ЛЕКЦІЯ 9

Архітектура графічної системи Windows. Загальний огляд систем GDI та GDI+.

Операційні системи

доц. Сінельнікова Т.Ф.

БАЗОВІ ПОНЯТТЯ

Кадровий буфер і формат пікселів. Всі сучасні відеоадаптери працюють на растровому принципі, це означає, що інформація в них зберігається у вигляді двовимірних масивів пікселів в області пам'яті відеоадаптера. Така область пам'яті називається **кадровим буфером (frame buffer).** Кадрові буфери мають різні розміри.

БАЗОВІ ПОНЯТТЯ

Розмір мінімального кадрового буфера, підтримуваного в ОС Windows, становить 640 пікселів в рядку на 480 рядків, тобто режим VGA 640x480 пікселів. Максимальні розміри кадрового буфера можуть досягати 1600 x 1200 і навіть 1920 x 1200 пікселів. Для більшості дозволів ширина і висота екрану знаходяться в пропорції 4:3 - наприклад, 640 x 480, 800 x 600, 1024 x 768, 1600 x 1200, а ось для дозволів 1280 x1024, 1920x 1080 і 1920x1200 існують вже інші пропорції - 5:4, 16:9 і 8:5 відповідно.

ФОРМАТ ПІКСЕЛІВ

15-ти розрядний формат пікселів

16-ти розрядний формат пікселів

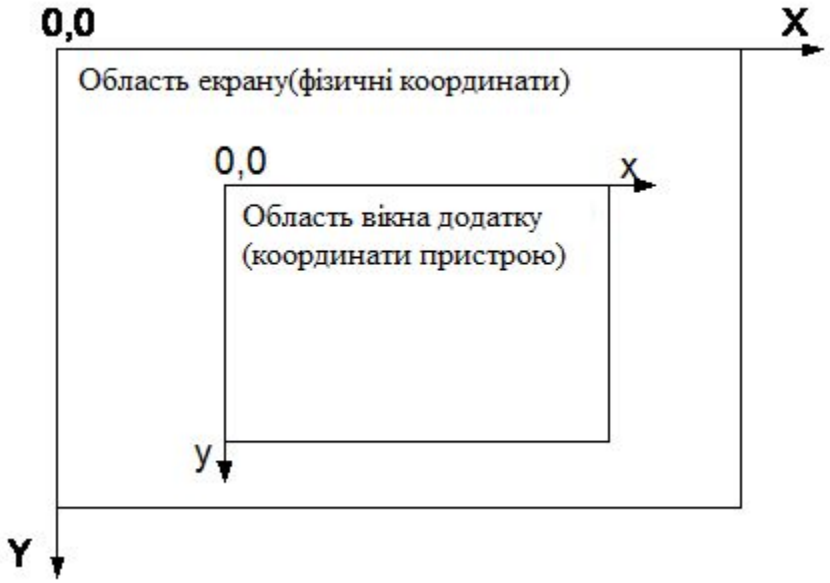
СИСТЕМИ КООРДИНАТ

Фізична система координат - складається з пікселів графічної поверхні фізичного пристрою. Розмір фізичної системи координат складає 227x227 одиниць по обох осях X і Y. Фізична система координат використовується драйвером графічного пристрою і являє собою матрицю пікселів фіксованої висоти і ширини. Початок відліку - точка (0,0) розташована в лівому верхньому кутку. Вісь x спрямована зліва направо, а вісь y - зверху вниз.

СИСТЕМИ КООРДИНАТ

Система координат пристрою - описує пікселі контексту пристрою. Вона підтримує відображення на прямокутні галузі фізичної системи координат. Розмір цієї системи координат складає 227x227 одиниць. Дана система орієнтована так само, як і фізична система координат, однак, початок координат розміщено у верхньому лівому кутку пристрою, створеного функціями CreateDC, CreateIC і CreateCompatibleDC. Для контекстів пристроїв, пов'язаних з вікнами програм, початок відліку вміщено у верхній лівий кут вікна. На малюнку 10.2 наведено співвідношення фізичної системи координат та системи координат пристрою.

