Rysowanie punktów na powierzchni graficznej

Tworzenie biblioteki rozpoczniemy od podstawowej funkcji graficznej gfxPlot() - rysowania pojedynczego punktu na zadanych współrzędnych i o zadanym kolorze RGB. Przy projektowaniu założymy, iż powierzchnia graficzna pracuje w trybie pikseli 32 bitowych. Dla innych trybów należałoby utworzyć analogiczne funkcje lub funkcję ogólną - stosowny przykład można znaleźć w dokumentacji SDL. Jednakże ogólność funkcji znacznie ją spowalnia, gdyż każdorazowo musi testować tryb graficzny powierzchni - przy dużej ilości punktów ma to znaczenie.

Definicja funkcji jest następująca:

UWAGA: Poniższy kod funkcji gfxPlot() dopisz do końca pliku SDL_gfx.cpp.

```
// gfxPlot() rysuje piksel 32 bitowy
// screen - wskaźnik struktury SDL_Surface
// x,y    - współrzędne piksela
// color  - 32 bitowy kod koloru piksela
//-------------------------------------------------
-----------------

void gfxPlot(SDL_Surface * screen, Sint32 x, Sint32
y, Uint32 color)
{
  Uint32 * p = (Uint32 *)screen->pixels + y *
screen->w + x;
  * p = color;
}
```

Przyjrzyjmy się dokładniej gfxPlot(). Lista parametrów obejmuje:

screen - wskaźnik do zainicjowanej struktury typu SDL_Surface. Utworzona wcześniej powierzchnia musi zawierać piksele 32 bitowe oraz być zablokowana dla innych wątków.
x, y - współrzędne punktu, który ma zostać narysowany na powierzchni wskazywanej przez parametr screen. Współrzędne x,y muszą wskazywać punkt leżący wewnątrz powierzchni. jeśli będzie on na zewnątrz, to powstaną nieprzewidywalne błędy - łącznie z zawieszeniem programu.
color - 32 bitowy kolor punktu.

Wewnątrz funkcji gfxPlot() tworzymy wskaźnik pikseli p. Ponieważ piksel jest daną 32 bitową, wskaźnik p posiada typ Uint32 *. Inicjujemy p adresem piksela. Pod adresem p umieszczamy kod piksela. To wszystko, funkcja jest gotowa do użytku.

UWAGA: Na końcu pliku nagłówkowego SDL_gfx.h dopisz:

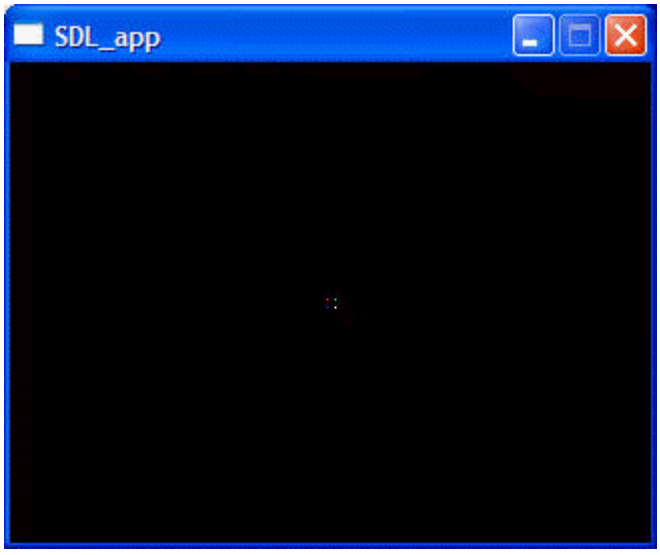
```
void gfxPlot(SDL_Surface * screen, Sint32 x, Sint32
y, Uint32 color);
```

Teraz napiszemy kilka prostych aplikacji wykorzystujących naszą funkcję gfxPlot(). Przejdź w edytorze do pliku **main.cpp** i wpisz poniższy kod:

