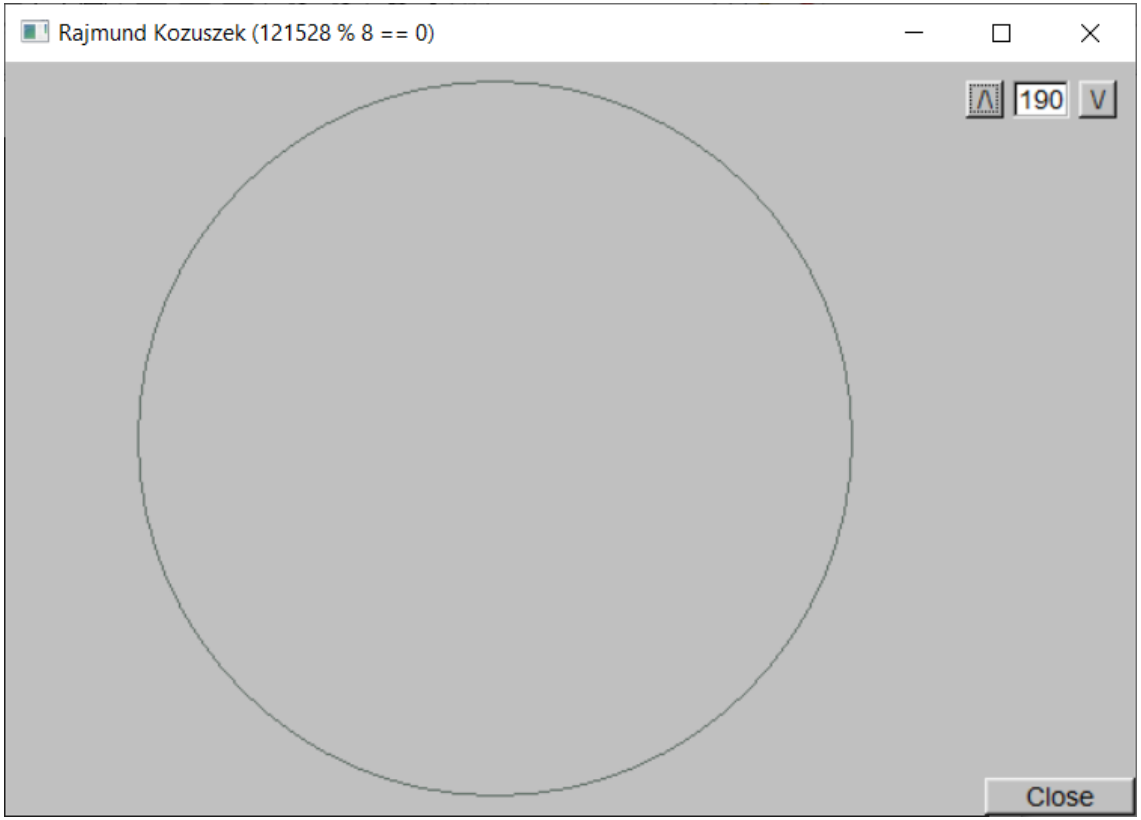
Interfejs użytkownika

Efekt końcowy wykonania trzeciego zadania najlepiej podsumowuje obraz okna aplikacji:



Uwaga: to jeden z 8 układów interfejsu zależnych od numeru indeksu (reszta z dzielenia przez 8 mojego wynosi 0 i dlatego taki układ).

Elementy składowe

Ponieważ manipulujemy elementami interfejsu w oknie proste okno (`SimpleWindow`), którego używaliśmy do tej pory, nie wystarczy. W programie trzeba użyć klasy `Window` zdefiniowanej w bibliotece `Graph_lib`.
Najważniejszym elementem projektu jest zestaw dwóch przycisków i pola tekstowego. Te trzy elementy interfejsu mają być zebrane w obiekcie jednej klasy (niech klasa nazywa się `Spinner`).
Ze względu na to, że będziemy potrzebować nieco niestandardowego działania przycisków kręciołka (*spinner*), warto przygotować sobie stosowną klasę dziedziczącą po `Button`.
Tytułem podpowiedzi, ujawnię dwa dość istotne element całego programu, który obsługuje okno z poprzedniej strony.

Jak współdziałają ze sobą elementy?

Jedynym elementem, który inicjuje akcje jest przycisk. Jego naciśnięcie powoduje wywołanie funkcji zwrotnej (*callback*). Jak wiemy z wykładu, w funkcji zwrotnej dostajemy dwa parametry, a drugi z nich umożliwia przekazanie danych użytkownika. W `graph_lib::Button` jest to wskazanie na okno, w którym jest osadzony przycisk. My potrzebujemy tutaj wskazania na kręciołek, który dalej zajmie się przetwarzaniem akcji (sprawdzi, czy promień nie wychodzi poza dopuszczalny przedział, zmodyfikuje jego wartość, zmieni wartość wyświetlaną w edytorze, narysuje kółko o większym promieniu). Własną klasę przycisku potrzebujemy po to, żeby w czeluści `fltk` przekazać informację: to jest wskazanie, które chcę dostać jako drugi parametr funkcji zwrotnej.