СИСТЕМИ КООРДИНАТ

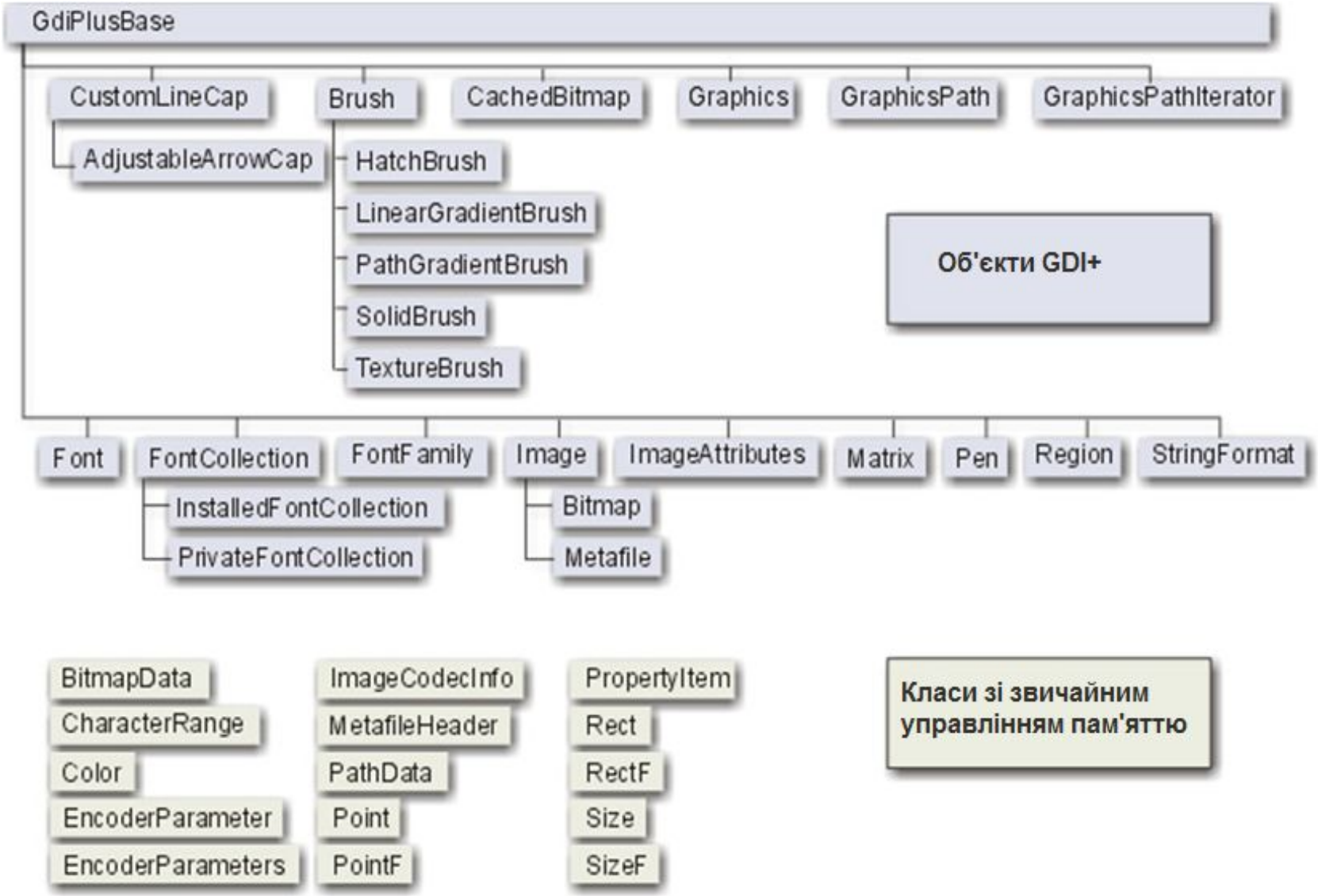


СИСТЕМИ КООРДИНАТ

Сторінкова система координат забезпечує деякий набір перетворень в систему координат пристрою. Розміри цієї системи координат 232x232. Дана система координат дає можливість додатку будувати геометричну модель з довільно обраним напрямом осей і фізичним масштабом. Дана система координат практично не залежить від конкретних пристроїв виводу і є єдиною логічною системою координат, що підтримується 16-ти розрядними ОС сімейства Windows, 32 розрядними версіями Windows, а також Windows CE.