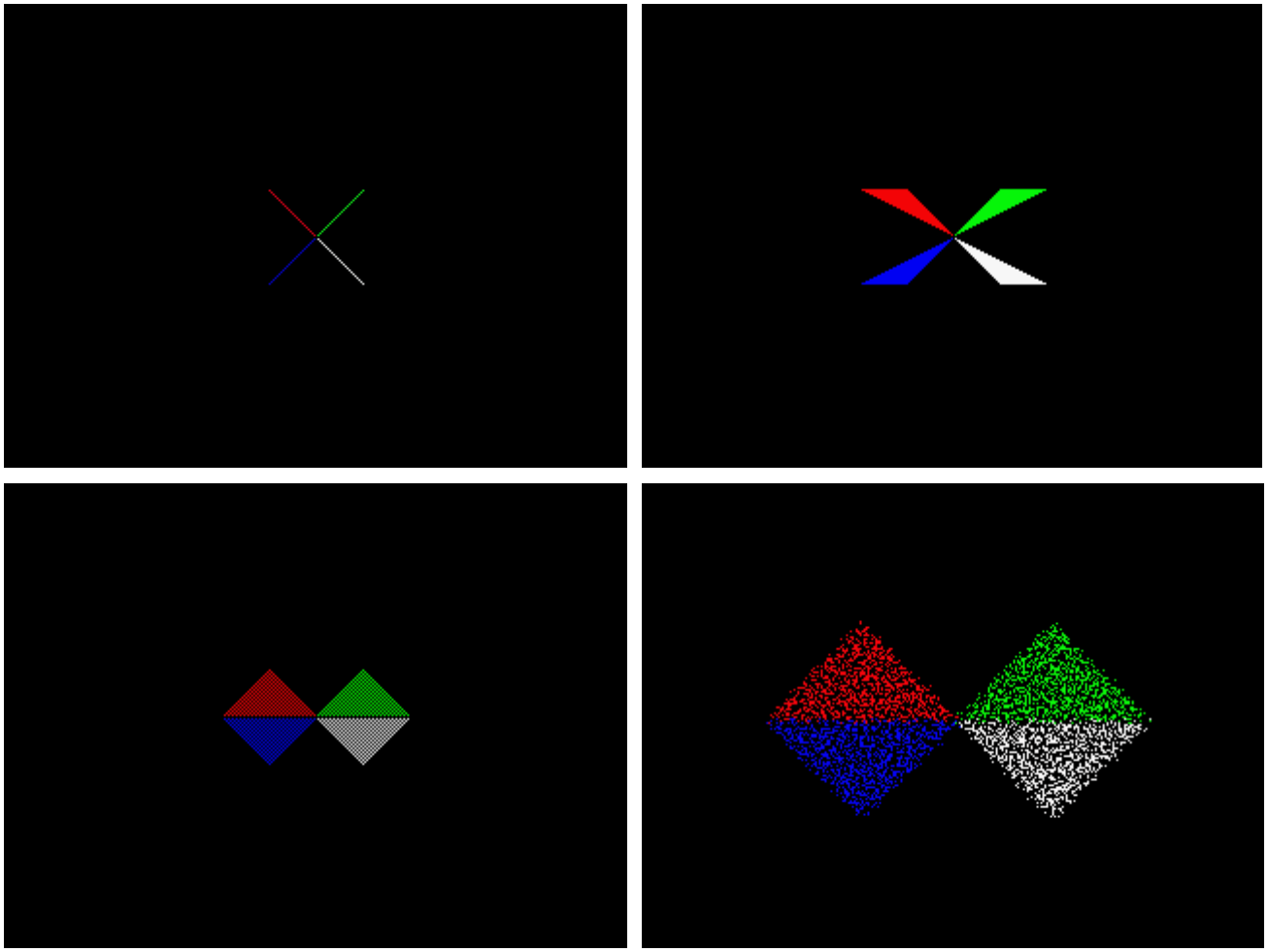
```
//
// P005 - test funkcji gfxPlot()
//------------------------------

#include <windows.h>
#include <SDL/SDL_gfx.h>

const int SCRX = 320; // stałe określające
szerokość i wysokość
const int SCRY = 240; // ekranu w pikselach

int main(int argc, char *argv[])
{

  if(SDL_Init(SDL_INIT_VIDEO))
  {
    MessageBox(0, SDL_GetError(), "Błąd
inicjalizacji SDL", MB_OK);
    exit(-1);
  }

  atexit(SDL_Quit);

  SDL_Surface * screen; // przechowuje wskaźnik
struktury SDL_Surface;

// tworzymy bufor obrazowy SCRXX x SCRY pikseli

  if(!(screen = SDL_SetVideoMode(SCRX, SCRY, 32,
SDL_HWSURFACE)))
  {
    MessageBox(0, SDL_GetError(), "Błąd tworzenia
bufora obrazowego", MB_OK);
    exit(-1);
  }

// testujemy nowo utworzoną funkcję Plot32

  if(SDL_MUSTLOCK(screen))
    if(SDL_LockSurface(screen) < 0)
    {
```

```
      MessageBox(0, SDL_GetError(), "Błąd blokady
bufora obrazowego", MB_OK);
      exit(-1);
    }

// Na środku ekranu rysujemy w narożach kwadratu
cztery punkty - czerwony,
// zielony, niebieski i biały

  int x = screen->w / 2, y = screen->h / 2; // wsp.
środka ekranu

  gfxPlot(screen, x - 2, y - 2, 0xff0000); // punkt
czerwony
  gfxPlot(screen, x + 2, y - 2, 0x00ff00); // punkt
zielony
  gfxPlot(screen, x - 2, y + 2, 0x0000ff); // punkt
niebieski
  gfxPlot(screen, x + 2, y + 2, 0xffffff); // punkt
biały

  if(SDL_MUSTLOCK(screen))
SDL_UnlockSurface(screen);

  SDL_UpdateRect(screen, x - 2, y - 2, 5, 5);

  MessageBox(0, "Kliknij przycisk OK", "Koniec",
MB_OK);
  return 0;
}
```

