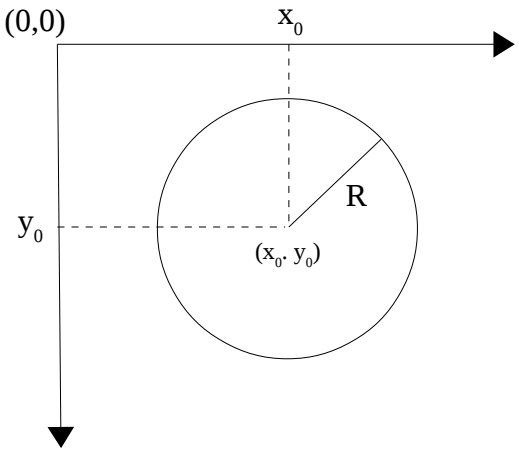
Instrukcja laboratoryjna **4**	**Grafika komputerowa 2D**
	Temat: Rysowanie okręgów oraz algorytmy wypełniania
	Przygotował: dr inż. Grzegorz Łukawski, mgr inż. Maciej Lasota, mgr inż. Tomasz Michno

1 Wstęp teoretyczny

1.1 Rysowanie okręgu

Algorytm rysowania okręgu polega na obliczaniu kolejnych pozycji pikseli zgodnie z poniższym wzorem:

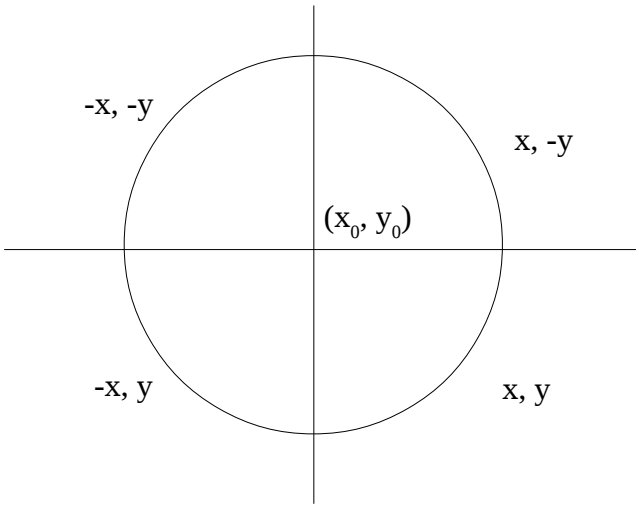
$$x = x_0 + R \cdot \cos(\alpha)$$
$$y = y_0 + R \cdot \sin(\alpha)$$
$$\alpha \in \langle 0, 2\pi \rangle$$



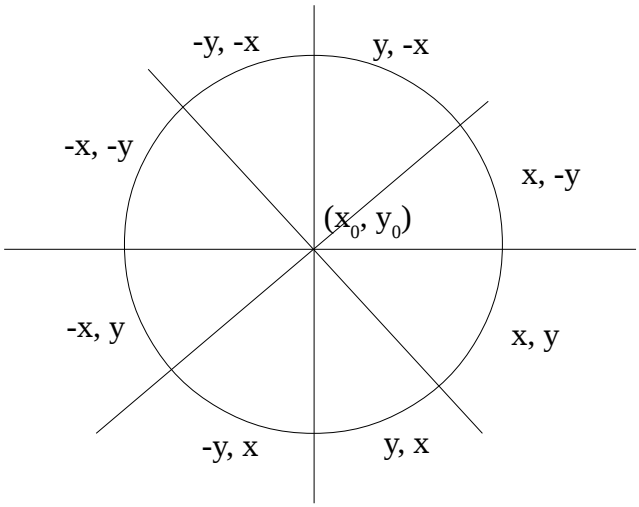
W celu narysowania okręgu wystarczy rysować punkty zgodnie z powyższym wzorem, zwiększając za każdym razem kąt α z krokiem 1/R w celu uniknięcia przerw pomiędzy pikselami.

