Aleksandra Niezbrzycka

Tworzenie aplikacji okienkowych w R z wykorzystaniem pakietu gWidgets

Abstrakt

Użytkownicy bardzo często są przyzwyczajeni do pracy z programami w trybie graficznym, najczęściej okienkowym, gdzie przy pomocy myszy „wyklikują" potrzebne operacje. Wykorzystanie bibliotek dostarczanych przez R daje nieograniczone dla programisty możliwości łatwego tworzenia własnych interfejsów graficznych. Dzięki powstaniu licznych narzędzi do tworzenia aplikacji okienkowych, zwiększyła się także liczba nakładek, które można wykorzystywać w różnych zastosowaniach.

Celem artykułu jest przedstawienie możliwości pakietu **gWidgets** w perspektywie tworzenia środowiska graficznego w R[1].

Słowa kluczowe: gWidgets, R, aplikacje okienkowe, GUI, interfejs graficzny, programowanie

Wstęp

Program R dostarcza użytkownikowi możliwości komponowania własnych elementów graficznych, okien i kontrolek dostosowanych do jego potrzeb. W odróżnieniu od innych programów, baza R nie jest wyposażona w GUI co znacznie zwiększa jego ergonomię. Przy pomocy specjalnych pakietów, które można zainstalować, możliwe jest tworzenie własnego środowiska graficznego, specjalnie dopasowanego do potrzeb użytkownika.

Samo tworzenie aplikacji okienkowych w R jest dość intuicyjne. Za każdym razem polega ono na stworzeniu komponentu, umiejscowieniu go w odpowiednim miejscu okienka programu i w końcu na oprogramowaniu jego funkcji.

W dalszych częściach artykułu zakładamy, że program R został poprawnie zainstalowany. Wszystkie przedstawione skrypty testowane były w wersji 2.11.1 programu R w środowisku Windows i Linux.

[1] www.cran.r-project.org

Instalacja i ładowanie biblioteki gWidgets

Język R dostarcza wielu pakietów umożliwiających tworzenie aplikacji okienkowych. Wykorzystany przy pisaniu skryptów zawartych w artykule pakiet gWidgets, jest jednym z najpopularniejszych ze względu na łatwość użycia i za razem duże możliwości pakietem do tworzenia interfejsu użytkownika z wykorzystaniem okien. Co istotne, pozwala on na wykonanie aplikacji okienkowych niezależnie od systemu operacyjnego[2]. Pakiety wspomagające, w oparciu o które można budować tego typu aplikacje to:

- `RGtk2` – użytkownicy systemów operacyjnych Mac i Windows najprawdopodobniej będą poproszeni o doinstalowanie GTK+ (biblioteki służącej do tworzenia interfejsu graficznego do programów komputerowych),
- `rJava` – użycie tej biblioteki spowoduje, że wygląd okien będzie charakterystyczny dla aplikacji napisanych w języku Java,
- `tcltk` – w przypadku instalacji tej biblioteki, użytkownicy systemu operacyjnego Mac będą zmuszeni zainstalować wersję 8.5 Tk

Aby rozpocząć pracę w gWidgets należy, o ile nie zrobiono tego wcześniej, zainstalować bibliotekę:

```
> install.packages("gWidgetsRGtk2", dep=TRUE)
```

Dzięki argumentowi dep=TRUE, R sam doinstalowuje pakiety, które są mu potrzebne do poprawnej pracy z gWidgets.

Komendy `options` oraz `library` umożliwiają załadowanie biblioteki:

```
> options(guiToolkit = "RGtk2")
> library("gWidgets")
```

Tworzenie prostych okien dialogowych i obsługa komponentów gWidgets

Okienka dialogowe tworzone w pakiecie gWidgets są modalne. Oznacza to, że wstrzymują przetwarzanie zdarzeń przez pozostałe okna aplikacji. Innymi słowy, użytkownik nie może przełączyć się na żadne inne okno dopóki nie wyłączy okna przyciskiem do tego dedykowanym.

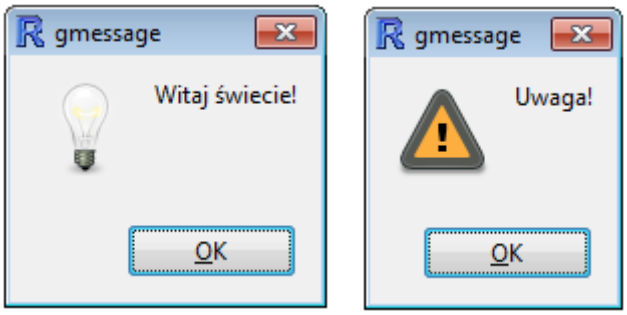
[2] Oprac. P. Szymański, Zintegrowane Systemy Statystyczne II, Bydgoszcz 2010, s. 1