Elementy składowe - myButton

Do ustalenia, który z elementów daje pierwszy parametr dla funkcji zwrotnej (callback) posłużymy się programem uruchomieniowym (debugger).
Okazuje się, że drugi parametr, to wartość składowej pw w klasie Button a dalej w klasie Widget, po której Button dziedziczy.

W GUI.h jest na temat tych pól następująca informacja:

```
protected:
    Window* own;     // every Widget belongs to a Window
    Fl_Widget* pw;   // connection to the FLTK Widget
```

Właśnie ze względu na to, że pw jest kwalifikowane jako protected potrzebna nam będzie własna klasa przycisku – będzie można wygodnie sterować parametrem funkcji zwrotnej.

Podpowiedź I – `MyButton::attach`

Dlaczego dostęp do `pw` jest istotny? Otóż dzięki wywołaniu odpowiedniej funkcji będziemy mogli przekazać informację o tym, że jako drugi parametr typu `Address` w funkcji callback chcemy widzieć wskazanie na macierzysty obiekt `Spinner`, czyli „rodzica”. Wygodnym miejscem na przekazanie tej informacji jest funkcja `attach`, którą trzeba po to w klasie `MyButton` zaimplementować:

```
void attach(Graph_lib::Window& wnd)
{
    Button::attach(wnd); // wywolanie attach z klasy bazowej
    // przekazanie informacji do fltk o drugim parametrze callback
    pw->callback(reinterpret_cast<Fl_Callback*>(do_it), pSpinner);
}
```

Podpowiedź I – main

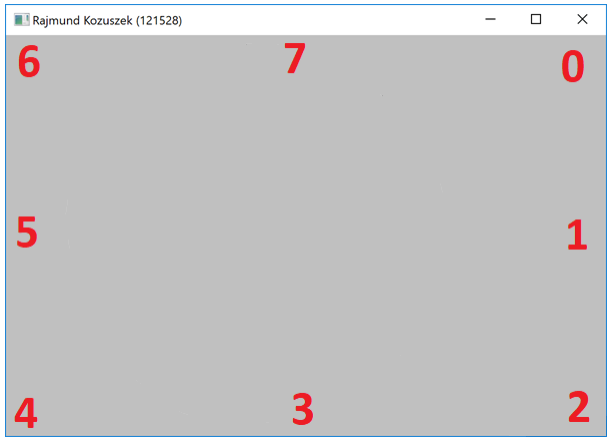
```
int main()
{
   Graph_lib::Window wnd(Point(100, 100), 600, 400,
      "Rajmund Kozuszek (121528)");
   Button btn(Point(wnd.x_max() - 80, wnd.y_max() - 20), 80,
      20, "Close", cb_close);
   wnd.attach(btn);

   Spinner spin(Point(wnd.x_max() - 90, 10), 100);
   spin.attachTo(wnd);

   gui_main();
   return 0;
}
```

Podpowiedź II – fragment Spinner

```
class Spinner
{
   /* ....  */
public:
   Spinner(Point loc, int startVal)
      :  btn_up(loc, 20, 20, "/\\", Spinner::cb_up, this),
         btn_down(Point(loc.x + 60, loc.y), 20, 20, "\\/",
            Spinner::cb_down, this),
         radiusBox(Point(loc.x + 25, loc.y), 30, 20, ""),
         curVal(startVal)
         {}
   void attach(Graph_lib::Window& wnd);
};
```

Położenie spinnera

Położenie spinnera jest zależne od reszty z dzielenia numeru indeksu przez 8:



Co istotne, przycisk `Close` powinien znajdować się przy tej samej krawędzi okna, co `spinner`. W czterech przypadkach jest możliwość wyboru, w czterech krawędź, do której przylega `Close` jest ściśle określona przez `spinner`.

Zadanie można realizować w parach, o ile obie osoby w parze mają tak samo podzielny numer indeksu ☺

Oddawanie

Rysowanie koła (powinno mieć promień ustawiony przez kręciołek) jest warte jeden punkt – tzn. bez koła można zaliczyć ćwiczenie, oczywiście jeśli kręciołek działa poprawnie, ale za maksymalnie 4 punkty. Proszę zwrócić uwagę na ograniczenie promienia – nie będzie dobrze, jeśli koło będzie dotykać krawędzi okna, lub co gorsza zachodzić na `spinner` lub przycisk `Close`.

Definicje wszystkich klas proszę umieścić w pliku `zad_3.cpp`. Ten plik proszę należy złożyć w `Moodle` do:

16 kwietnia 2023 23:59