Czterokrotna i ośmiokrotna symetria

W celu przyśpieszenia algorytmu i zmniejszenia liczby obliczeń można zastosować tzw. czterokrotną lub ośmiokrotną symetrię. Czterokrotna symetria polega na podziale okręgu na cztery równe części, natomiast ośmiokrotna na osiem części:



Czterokrotna symetria (rysunek dostosowany do układu współrzędnych ekranowych)



Ośmiokrotna symetria (rysunek dostosowany do układu współrzędnych ekranowych)

Dzięki takiemu rozwiązaniu wystarczy obliczyć wartości x oraz y jedynie w przedziałach $\alpha \in \langle 0, \pi/2 \rangle$ dla czterokrotnej i $\alpha \in \langle 0, \pi/4 \rangle$ dla ośmiokrotnej symetrii. Reszta okręgu rysowana jest poprzez zamianę wartości x i y oraz/lub zmianę znaków.

Przykładowo, aby narysować prawą górną oraz prawą dolną ćwiartkę okręgu dla symetrii czterokrotnej należy dla każdego kroku z przedziału $\alpha \in \langle 0, \pi/2 \rangle$:

1. obliczyć x oraz y zgodnie ze wzorem równania okręgu, pomijając we wzorach x_0 oraz y_0
2. narysować punkt w punkcie (x_0+x, y_0+y) – prawa dolna ćwiartka
3. narysować punkt w punkcie (x_0+x, y_0-y) – prawa górna ćwiartka

1.2 Wypełnianie kolorem

Wypełnianie metodą rekurencyjną

Jednym z najprostszych algorytmów wypełniania jest algorytm rekurencyjny. Jego działanie polega na wypełnianiu wszystkich punktów po kolei (zaczynając od dowolnego punktu w figurze), aż do napotkania krawędzi. Ponieważ często jednak nie posiadamy wiedzy na temat koloru krawędzi, lepszym rozwiązaniem jest odczytanie koloru tła piksela początkowego i wypełnianie aż do napotkania innego koloru (taką wersję możemy spotkać w większości programów graficznych).

Po prawej stronie znajduje się rysunek przedstawiający oznaczenia pikseli użyte w poniższym algorytmie:

	N	
W	**P**	E
	S	

P – aktualnie przetwarzany punkt; P=(x, y)
N – punkt znajdujący się bezpośrednio nad P; N=(x, y-1)
S – punkt znajdujący się bezpośrednio pod P; S=(x, y+1)
W – punkt znajdujący się na lewo od P; W=(x-1, y)
E – punkt znajdujący się na prawo od P; E=(x+1, y)

Algorytm można zapisać następująco:

void wypełnij(P, kolorWypełnienia, kolorTła):

1. Pobierz kolor punktu P.
2. Jeśli kolor punktu == kolorWypełnienia, przejdź do 4
3. Jeśli kolor punktu == kolorTła wykonuj:
   - ustaw kolor punktu na kolorWypełnienia
   - wywołaj rekurencyjnie wypełnij(...) dla każdego z punktów: N, S, W, E:
     wypełnij(N, kolorWypełnienia, kolorTła); wypełnij(S, kolorWypełnienia, kolorTła);
     wypełnij(W, kolorWypełnienia, kolorTła); wypełnij(E, kolorWypełnienia, kolorTła);
4. Koniec.

Powyższy algorytm posiada poważną wadę, jaką jest bardzo duże użycie rekurencji, a co za tym idzie bardzo szybkie przepełnienie się sterty. Dla małych obszarów program będzie działał poprawnie, natomiast dla większych dostępna pamięć zostanie szybko zapełniona i aplikacja zakończy działanie z błędem. Rozwiązaniem tego problemu jest utworzenie wersji iteracyjnej algorytmu (najprościej z wykorzystaniem stosu, na który odkładane są piksele, które byłyby użyte w wywołaniach rekurencyjnych).