Przedstawione zostały przykłady prostych okien dialogowych:

Przykład 1

```
> gmessage("Witaj świecie!", title="gmessage")
> gmessage("Uwaga!", title="gmessage", icon="warning")
```

Rysunek 1: Przykład komponentu gmessage

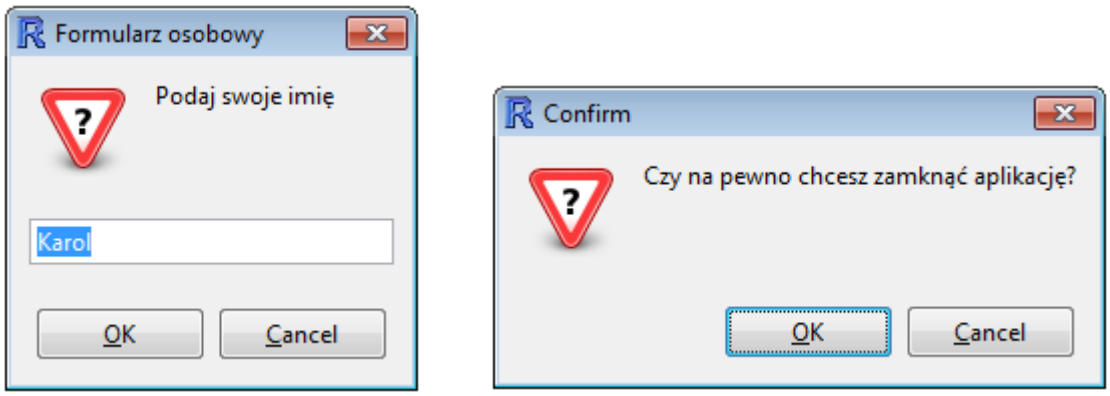


Argument title pozwala ustawić nazwę pojawiającego się okna, argument icon umożliwia wyświetlanie dodatkowo ikonki.

Przykład 2

```
> ginput("Podaj swoje imię", text="Karol", title="Formularz osobowy",
  icon="question")
> gconfirm("Czy na pewno chcesz zamknąć aplikację?", title="gconfirm")
```

Rysunek 2: Przykład okna dialogowego z polem tekstowym oraz komponentu gconfirm



Argument text umożliwia wpisanie tekstu, który pojawi się domyślnie w niewielkim polu tekstowym po uruchomieniu aplikacji. Okno dialogowe gconfirm pokazuje komunikat z przyciskami OK. i anulacji.

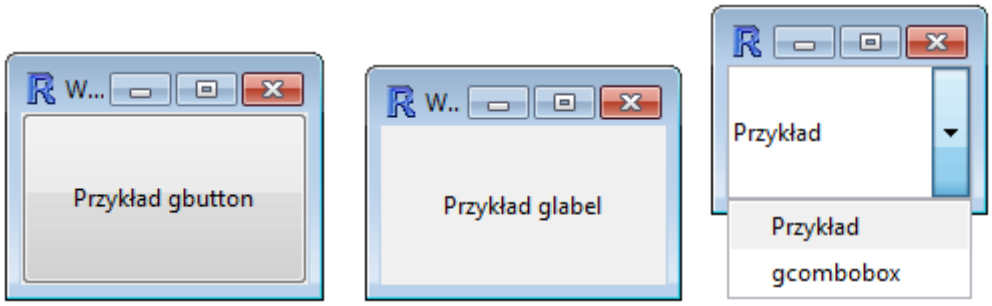
Pakiet gWidgets daje możliwość tworzenia komponentów (kontrolek) oraz ich obsługę. Poniżej znajduje się kilka prostych instrukcji prezentujących sposób tworzenia kontrolek.

Przykład 3

```
> kontrolka1 <- gbutton("Przykład gbutton", cont=TRUE)
> kontrolka2 <- glabel("Przykład glabel", cont=TRUE)
> kontrolka3 <- gedit("Przykład gedit", cont=TRUE)
> kontrolka4 <- gtext("Przykład gtext", cont=TRUE)
> kontrolka5 <- gcheckbox("Przykład gcheckbox", cont=TRUE)
> kontrolka6 <- gradio(c("Przykład","gradio"), cont=TRUE)

> kontrolka7 <- gcombobox(c("Przykład","gcombobox"), cont=TRUE)
> kontrolka8 <- gtable(c("Przykład","gtable"), cont=TRUE)
```

Rysunek 3: Przykłady komponentów (kontrolek)



W dwóch ostatnich przykładach widać użycie metody c(). Jest to funkcja, która łączy ze sobą argumenty i tworzy z nich wektor. Argument cont=TRUE można zastąpić także argumentem cont=gwindow(). Obydwa argumenty sprawiają, że komponent będzie pojawiał się w nowym oknie.

