26

Graficzny interfejs użytkownika w X Window

Interfejs użytkownika opisanych dotąd aplikacji, mówiąc delikatnie, pozostawia co nieco do życzenia: wszystkie polecenia oprócz zmieniania wymiarów okna i położenia obserwatora użytkownik wydaje, naciskając jakiś klawisz. Nie da się w ten sposób wygodnie wprowadzać wielkości analogowych, takich jak parametry oświetlenia lub parametry artykulacji, a zresztą klawiatura bywa potrzebna do wprowadzania napisów (liczb, nazw plików itp.), a wtedy użytkownik powinien na bieżąco widzieć, co pisze. Dlatego w bardziej skomplikowanych aplikacjach potrzebny jest **graficzny interfejs użytkownika** (*GUI, graphical user interface*), czyli rozmaite **wihajstry** (*widgets*), które użytkownik widzi w oknie i za których pośrednictwem może wprowadzać dane i wydawać polecenia. Niestety, biblioteka FreeGLUT ma tylko bardzo ograniczony i niedziałający poprawnie z nowym OpenGL-em (zobacz p. 3.1.1) zestaw procedur realizujących GUI, a w bibliotece GLFW nie ma nawet tego.

Mój kłopot polega na tym, że nie chcę zbytnio oddalać się od kursu OpenGL-a, a jednocześnie nie chcę narażać Czytelników na studiowanie kiepskiego opisu jakiejś biblioteki GUI, której akurat nie mają i z rozmaitych powodów nie mogą sobie zainstalować. Oczywiście, można stworzyć znakomity GUI w aplikacji FreeGLUT-a lub GLFW, w którym wihajstry rysuje OpenGL, ale (wobec konieczności dostarczenia odpowiednich szaderów i utworzenia buforów z danymi opisującymi wihajstry) jest to dużo bardziej pracochłonne niż pouczające. Jeśli więc obrazy wihajstrów nie przedstawiają skomplikowanych obiektów trójwymiarowych, to łatwiej jest użyć dostępnych procedur grafiki dwuwymiarowej i rysować wihajstry w (znacznie prostszym do użycia) trybie natychmiastowym. Obrazy większości wihajstrów są na tyle nieskomplikowane, że czas ich rysowania będzie niezauważalny.

Ryzykując utratę zainteresowania Czytelników, których komputery nie znają dobrodziejstw systemu X Window, a także tych, którzy już mają swoją ulubioną bibliotekę GUI, postanowiłem napisać swój zestaw procedur, rysujący wihajstry za pomocą biblioteki Xlib. Decydując się na użycie tej biblioteki do rysowania, muszę też użyć zawartych w niej procedur obsługi komunikatów o zdarzeniach wejściowych i użyć biblioteki GLX do utworzenia kontekstu OpenGL-a. Mam nadzieję, że żaden Czytelnik nie będzie przez tę decyzję czuł się

pokrzywdzony bardziej niż inni[1]. Opisany tu mechanizm komunikowania się aplikacji z wihajstrami może być użyty do zaimplementowania wihajstrów rysowanych przez OpenGL-a. W razie potrzeby można też dorabiać nowe rodzaje wihajstrów — trzeba tylko wymyślić dla każdego z nich wygląd i sposób działania.

26.1. Struktury danych i procedury podstawowe

Do zrealizowania wihajstra potrzebne są dwie procedury[2]. Pierwsza z nich przetwarza wejście (tj. reaguje na komunikaty o działaniach użytkownika), a druga wyświetla odpowiedni obraz w oknie, aby użytkownik widział, gdzie ma umieścić kursor przed naciśnięciem przycisku myszy albo w jakim wihajster jest stanie (np. czy wihajster — przełącznik — jest w danej chwili włączony).

Listing 26.1 przedstawia typy danych zdefiniowane w celu zaimplementowania GUI. Każdy wihajster jest opisany przez strukturę typu `xwidget`, której pola to: `id` — identyfikator wihajstra, `r` — opis prostokąta zajmowanego przez wihajster w oknie, `state` — stan wihajstra, `input` i `redraw` — wskaźniki procedury przetwarzającej komunikaty wejściowe i procedury rysującej wihajster, `wm` — wskaźnik struktury menu okna, w którym wihajster ma się pojawić, `link` — para wskaźników tworzących listę wihajstrów tego menu, oraz `data0`, `data1` — wskaźniki danych specyficznych dla wihajstra konkretnego rodzaju.

Struktura typu `xwinmenu` reprezentuje zbiór wihajstrów należących do danego okna (lub podokna) utworzonego przez system X Window. Pole `window` jest identyfikatorem okna. Pole `pixmap` zawiera identyfikator **kanwy** (*pixmap*), na której odbywa się rysowanie wihajstrów; można by je rysować bezpośrednio w oknie, ale choć to zabiera znikomą ilość czasu, byłoby widoczne migotanie (spowodowane wyświetlaniem w oknie niedokończonych obrazów). Dlatego wihajstry mają być rysowane na tej kanwie, a jej zawartość będzie kopiowana do okna, gdy obrazy wszystkich wihajstrów będą gotowe. Pole `r` opisuje wymiary okna (i kanwy). W polach `prevx`, `prevy` i `prevmask` będą pamiętane położenia kursora w oknie i stan przycisków myszy *po* zakończeniu obsługi komunikatu, aby można ich było użyć podczas obsługi następnego komunikatu. Pole `changed` ma przypisywaną wartość niezerową, gdy któryś wihajster lub aplikacja sygnalizuje potrzebę odświeżenia obrazu w oknie. Pole `expose_sent` służy do tego, aby zapobiegać wysyłaniu do okna niepotrzebnych komunikatów `Expose` podczas przetwarzania komunikatów o przesunięciu kursora, co będzie wyjaśnione dalej. Wskaźnik `data` jest przeznaczony do użytku aplikacji. Pole `wlist` jest nagłówkiem listy dwukierunkowej wihajstrów. Pole `redraw` jest wskaźnikiem procedury rysującej zawartość okna, czyli tło i na nim wszystkie wihajstry. Procedura ta ma używać do rysowania *albo* procedur systemu X Window (z biblioteki Xlib), *albo* OpenGL-a, przy czym ten sam sposób rysowania w oknie obowiązuje procedury rysujące *wszystkie* wihajstry w tym oknie. W tym rozdziale opisałem tylko najprostsze przykłady wihajstrów rysowanych przez

[1] Użytkownikom Windowsów pozostaje użycie jakiegoś gotowego oprogramowania lub własnoręczne napisanie działającej w tym środowisku biblioteki, naśladującej w ostateczności moje rozwiązanie dla systemu X Window. To też może być dobra zabawa, życzę powodzenia.

[2] W języku C++ wihajster powinien być obiektem z dwiema metodami wirtualnymi.