1.3 Nieblokujące sprawdzanie stanu klawiszy

Do tej pory w celu sprawdzenia naciśnięcia klawisza korzystaliśmy z funkcji readkey(), która blokowała program, aż do naciśnięcia dowolnego klawisza. Allegro dostarcza również nieblokującego mechanizmu poprzez zmienną (tablicę) key, która podobnie jak screen jest dostępna bez konieczności wcześniejszej deklaracji. Przechowuje ona flagi informujące o stanie klawiszy, dzięki czemu możliwe jest sprawdzenie np. czy wciśnięto jednocześnie dwa różne klawisze.

Sprawdzenie stanu klawisza odbywa się poprzez odczyt wartości odpowiedniego elementu tablicy key korzystając ze scancodu, np.

```
if(key[KEY_ESC]) allegro_message("Wciśnięto ESC");
```

Wartości wszystkich scancodów można znaleźć pod adresem:
http://alleg.sourceforge.net/latestdocs/en/alleg006.html#key
Pętla główna może wyglądać teraz następująco:

```
while( true){

  // kod wykonywany w pętli za każdym razem

  if(key[KEY_ESC]) break;
  // sprawdzanie innych klawiszy

}
```

Ponieważ pętla główna wykonuje się bez zatrzymania, dobrym rozwiązaniem jest przeniesienie funkcji rysującej obraz przed pętlę. Następnie, odrysowanie ekranu można wykonywać w przypadku wystąpienia akcji (np. naciśnięcie klawisza).

1.4 Obsługa myszy

Obsługę myszy, podobnie jak obsługę klawiatury należy włączyć za pomocą funkcji install_mouse(). Następnie w celu wyświetlenia kursora systemowego (o ile nie jest widoczny) można użyć funkcji show_os_cursor(1);

Odczyt położenia myszy:

Położenie myszy można odczytać za pomocą dwóch zmiennych (które podobnie jak screen i key są już zdefiniowane przez bibliotekę Allegro): mouse_x oraz mouse_y.

Odczyt stanu klawiszy myszy:

Stan klawiszy myszy jest przechowywany w zmiennej Allegro o nazwie mouse_b. W celu sprawdzenia, czy dany klawisz został naciśnięty należy sprawdzić wartość odpowiedniego bitu:

1 bit – lewy klawisz – sprawdzenie za pomocą: mouse_b & 1

2 bit – prawy klawisz - mouse_b & 2

3 bit – rolka - mouse_b & 4

Przykład:

```
#include <iostream>
#include <allegro.h>

using namespace std;

void getMouseInfo(){

     if(mouse_b & 1){
             allegro_message("lewy klawisz, polozenie: %d, %d", mouse_x,
                             mouse_y);
    }

     if(mouse_b & 2){
             allegro_message("prawy klawisz, polozenie: %d, %d", mouse_x,
                             mouse_y);
       }

      if(mouse_b & 4){
             allegro_message("rolka, polozenie: %d, %d", mouse_x, mouse_y);
       }
}

int main(void)
{
    if(allegro_init()!=0){
      allegro_message("Wystapil blad przy inicjalizacji biblioteki.\n%s",
                      allegro_error);
      exit(1);
    }
    set_gfx_mode(GFX_AUTODETECT_WINDOWED, 640, 480, 0, 0);
    install_keyboard();
```

```
    install_mouse();
    show_os_cursor(1); // włączenie wyświetlania kursora systemowego
    while( true){
      getMouseInfo();
      if(key[KEY_ESC]) break;
    }
    allegro_exit();
    return 0;
}
END_OF_MAIN()
```

Więcej o obsłudze myszy można poczytać na stronie:

http://alleg.sourceforge.net/latestdocs/en/alleg004.html

2 Zadanie

Napisz program, który będzie rysował okręgi metodą czterokrotnej lub ośmiokrotnej symetrii. Liczbę okręgów oraz ich położenie należy losować. Następnie dodaj wypełnianie kolorem punktu wskazanego lewym klawiszem myszy. Wypełnianie zrealizuj za pomocą algorytmu rekurencyjnego lub własnej wersji iteracyjnej (przy testach warto pamiętać, że algorytm rekurencyjny dla dużych powierzchni szybko zapełni stertę i wyłączy program). Kolor wypełniania powinien być losowany za pomocą prawego przycisku myszy.