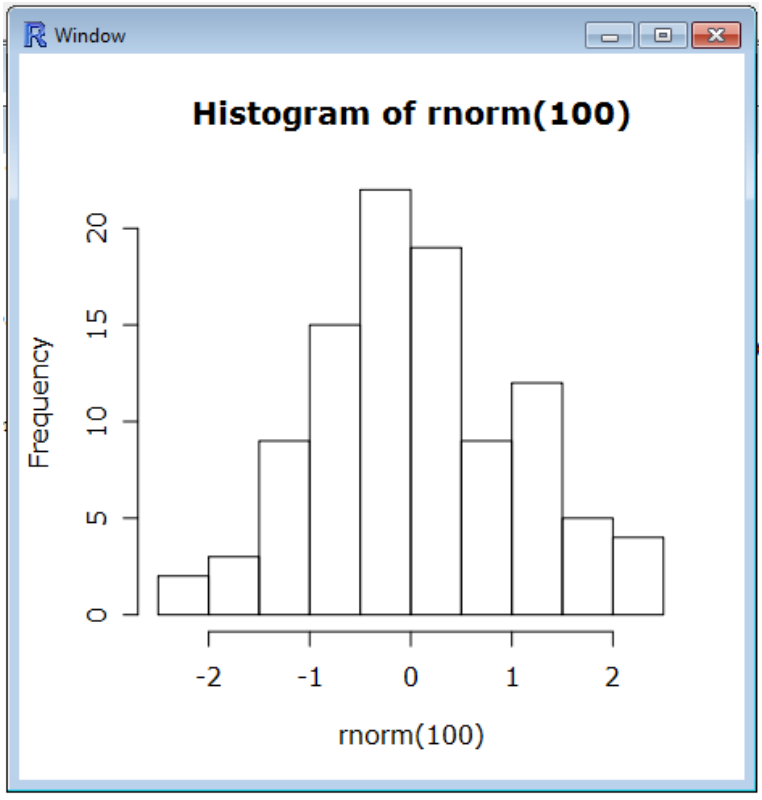
Pakiet gWidgets umożliwia wyświetlanie grafiki w osobnym oknie aplikacji. Należy jednak pamiętać, że jest to możliwe tylko przy użyciu `RGtk2`. Biblioteki `tcltk` oraz `Java` nie dają takiej możliwości.

Poniżej przedstawione zostanie wyświetlanie rozkładu normalnego za pomocą histogramu przy użyciu komponentu `ggraphic`.

Przykład 4

```
> ggraphics(cont=TRUE)
> hist(rnorm(100))
```

Rysunek 4: Przykład użycia komponentu ggraphic



Tworzenie zaawansowanych aplikacji okienkowych

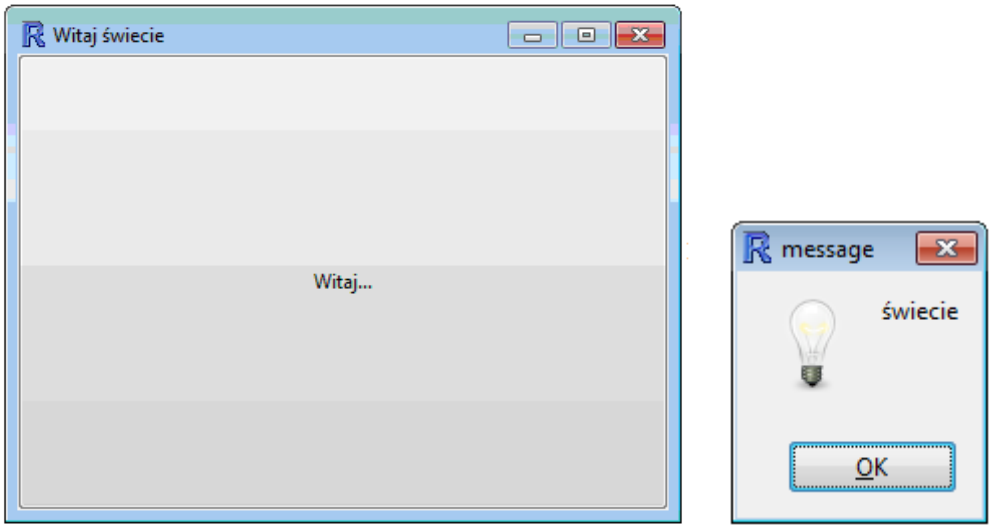
Ważnym atrybutem każdego komponentu jest `handler`, który służy do reagowania na zdarzenia zainicjowane przez użytkownika. Może to być np. kliknięcie myszy, wciśnięcie klawisza Enter lub jakiegokolwiek innego klawisza. Każda kontrolka posiada swój domyślny

handler (np. w gedit jest to klawisz Enter). Tworzenie własnego handler'a może odbywać się na dwa sposoby.
Poniższy przykład ilustruje pierwszy ze sposobów polegający na bezpośrednim zdefiniowaniu funkcji handlera w definicji obiektu.