Efektem działania tego programu jest okienko graficzne z czterema punktami umieszczonymi w narożnikach małego kwadratu.



Poeksperymentuj z powyższym programem. Zastanów się, co należy w nim zmienić, aby otrzymać poniższe efekty graficzne:



Opanowawszy rysowanie pojedynczych punktów zajmiemy się rysowaniem linii. Na początek przypadki proste - rysowanie linii poziomych gfxHLine() i pionowych gfxVLine().

UWAGA: Poniższy kod funkcji gfxHLine() gfxVLine dopisz do końca pliku SDL_gfx.cpp.

```
// gfxHLine() rysuje linię poziomą
// screen - wskaźnik struktury SDL_Surface
// x,y    - współrzędne lewego początku lini
// color  - 32 bitowy kod koloru linii
// len    - długość linii w pikselach
//------------------------------------------------
-----------------

void gfxHLine(SDL_Surface * screen, Sint32 x,
Sint32 y, Uint32 color, Uint32 len)
{
```

```
  Uint32 * p = (Uint32 *)screen->pixels + y *
screen->w + x;
  for(int i = 0; i < len; i++) * p++ = color;
}

// gfxVLine() rysuje linię pionową
// screen - wskaźnik struktury SDL_Surface
// x,y    - współrzędne górnego początku lini
// color  - 32 bitowy kod koloru linii
// len    - długość linii w pikselach
//-----------------------------------------------
-----------------

void gfxVLine(SDL_Surface * screen, Sint32 x,
Sint32 y, Uint32 color, Uint32 len)
{
  Uint32 ylen = screen->w;
  Uint32 * p  = (Uint32 *)screen->pixels + y * ylen
+ x;
  for(int i = 0; i < len; i++)
  {
    * p = color; p += ylen;
  }
}
```

UWAGA: Na końcu pliku nagłówkowego SDL_gfx.h dopisz:

```
void gfxHLine(SDL_Surface * screen, Sint32 x,
Sint32 y, Uint32 color, Uint32 len);

void gfxVLine(SDL_Surface * screen, Sint32 x,
Sint32 y, Uint32 color, Uint32 len);
```

Obie funkcje przyjmują następujące parametry:

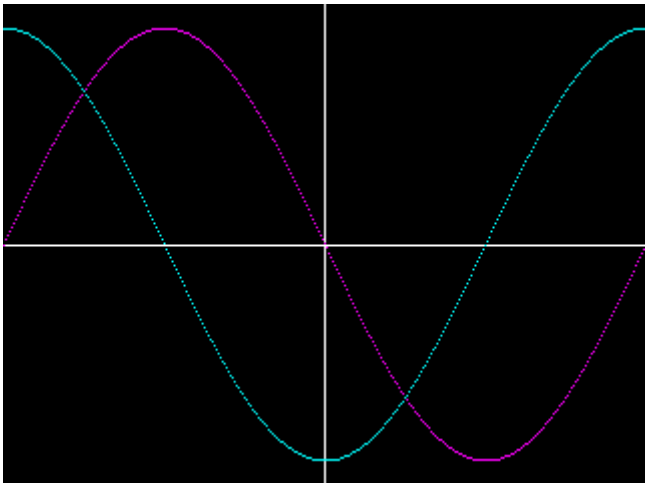
n - wskaźnik do zainicjowanej struktury typu SDL_Surface. Utworzona wcześniej powierzchnia musi zawierać piksele 32 bitowe oraz być zablokowana dla innych wątków.
x, y - współrzędne początku linii - dla gfxHLine jest to lewy wierzchołek, a dla gfxVLine jest to górny wierzchołek.
color - 32 bitowy kolor punktu.
len - długość linii w pikselach. gfxHLine rysuje linię od punktu x,y w prawo, gfxVLine rysuje ją od punktu x,y w dół. Rysowana linia powinna w całości mieścić się na zarezerwowanej powierzchni graficznej, inaczej powstaną nieprzewidywalne efekty.

Działanie obu funkcji jest bardzo podobne. Najpierw w zmiennej p obliczamy adres pierwszego punktu linii. Następnie rysujemy len punktów, każdorazowo zwiększając adres dla gfxHLine() o szerokość piksela, a dla gfxVLine() o

szerokość linii. W efekcie powstaje linia pozioma lub pionowa. Poniższy program rysuje wykresy funkcji **sinus** i **cosinus** wykorzystując zaprojektowane funkcje biblioteczne - zwróć uwagę na jego względną prostotę w stosunku do osiąganego wyniku!

```
//
// P006 - wykresy funkcji sinus i cosinus
//---------------------------------------

#include <windows.h>
#include <SDL/SDL_gfx.h>
#include <math.h>      // dostęp do funkcji
trygonometrycznych

const int SCRX = 320; // stałe określające
szerokość i wysokość
const int SCRY = 240; // ekranu w pikselach

int main(int argc, char *argv[])
{

  if(SDL_Init(SDL_INIT_VIDEO))
  {
    MessageBox(0, SDL_GetError(), "Błąd
inicjalizacji SDL", MB_OK);
    exit(-1);
  }

  atexit(SDL_Quit);

  SDL_Surface * screen;

  if(!(screen = SDL_SetVideoMode(SCRX, SCRY, 32,
SDL_HWSURFACE)))
  {
    MessageBox(0, SDL_GetError(), "Błąd tworzenia
bufora obrazowego", MB_OK);
    exit(-1);
  }

  if(SDL_MUSTLOCK(screen))
    if(SDL_LockSurface(screen) < 0)
    {
      MessageBox(0, SDL_GetError(), "Błąd blokady
bufora obrazowego", MB_OK);
```

```
      exit(-1);
    }

  int x = screen->w / 2, y = screen->h / 2;     //
współrzędne środka ekranu

  double a = y * 0.9;                           //
współczynnik skalowania wykresu

  gfxHLine(screen, 0, y, 0xffffff, screen->w); //
oś x
  gfxVLine(screen, x, 0, 0xffffff, screen->h); //
oś y

  for(x = 0; x < screen->w; x++)                //
wykres sinus i cosinus
  {
    gfxPlot(screen, x, y - a * sin(6.28 * x /
screen->w), 0xff00ff);
    gfxPlot(screen, x, y - a * cos(6.28 * x /
screen->w), 0x00ffff);
  }

  if(SDL_MUSTLOCK(screen))
SDL_UnlockSurface(screen);

  SDL_UpdateRect(screen, 0, 0, 0, 0);

  MessageBox(0, "Kliknij przycisk OK", "Koniec",
MB_OK);
  return 0;
}
```



Wykorzystaj program do narysowania innych wykresów funkcji. Zastanów się jak wyskalować wykres, aby nie wychodził poza obszar okna - możesz również sprawdzać, czy współrzędna y punktu mieści się w zakresie od 0 do screen->h - 1. Jeśli tak, punkt rysujesz, jeśli nie, nie rysujesz punktu. Proste?

