БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий

и робототехники

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

О Т Ч Ё Т

о лабораторной работе № 6

Дисциплина

«Системное программирование»

Тема

«Графика Windows. Основы управления выводом графической и текстовой информации на базе библиотеки GDI. Растровая графика»

Выполнил: студент гр. 10702217 Храмков Д. С.

Проверил: Разорёнов Н. А.

Минск 2019

***Лабораторная работа № 6***

**ГРАФИКА WINDOWS. ОСНОВЫ УПРАВЛЕНИЯ ВЫВОДОМ ГРАФИЧЕСКОЙ И ТЕКСТОВОЙ ИНФОРМАЦИИ НА БАЗЕ БИБЛИОТЕКИ GDI. РАСТРОВАЯ ГРАФИКА.**

**Цель работы:** изучить основы управления выводом текстовой и графической информации на базе библиотеки GDI. Изучить структуру и основные операции по обработке растровых изображений в Windows.

**Изучаемые вопросы**

1. Графическое устройство и его контекст.
2. Атрибуты системы координат, их влияние на вывод информации.
3. Шрифты, классификация, параметры шрифта, установка в конеткст устройства.
4. Атрибуты контекства устройства, влияющие на вывод текства.
5. Методы GDI для вывода текста и векторной графики.
6. Структура файлов \*.bmp:
   1. Заголовки;
   2. Палитра цветов;
   3. Битовый массив образа.
7. Структура BITMAP:
   1. Создание битовой карты;
   2. Заполнение битовой карты;
   3. Вывод битовой карты.
8. Растровые операции.

***Постановка задачи***

1. Нарисовать геометрическую фигуру в заданной области.
2. Осуществить вывод текста в заданной области (по контуру фигуры) согласно индивидуальному заданию из приложения (выдаётся преподавателем).
3. Создать в графическом редакторе растровые изображения размером 12x12 (16 бит), 16х16 (4 бита), 32х32 (8 бит). Рисунки и дамп включить в отчёт
4. Расшифровку дампов файлов рисунков привести в виде таблицы.
5. Написать программу, где, в соответствии с вариантом из приложения, фигуры из каждого сектора экрана через t после запуска программы начинают двигаться вверх. Схема окна приложения приведена на рисунке 6.1. Движением каждой фигуры управляет отдельный поток.
6. **Графическое устройство и его контекст.**

Контекст устройства – внутренняя структура данных, которая определяет набор графических объектов и ассоциированных с ними атрибутов, а также графических режимов, влияющих на вывод. Поведение контекста определяется его атрибутами. Для создания и освобождения контекста применяются пары функций BeginPaint и EndPaint или GetDC и ReleaseDC.

HDC hdc = BeginPaint(hWnd, &ps);

ReleaseDC(hWnd, hdc);

1. **Атрибуты системы координат, их влияние на вывод информации.**

Основной единицей измерения для системы координат служит пиксел. Точки задаются парой координат (x, y). Для изменения режима отображения в функцию SetMapMode вторым параметром могут передаваться такие значения, как MM\_HIMETRIC, MM\_TEXT (default), MM\_LOMETRIC и т.д.

SetMapMode(hdc, MM\_TEXT);

1. **Шрифты, классификация, параметры шрифта, установка в контекст устройства.**

В GDI шрифты бывают растровыми, векторными и типа TrueType. Создаётся же шрифт функцией CreateFontInderect.

LOGFONT font;

font.lfHeight = 70;

font.lfWidth = 125;

font.lfEscapement = 0;

font.lfOrientation = 0;

font.lfWeight = 50;

font.lfItalic = FALSE;

font.lfUnderline = FALSE;

font.lfStrikeOut = FALSE;

font.lfCharSet = DEFAULT\_CHARSET;

font.lfPitchAndFamily = DEFAULT\_PITCH;

font.lfClipPrecision = CLIP\_LH\_ANGLES;

HFONT hFont = CreateFontIndirect(&font);

HFONT hOldFont = (HFONT)SelectObject(hdc, hFont);

1. **Атрибуты контекста устройства, влияющие на вывод текста.**

Как уже указывалось ранее, на вывод текста может влиять такой параметр, как режим отображения. Кроме него, существуют такие параметры как цвет текста и цвет фона, изменение которых прида ёт новый вид программе. Функции SetTextColor и SetBkColor изменяют цвет текста и текстового фона соответственно.