Przykład 5

```
> win <- gwindow("Witaj świecie", visible=TRUE)
> obj <- gbutton("Witaj...",container=win, handler= function(h,...) +
gmessage("świecie"))
```

Rysunek 5: Przykład tworzenia handlera, komponenty gwindow i gmessage



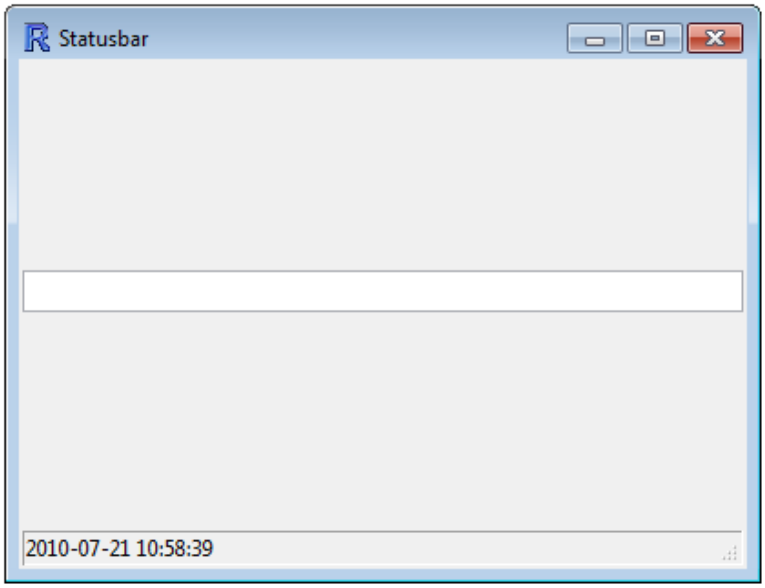
Drugim ze sposobów jest wykorzystanie funkcji z rodziny addHandlerXXX, gdzie XXX oznacza konkretne zdarzenie.
W poniższym przykładzie użyty został handleridle aktualizujący komponent statusbar co sekundę do czasu systemowego.

Przykład 6

```
> window = gwindow("Statusbar")
> g = ggroup(horizontal=FALSE, container=window)
> obj <- gedit("", cont=g, expand=T)
> sb <- gstatusbar(as.character(Sys.time()), cont=g)
> addhandleridle(obj, handler=function(h,...){
+ svalue(sb) <- as.character(Sys.time())
```

```
+ },
+ interval=1000)
```

Rysunek 6: Przykład statusbar



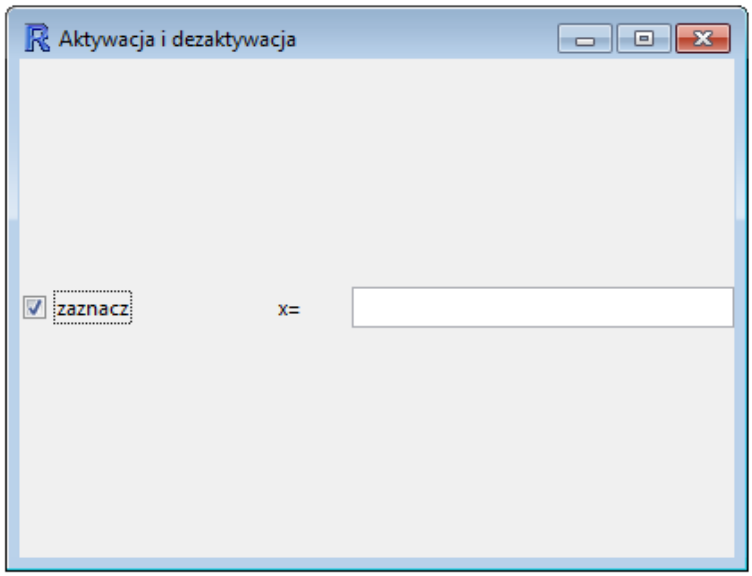
Użyty w przykładzie 6 kontener ggroup służy do „pakowania” poszczególnych komponentów w poziomie lub w pionie i zostanie on dokładnie omówiony w dalszej części.

Poszczególne komponenty można dezaktywować w taki sposób aby nie przetwarzały żadnych danych, służy do tego polecenie enabled (). Nieaktywny komponent zmienia wówczas swój kolor na szary.
W poniższym przykładzie, kontrolka gedit została domyślnie ustawiona jako nieaktywna. Kliknięcie w komponent gcheckbox aktywuje ją bądź dezaktywuje.