Kolejna funkcja biblioteczna gfxRectangle() będzie rysowała prostokątne ramki. Funkcja wykorzystuje zdefiniowane wcześniej funkcje gfxHLine() i gfxVLine() do rysowania boków poziomych i pionowych.

UWAGA: Poniższy kod funkcji gfxRectangle dopisz do końca pliku SDL_gfx.cpp.

```
// gfxRectangle() rysuje ramkę na obrysie
prostokąta
// screen - wskaźnik struktury SDL_Surface
// r      - definiuje prostokąt ramki
// color  - 32 bitowy kod koloru ramki
//-------------------------------------------------
-----------------

void gfxRectangle(SDL_Surface * screen, SDL_Rect *
r, Uint32 color)
{
  if(r)
  {
    gfxHLine(screen, r->x, r->y, color, r->w);
    gfxHLine(screen, r->x, r->y + r->h - 1, color,
r->w);
    gfxVLine(screen, r->x, r->y, color, r->h);
    gfxVLine(screen, r->x + r->w - 1, r->y, color,
r->h);
  }
  else
  {
    gfxHLine(screen, 0, 0, color, screen->w);
    gfxHLine(screen, 0, screen->h - 1, color,
screen->w);
    gfxVLine(screen, 0, 0, color, screen->h);
    gfxVLine(screen, screen->w - 1, 0, color,
screen->h);
  }
}
```

Jeśli dokładnie przeanalizujesz kod funkcji gfxRectangle(), to zauważysz, iż boki pionowe są rysowane pomiędzy punktami narożnymi prostokąta, w

których postawiliśmy już piksele przy okazji rysowania boków poziomych. Może wpadniesz na pomysł zmodyfikowania współrzędnych i długości odcinków pionowych - nie rób tego. Czas tracony przez komputer na wyliczenia współrzędnych i długości może być większy od czasu postawienia tych czterech pikseli (a może nie będzie - sprawdź oba warianty, jeśli potrafisz).

Funkcja gfxRectangle() posiada następujące parametry:

screen - wskaźnik do zainicjowanej struktury typu SDL_Surface. Utworzona wcześniej powierzchnia musi zawierać piksele 32 bitowe oraz być zablokowana dla innych wątków.
r - wskaźnik struktury typu SDL_Rect, która definiuje prostokąt. Jeśli r zawiera NULL, prostokąt jest rysowany na całej powierzchni graficznej.
color - 32 bitowy kolor punktu.

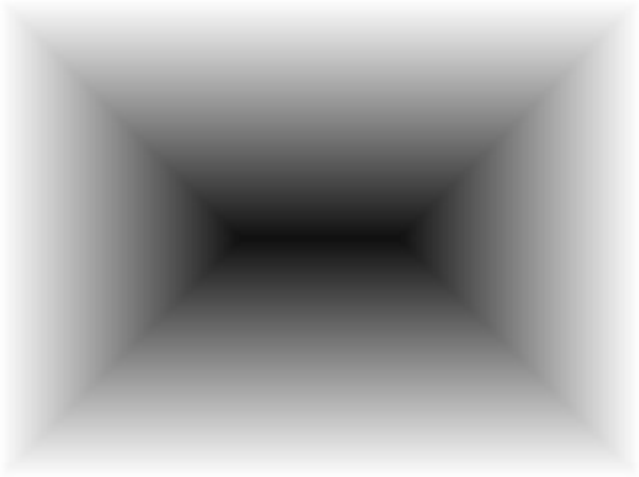
UWAGA: Na końcu pliku nagłówkowego SDL_gfx.h dopisz:

```
void gfxRectangle(SDL_Surface * screen, SDL_Rect *
r, Uint32 color);
```