Світова система координат описує двовимірний простір розміром 232 x 232. При відображенні точок даної системи координат в сторінкову систему координат з'явилася можливість здійснювати різні перетворення, в тому числі і аффінні. Дана система координат підтримується тільки в ОС сімейства Windows NT/2000.

ІЄРАРХІЯ КЛАСІВ GDI+



ПРИКЛАД ДОДАТКУ З GDI+

```
#define UNICODE
#include <windows.h>
#include <gdiplus.h>
using namespace Gdiplus;
 VOID OnPaint(HDC hdc){
   Graphics graphics(hdc);
// Все строки - в кодировке Unicode
  WCHAR welcome[]=L"Welcome, GDI+ !";
// Создаем контекст рисования и устанавливаем
// пиксельную систему координат
graphics.SetPageUnit(UnitPixel);
RectF bounds(0, 0, float(rc.right), float(rc.bottom));
// Загружаем фоновое изображение и растягиваем его на все окно

Image bg(L"BACKGRND.gif");
graphics.DrawImage(&bg, bounds);

// Создаем кисть с градиентом на все окно и полупрозрачностью
LinearGradientBrush brush(bounds, Color(130, 255, 0, 0), Color(255,0,0,255),
   LinearGradientModeBackwardDiagonal);

// Готовим формат и параметры шрифта StringFormat format;
   format.SetAlignment(StringAlignmentCenter);
   format.SetLineAlignment(StringAlignmentCenter); Font font(L"Arial", 48,
   FontStyleBold); // Выводим текст приветствия, длина -1 означает, // что строка
   заканчивается нулем graphics.DrawString(welcome, -1, &font, bounds, &format,
   &brush);
// Рисуем линию
   Pen      pen(Color(255, 0, 0, 255));
   graphics.DrawLine(&pen, 0, 0, 200, 100);
}
```

ПРИКЛАД ДОДАТКУ З GDI+

```
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT WINAPI WinMain(HINSTANCE hInstance, HINSTANCE, PSTR, INT iCmdShow)
{
   HWND                hWnd;
   MSG                 msg;
   WNDCLASS            wndClass;
   GdiplusStartupInput gdiplusStartupInput;
   ULONG_PTR           gdiplusToken;

   // Initialize GDI+.
   GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

   wndClass.style          = CS_HREDRAW | CS_VREDRAW;
   wndClass.lpfnWndProc    = WndProc;
   wndClass.cbClsExtra     = 0;
   wndClass.cbWndExtra     = 0;
   wndClass.hInstance      = hInstance;
   wndClass.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
   wndClass.hCursor        = LoadCursor(NULL, IDC_ARROW);
   wndClass.hbrBackground  = (HBRUSH)GetStockObject(WHITE_BRUSH);
   wndClass.lpszMenuName   = NULL;
   wndClass.lpszClassName  = TEXT("GettingStarted");

   RegisterClass(&wndClass);
```

ПРИКЛАД ДОДАТКУ З GDI+

```
hWnd = CreateWindow(
       TEXT("GettingStarted"),    // window class name
       TEXT("Getting Started"),   // window caption
       WS_OVERLAPPEDWINDOW,       // window style
       CW_USEDEFAULT,             // initial x position
       CW_USEDEFAULT,             // initial y position
       CW_USEDEFAULT,             // initial x size
       CW_USEDEFAULT,             // initial y size
       NULL,                      // parent window handle
       NULL,                      // window menu handle
       hInstance,                 // program instance handle
       NULL);                     // creation parameters

   ShowWindow(hWnd, iCmdShow);
   UpdateWindow(hWnd);

   while(GetMessage(&msg, NULL, 0, 0))
   {
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }

   GdiplusShutdown(gdiplusToken);
   return msg.wParam;
}  // WinMain
```

ПРИКЛАД ДОДАТКУ З GDI+

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
   WPARAM wParam, LPARAM lParam)
{
   HDC          hdc;
   PAINTSTRUCT  ps;

   switch(message)
   {
   case WM_PAINT:
      hdc = BeginPaint(hWnd, &ps);
      OnPaint(hdc);
      EndPaint(hWnd, &ps);
      return 0;
   case WM_DESTROY:
      PostQuitMessage(0);
      return 0;
   default:
      return DefWindowProc(hWnd, message, wParam, lParam);
   }
} //WndProc
```