Przykład 7

```
> window = gwindow("Aktywacja i dezaktywacja")
> kontr1 = gcheckbox("zaznacz", checked = FALSE, cont=window)
> kontr2 = glabel("x=", cont=window)
> kontr3 = gedit("", cont=window)
> enabled(kontr3) <- FALSE
> addHandlerClicked(kontr1, handler = function(h,...) {
+  enabled(kontr3) <- svalue(kontr1)
+ })
```

Rysunek 7: Aktywacja i dezaktywacja komponentu gedit



Tworząc zaawansowane aplikacje okienkowe, bardzo często niezbędną rzeczą jest dodanie menu lub paska narzędzi. W gWidgets są one określone poprzez tworzenie list.

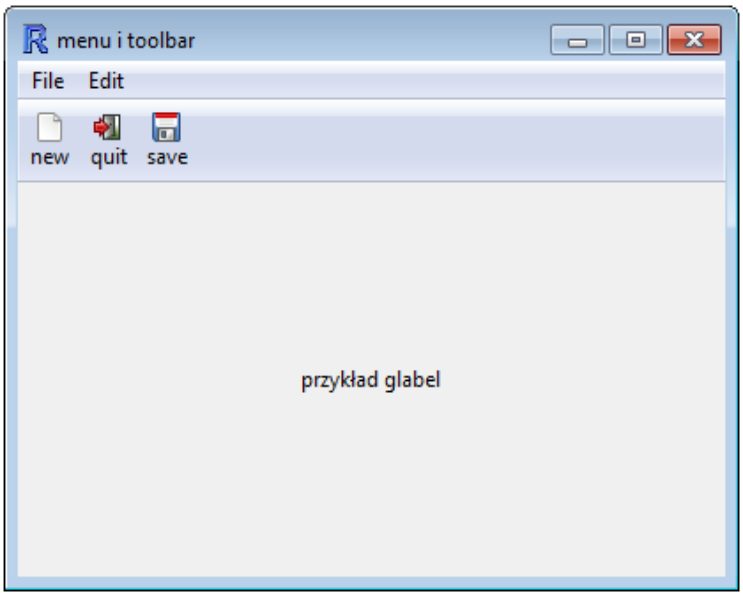
Przykład 8 przedstawia tworzenie prostego menu oraz paska zadań. Po kliknięciu w odpowiednie ikonki wyświetla się komponent glabel.

Przykład 8

```
> defHandler = function(h,...) glabel("hej!", cont=TRUE)
> menulist = list(
+   File = list(
+     openFile = list(handler=defHandler, icon="open"),
+     close = list(handler=defHandler, icon="cancel")
+     ),
+   Edit = list(
+     paste = list(handler=defHandler),
+     copy =  list(handler=defHandler)
+     )
+   )
> toolbarlist = list(
+   new = list(handler = defHandler, icon="new"),
+   quit = list(handler= defHandler, icon="quit"),
+   save = list(handler= defHandler, icon="save")
```

```
+    )
>  window = gwindow("menu i toolbar")
>  g = ggroup(cont = window, horizontal = FALSE)
>  mb = gmenu(menulist, cont=window)
>  tb = gtoolbar(toolbarlist, cont=window)
>  g1 = ggroup(cont=g, expand=TRUE, horizontal = FALSE)
>  l <- glabel("przykład glabel", cont=g1, expand=TRUE)
>  size(l) <- c(100,200)
```

Rysunek 8: Menu i toolbar



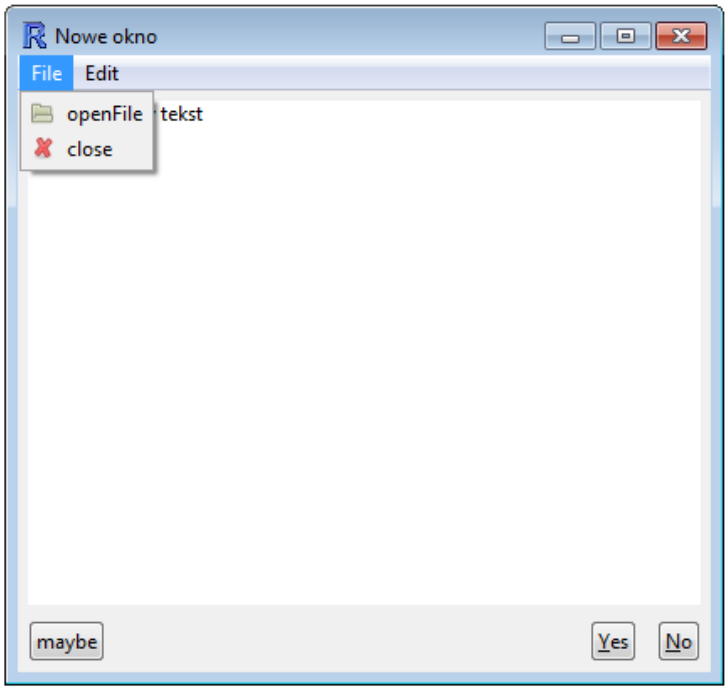
Zastosowany w przykładzie kontener ggroup pakuje kontrolki lub inne kontenery w poziomie lub w pionie (argument horizontal).

