknietoPrzycisk,
                               NULL);
    gtk_toolbar_append_item (GTK_TOOLBAR (pasek_narz),
                             "Wklej", "Wklej", "",
                               UtworzKontrolkeZXpm (ypole, (gchar **)
xpm_kopiuj),
                             (GtkSignalFunc) KliknietoPrzycisk,
```

```
                                NULL);

    /* --- Dodajemy odstęp --- */
    gtk_toolbar_append_space (GTK_TOOLBAR (pasek_narz));

    /* --- Tworzymy pole kombinowane. Funkcja UtworzPoleCombo
     *    tworzy pole kombinowane i przypisuje do niego funkcję
     *    obsługi zdarzenia. */
    kontrolka = UtworzPoleCombo ();

    /* --- Dodajemy pole kombinowane do paska narzędziowego --- */
    gtk_toolbar_append_widget (GTK_TOOLBAR (pasek_narz),
                               kontrolka,
                               "Czcionka", "Wybierz czcionkę");

    /* --- mały odstęp --- */
    gtk_toolbar_append_space (GTK_TOOLBAR (pasek_narz));
    /*
     * --- Tworzymy przełącznik dla opcji Pogrubienie
     */
  pasek_pogrub      =      gtk_toolbar_append_element      (GTK_TOOLBAR
(pasek_narz),
                          GTK_TOOLBAR_CHILD_TOGGLEBUTTON,
                          NULL,
                          NULL, "Pogrubienie", NULL,
                          UtworzKontrolkeZXpm (ypole, (gchar **) xpm_pogrub),
                         (GtkSignalFunc) KliknietoPrzycisk,
                          "pogrubienie");

    /*
     * --- Tworzymy przełącznik dla opcji Kursywa
     */
  pasek_kurs            =     gtk_toolbar_append_element     (GTK_TOOLBAR
(pasek_narz),
                        GTK_TOOLBAR_CHILD_TOGGLEBUTTON,
                        NULL,
                       "Kursywa", "Kursywa", "Kursywa",
                        UtworzKontrolkeZXpm (ypole, (gchar **) xpm_kurs),
                      (GtkSignalFunc) KliknietoPrzycisk,
                       "kursywa");

    /*
     * --- Tworzymy przełącznik dla opcji Podkreślenie
```

```
    */
    pasek_podkr = gtk_toolbar_append_element (GTK_TOOLBAR
(pasek_narz),
                        GTK_TOOLBAR_CHILD_TOGGLEBUTTON,
                        NULL,
                        "Podkreślenie", "Podkreślenie", "Podkreślenie",
                        UtworzKontrolkeZXpm (ypole, (gchar **) xpm_podkr),
                       (GtkSignalFunc) KliknietoPrzycisk,
                        "podkreślenie");
}
```

Synchronizacja elementów menu i paska narzędziowego

Przyciski na pasku narzędziowym i elementy menu są z punktu widzenia GTK+ zupełnie odrębnymi kontrolkami. Nie istnieje między nimi żaden związek, o ile nie stworzymy go w kodzie. Skojarzenie dwóch kontrolek polega zazwyczaj na wyznaczeniu tej samej funkcji zwrotnej dla elementów menu i przycisków paska narzędziowego, co powoduje, że kliknięcie menu lub przycisku daje w aplikacji te same rezultaty.

W niektórych programach informacja o statusie pojawia się w kilku miejscach. W tworzonej przez nas aplikacji informacje o parametrach czcionki pojawiają się na pasku narzędziowymi i w zaznaczalnym elemencie menu. Jeśli użytkownik wybierze przycisk „Pogrubienie", to opcja menu „Pogrubienie" powinna odzwierciedlać zmianę stanu. Jednakże synchronizacja pomiędzy przełącznikiem na pasku i zaznaczalnym menu nie odbywa się automatycznie, ponieważ aplikacja nie posiada żadnej wiedzy na temat związku pomiędzy tymi elementami.

Jak uzyskać wymaganą synchronizację? Przypomnijmy sobie, że przyciski paska narzędziowego operujące na czcionce posiadają funkcję zwrotną KliknietoPrzycisk. Funkcja ta wie, który przycisk został wciśnięty i może właściwie ustawić odpowiednik przycisku w menu. Przypomnijmy sobie również, że zachowaliśmy wskaźnik do odpowiedzialnego za wybór czcionki elementu menu, kiedy tworzyliśmy ten element. Możemy sprawdzić stan przełącznika przy pomocy następującego kodu:

```
/* --- Przycisk włączony czy wyłączony? --- */
nStan = GTK_TOGGLE_BUTTON (kontrolka)->active;
```

Kontrolka jest przekształcana na przełącznik, aby można było sprawdzić jego stan. Możemy napisać niewielką funkcję KliknietoPrzycisk, która odpowiednio ustawi element menu:

```
/*
```

```
 * KliknietoPrzycisk
 */
void KliknietoPrzycisk (GtkWidget *kontrolka, gpointer dane)
{
   int nStan = GTK_TOGGLE_BUTTON (kontrolka)->active;

   UstawMenu ((char *) dane, nStan);
}
```

Parametr dane został ustawiony podczas tworzenia kontrolki, aby przekazywała ona do funkcji zwrotnej dane użytkownika. W tym przypadku zmienna dane określa, który przycisk został wciśnięty. Jak pamiętamy, do ustawienia zaznaczalnego elementu menu możemy wykorzystać funkcję gtk_check_menu_item_set_state. Wybrany przycisk przekazuje parametr dane, który zawiera tekst, opisujący wciśnięty przycisk.

```
/*
 * UstawMenu
 *
 * Ustawia zaznaczalny element menu na podstawie nazwy i stanu
 *
 * szPrzycisk - nazwa elementu, który należy zmienić
 * nStan - Stan, w jaki ma znaleźć się element
 */
void UstawMenu (char *szPrzycisk, int nStan)
{
   GtkCheckMenuItem *elem = NULL;
   /* --- pokazujemy, w jaki sposób zmieniamy element menu --- */
   printf ("check_menu_set_state - %d\n", nStan);

   /* --- czy to przycisk Pogrubienie?  --- */

   if (!strcmp (szPrzycisk, "pogrubienie")) {
      elem = GTK_CHECK_MENU_ITEM(menuPogrub);
   } else if (!strcmp (szPrzycisk, "kursywa")) {
      elem = GTK_CHECK_MENU_ITEM(menuKurs);
   } else if (!strcmp (szPrzycisk, "podkreślenie")) {
      elem = GTK_CHECK_MENU_ITEM(menuPodkr);
   }
   if (elem) {
      gtk_check_menu_item_set_state (elem, nStan);
   }
}
```

Menu musi z kolei ustawiać stan przycisku na pasku narzędziowym. Różnica polega na tym, że do ustawienia stanu przycisku używamy funkcji gtk_toggle_button_set_state, i musimy przekształcić kontrolkę przy pomocy GTK_CHECK_MENU_ITEM, aby sprawdzić, który element menu wybrano.

```
void ZaznaczMenu (GtkWidget *kontrolka, gpointer dane)
{
   GtkToggleButton *przycisk = NULL;
   char *szPrzycisk;

   /* --- Sprawdzamy stan elementu menu --- */
   int nStan = GTK_CHECK_MENU_ITEM (kontrolka)->active;

   /* --- pokazujemy parametry i stan przycisku --- */
   szPrzycisk = (char *) dane;
   printf ("wybrane menu %s - %d\n", szPrzycisk, nStan);
   /* --- przełączamy przycisk na pasku narzędziowym  --- */
   if (!strcmp (szPrzycisk, "pogrubienie")) {
      przycisk = GTK_TOGGLE_BUTTON (pasek_pogrub);
   } else if (!strcmp (szPrzycisk, "kursywa")) {
      przycisk = GTK_TOGGLE_BUTTON (pasek_kurs);
   } else if (!strcmp (szPrzycisk, "podkreślenie")) {
      przycisk = GTK_TOGGLE_BUTTON (pasek_podkr);
   }
   if (przycisk) {
      gtk_toggle_button_set_state (przycisk, nStan);
   }
}
```

Obie procedury są bardzo podobne. Obecnie po kliknięciu przycisku, odpowiedzialnego za styl czcionki, uaktualniane jest menu, a po kliknięciu menu uaktualniany jest pasek narzędziowy. Dzięki tego rodzaju kosmetyce aplikacja sprawia wrażenie solidnej. Nie ma nic gorszego, niż wyświetlanie odmiennych stanów przez element menu i pasek narzędziowy; no dobrze, *są* gorsze rzeczy, ale użytkownik aplikacji z pewnością nie byłby tym zachwycony.

Podsumowanie

Aplikacje potrzebują dobrego interfejsu użytkownika. Menu, paski narzędziowe, piksmapy i podpowiedzi pomagają w stworzeniu profesjonalnie wyglądającej aplikacji. Istnieje trudniejszy i łatwiejszy sposób two-

rzenia menu: sposób trudniejszy jest bardziej elastyczny i pozwala na więcej, na przykład na dodanie do menu pól wyboru. Istnieje wiele funkcji, pomagających w dodawaniu elementów do paska narzędziowego. Piksmapy są rysunkami, które można przekształcić w kontrolki piksmapy i wstawić do przycisku, tworząc tym samym przycisk zaopatrzony w ikonę. W rozdziale napisaliśmy krótkie programy, ilustrujące tworzenie szkieletu aplikacji przy pomocy fabryki elementów lub ręcznego kodowania menu.