Poniższy program testuje funkcję gfxRectangle() rysując na środku powierzchni graficznej coraz mniejsze ramki o malejącej jasności. Osiągamy dosyć ciekawy efekt graficzny. Zwróć uwagę jak modyfikowany jest kolor kolejnej ramki - zmniejszeniu ulegają wszystkie składowe R, G i B.

```
//
// P007 - test ramek
//------------------

#include <windows.h>
#include <SDL/SDL_gfx.h>

const int SCRX = 320; // stałe określające
szerokość i wysokość
const int SCRY = 240; // ekranu w pikselach

int main(int argc, char *argv[])
{

  if(SDL_Init(SDL_INIT_VIDEO))
  {
    MessageBox(0, SDL_GetError(), "Błąd
inicjalizacji SDL", MB_OK);
    exit(-1);
  }

  atexit(SDL_Quit);
```

```
  SDL_Surface * screen;

  if(!(screen = SDL_SetVideoMode(SCRX, SCRY, 32,
SDL_HWSURFACE)))
  {
    MessageBox(0, SDL_GetError(), "Błąd tworzenia
bufora obrazowego", MB_OK);
    exit(-1);
  }

  if(SDL_MUSTLOCK(screen))
    if(SDL_LockSurface(screen) < 0)
    {
      MessageBox(0, SDL_GetError(), "Błąd blokady
bufora obrazowego", MB_OK);
      exit(-1);
    }

  Uint32 color = 0xffffff;
  SDL_Rect * r = new SDL_Rect;        // wskaźnik
prostokąta

  for(int i = 0; i < 120; i++)
  {
    r->x = r->y = i;                  // tworzymy
definicję prostokąta ramki
    r->w = screen->w - 2 * i;
    r->h = screen->h - 2 * i;

    gfxRectangle(screen, r, color);   // rysujemy
ramkę

    color -= 0x020202;                // obliczamy
kod koloru dla kolejnej ramki
  }

  if(SDL_MUSTLOCK(screen))
SDL_UnlockSurface(screen);

  SDL_UpdateRect(screen, 0, 0, 0, 0);

  MessageBox(0, "Kliknij przycisk OK", "Koniec",
MB_OK);
  return 0;
}
```



Do wypełniania prostokątnych obszarów na powierzchni graficznej w bibliotece SDL istnieje funkcja SDL_FillRect() o następującej definicji:

```
int SDL_FillRect(SDL_Surface * screen, SDL_Rect *
r, Uint32 color);
```

screen - wskazuje strukturę SDL_Surface z powierzchnią graficzną, po której chcemy rysować. Jeśli na powierzchni jest zdefiniowany prostokąt obcinania, to prostokąt wypełniający pojawi się tylko wewnątrz prostokąta obcinania, w obszarze wspólnym. Powierzchnia graficzna może być dowolnego typu, niekoniecznie 32-bitowa.

r - wskazuje strukturę SDL_Rect definiującą prostokątny obszar. Jeśli wskaźnik ma wartość NULL, to wypełniona zostanie cała powierzchnia graficzna (patrz wyżej - prostokąt obcinania).

color - 32 bitowy kolor punktu. Kod piksela musi być kompatybilny z formatem graficznym powierzchni. Dla dowolnej powierzchni można go uzyskać przy pomocy wywołania funkcji biblioteki SDL:

```
Uint32 SDL_MapRGB(SDL_PixelFormat *fmt, Uint8 r, Uint8 g, Uint8 b);
```

fmt - wskazuje strukturę SDL_PixelFormat definiującą format pikseli. Można tutaj podać zawartość pola format struktury SDL_Surface, po której chcemy rysować.

r,g,b - składowe kolorów bazowych: r - czerwony, g - zielony i b - niebieski

Wykorzystaj tę funkcję wraz z gfxRectangle() do utworzenia poniższych rysunków (zastanów się nad efektywnym wykorzystaniem struktur SDL_Rect):