Poniższy przykład również prezentuje zastosowanie kontenera ggroup, jednak co ważniejsze przedstawia użycie funkcji addSpring, która rozsuwa kontrolki do krańcowych krawędzi okna. Po wciśnięciu dowolnego przycisku okno aplikacji zamyka się (funkcja dispose(window))

Przykład 9

```
> defHandler = function(h,...) dispose(window)
>   menulist = list(
+     File = list(
+       openFile = list(handler=defHandler, icon="open"),
+       close = list(handler=defHandler, icon="cancel")
+       ),
+     Edit = list(
+       paste = list(handler=defHandler),
+       copy =  list(handler=defHandler)
+       )
+     )
>
>   window = gwindow("Nowe okno")
>   g = ggroup(cont = window, horizontal = FALSE)
>   menu = gmenu(menulist, cont=window)
>   g1 = ggroup(cont=g, expand=TRUE, horizontal = FALSE)
>
>   g = ggroup(horizontal = FALSE, cont=g1)
>   l <- gtext("Przykładowy tekst", cont=g, expand = TRUE)
>   size(l) <- c(400,300)
>
>   przyciski = ggroup(horizontal = TRUE, cont=g1)
>   maybe = gbutton("maybe", cont=przyciski)
>   addSpring(przyciski)
>   yes = gbutton("yes", cont=przyciski)
>   addSpace(przyciski, 2)
>   no = gbutton("no", cont=przyciski)
```

Rysunek 9: Przykład zastosowania kontenera ggroup

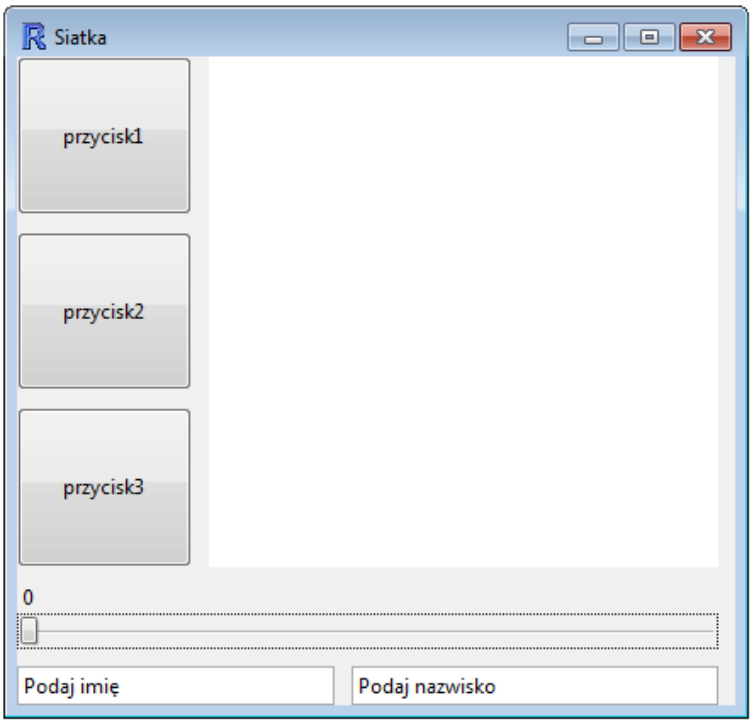


Jednym z najłatwiejszych w obsłudze i zarazem dającym duże możliwości kontenerem umożliwiającym precyzyjne ułożenie kontrolek jest glayout. Kontener tworzy wirtualną siatkę, na której rozmieszczane są poszczególne komponenty. Odwoływanie do poszczególnych komórek siatki odbywa się na tej samej zasadzie jak odwoływanie się tabeli dwuwymiarowej.

Przykład 10

```
> window = gwindow("Siatka")
> tbl <- glayout(cont = window)
> tbl[1:3,2:4] <- (tb <- gtext("", cont=tbl))
> size(tb) <- c(300,300)
> tbl[1,1] <- (Button1 <- gbutton("przycisk1", cont=tbl))
> tbl[2,1] <- (Button2 <- gbutton("przycisk2", cont=tbl))
> tbl[3,1] <- (Button3 <- gbutton("przycisk3", cont=tbl))
> tbl[4,1:4] <- (Spin1 <- gslider (from = 0, to = 100, by = 1,  horizontal =
  TRUE))
> tbl[5,1:2] <- (Edit1 <- gedit ("Podaj imię", cont=tbl))
> tbl[5,3:4] <- (Edit2 <- gedit ("Podaj nazwisko", cont=tbl))
```

Rysunek 10: Wykorzystanie kontenera glayout



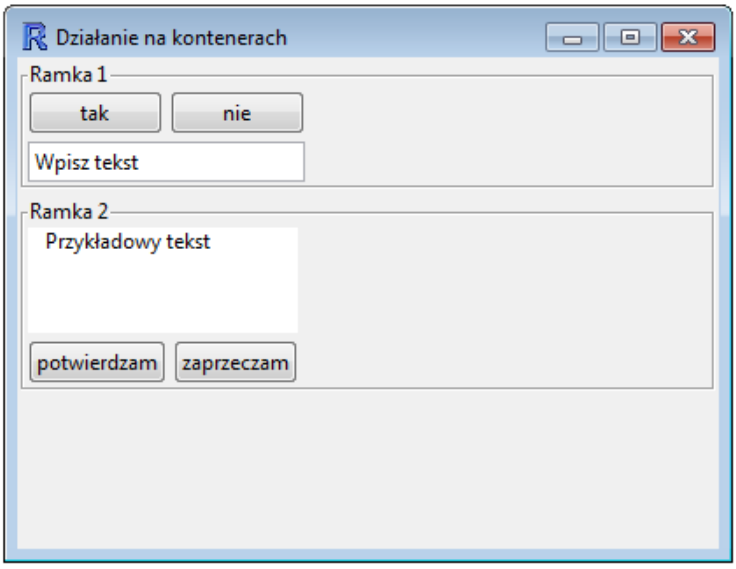
Poszczególne kontenery można także „pakować” do innych kontenerów. Poniżej przedstawiono zagnieżdżenie kontenera glayout w innym kontenerze gframe.

Przykład 11

```
> window <- gwindow("Działanie na kontenerach")
> window2 <- ggroup(horizontal=F, cont=window)
> ramka1 <- gframe("Ramka 1", con=window2)
> ramka2 <- gframe("Ramka 2", con=window2)
>  manage1 <- glayout(cont=ramka1, spacing=4)
>  manage2 <- glayout(cont=ramka2, spacing=4)
> przycisk1 <-gbutton("tak")
> przycisk2 <-gbutton("nie")
> edit1 <-gedit("Wpisz tekst")
>  manage1[1,1] <- przycisk1
>  manage1[1,2] <- przycisk2
>  manage1[2,1:2] <- edit1
> text1 <-gtext("Przykładowy tekst")
> przycisk3 <-gbutton("potwierdzam")
```

```
> przycisk4 <-gbutton("zaprzeczam")
>  manage2[1,1:2] <- text1
>  manage2[2,1] <- przycisk3
>  manage2[2,2] <- przycisk4
```

Rysunek 11: Przykład działania na kontenerach



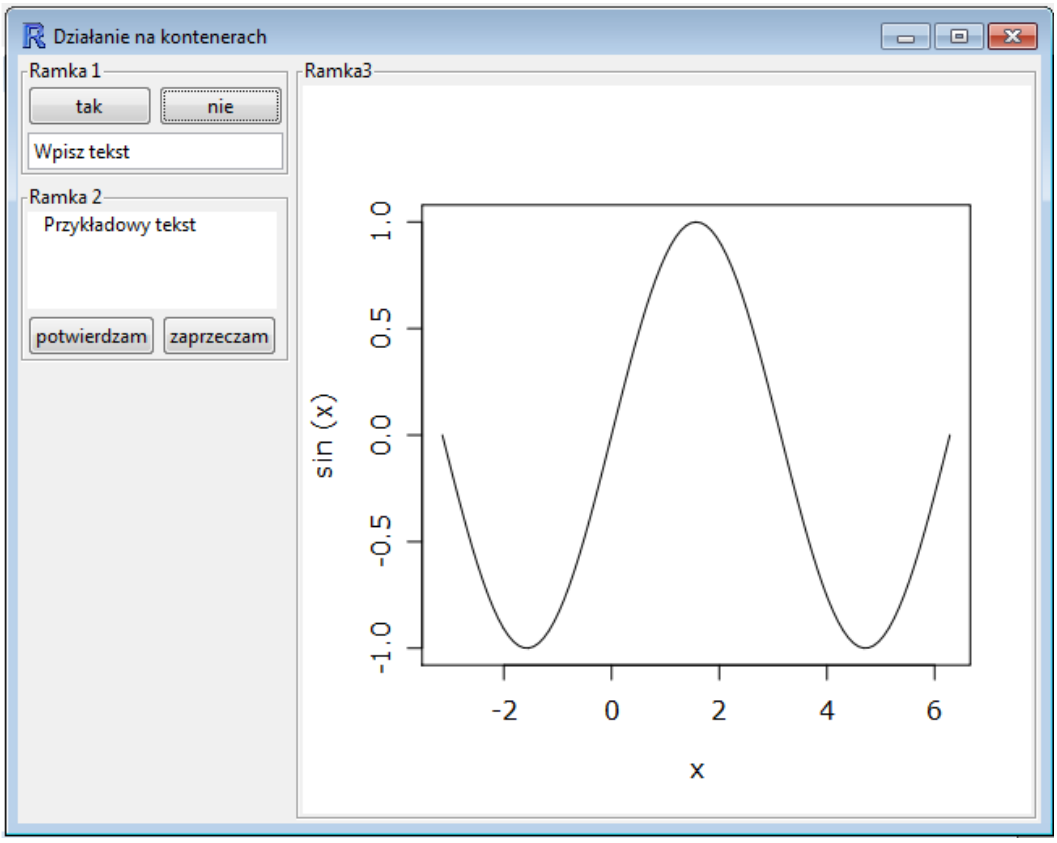
Grafika również może być umieszczana w osobnym kontenerze. Aby wykres wyświetlał się jednak w tym samym oknie co reszta komponentów, wymagane jest zainstalowanie i załadowanie dodatkowo bibliotek: gWidgetstcltk, tcltk oraz cairoDevice.

Przykład 12

```
>  library("gWidgetstcltk")
>  library("tcltk")
>  library("cairoDevice")
>  window <- gwindow("Działanie na kontenerach")
>  window2 <- ggroup(horizontal=F, cont=window)
>  window3 <-ggroup(horizontal=T, cont=window)
>   ramka1 <- gframe("Ramka 1", con=window2)
>   ramka2 <- gframe("Ramka 2", con=window2)
>   ramka3 <- gframe("Ramka3", cont=window3)
>    manage1 <- glayout(cont=ramka1, spacing=4)
>    manage2 <- glayout(cont=ramka2, spacing=4)
```

```
>    manage3 <- glayout(cont=ramka3, spacing=4)
>   przycisk1 <-gbutton("tak")
>   przycisk2 <-gbutton("nie")
>   edit1 <-gedit("Wpisz tekst")
>    manage1[1,1] <- przycisk1
>    manage1[1,2] <- przycisk2
>    manage1[2,1:2] <- edit1
>   text1 <-gtext("Przykładowy tekst")
>   przycisk3 <-gbutton("potwierdzam")
>   przycisk4 <-gbutton("zaprzeczam")
>    manage2[1,1:2] <- text1
>    manage2[2,1] <- przycisk3
>    manage2[2,2] <- przycisk4
>   wykres1 <-ggraphics()
>    manage3[1,1] <-wykres1
>     plot(sin, -pi, 2*pi)
```

Rysunek 12: Przykład działania na kontenerach z wykorzystaniem wykresu



Podsumowanie

W obecnych czasach, kiedy użytkownicy przyzwyczajeni są do obsługi programów za pomocą myszy oraz do „wyklikiwania” potrzebnych operacji, tworzenie aplikacji posiadających interfejs graficzny jest po prostu niezbędne.
Wykorzystanie odpowiednich bibliotek dostarczanych przez program R daje nieograniczone możliwości tworzenia własnych aplikacji okienkowych. Samo komponowanie własnych elementów graficznych, okien, kontrolek dostosowanych do potrzeb użytkownika jest w R bardzo intuicyjne i nie wymaga zaawansowanych umiejętności programistycznych. Zaprezentowane w artykule skrypty mogą stanowić dobry wstęp do zapoznania się możliwościami pakietu **gWidgets**, który jest tylko jednym z kilku innych pakietów służących do tworzenia środowiska graficznego w R.

Bibliografia

Biecek Przemysław, Przewodnik po pakiecie R, wyd. 1, Wrocław, 2008, ISBN: 978-83-89020-79-6

Komsta Łukasz, Wprowadzenie do środowiska R, 2004

Kopczewska Katarzyna, Kopczewski Tomasz, Wójcik Piotr, Metody ilościowe w R, wyd. 1, Warszawa, 2009, ISB: 978-83-7556-150-0

Szymański Piotr, Zintegrowane Systemy Statystyczne II Pakiet R, wyd.1, Bydgoszcz, 2010

Verzani John, Examples for gWidgets, 2010

http://cran.r-project.org/