SetTextColor(hdc, RGB(255, 255, 0));

SetBkColor(hdc, RGB(0, 0, 139));

1. **Методы GDI для вывода тектса и векторной графики.**

Функции GDI позволяют нам настраивать свои перья, кисти, шрифты, палитры и растровые изображения. CreatePen и CreateSolidBrush создают перо и кисть, установив которые в контекст функцией SelectObject, можно использовать для рисования объектов.

HBRUSH hBrush = CreateSolidBrush(RGB(255, 255, 0));

HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

1. **Структура файлов \*.bmp:**
   1. **Заголовки.**

Заголовок файла \*.bmp содержит в себе структуру, хранящую: информацию о байтах “BM” для битовых образцов; общий размер файла; зарезервированные поля; смещение битов битового образа от начала файла. За этой информацией следует другой заголовок, определённый структурой BITMAPINFOHEADER, содержащий общую информацию о файле (ширина, высота, число цветов и т.д.).

BITMAPINFOHEADER bmpInfoHeader = { 0 };

ReadFile(BmpFile, &bmpInfoHeader, sizeof(BITMAPINFOHEADER), NULL, NULL);

* 1. **Палитра цветов.**

Палитра цветов определяется таблицей цветов, состоящей из двух или более структур RGBQUAD, содержащей значение красного, зелёного и синего цветов. Число этих структур определяется значенем поля bBitCount

tagRGBQUAD rgbQuad = { 0 };

ReadFile(BmpFile, &rgbQuad, sizeof(tagRGBQUAD), NULL, NULL);

stream << "\r\nRGBQUAD :\r\n"

<< " Blue : " << (int)rgbQuad.rgbBlue << "\r\n"

<< " Green : " << (int)rgbQuad.rgbGreen << "\r\n"

<< " Red : " << (int)rgbQuad.rgbRed << "\r\n"

<< " Reserved : " << (int)rgbQuad.rgbReserved << "\r\n";

* 1. **Битовый массив образа.**

Битовый массив начинается с нижней строки пикселей, каждая строка начинается с самого левого пикселя, каждый пиксель представлен 1, 4, 8 или 256 битами. Для монохромных битовых образов с 1 битом цвета на пиксель первый пиксель в каждой строке представляется наиболее значащим битом первого байта в каждой строке, в случае 16-цветного представления с 4 битами на пиксель – четырьмя самыми значащими битами первого байта в каждой строке. В случае 256-цветного битового образа каждый байт соответствует одному пикселю.

stream << " BitCount : " << bmpInfoHeader.biBitCount << "\r\n"

1. **Структура BITMAP:** 
   1. **Создание битовой карты.**

Битовая карта создаётся функцией CreateBitmap. Битовый образ виртуального контекста создаётся при помощи CreateCompatibleDC.

HDC memdc = CreateCompatibleDC(hdc);

* 1. **Заполнение битовой карты.**

Заплоняется битовая карта с использованием битового образа виртуального контекста, который передаётся в качестве параметра в различные отрисовывающие функции. Для заполнения битовой карты можно использовать побитовое отображение цветов изображения используется функция SetBitmapBits.

GetObject(hBmp2, sizeof(bm), &bm);

* 1. **Вывод битовой карты.**

Битовая карта выводится функцией BitBlt, которая выполняет поблоковывй вывод битов из источника по указанным координатам.

BitBlt(hdc, 30, 0, bm.bmWidth, bm.bmHeight, memdc, 0, 0, SRCCOPY);

1. **Растровые операции.**

Смысл использования растровых операций заключается в определении комбинации цветов, необходимых для получения желаемого цвета. Наиболее широко применяемыми операциями являются DSTINVERT, MERGECOPY, PATCOPY, SRCCOPY и SRCNIVERT.

GetObject(hBmp3, sizeof(bm), &bm);

BitBlt(hdc, 770, 0, bm.bmWidth, bm.bmHeight, hMemDc, 0, 0, SRCINVERT);

**Алгоритм выполнения программы:**

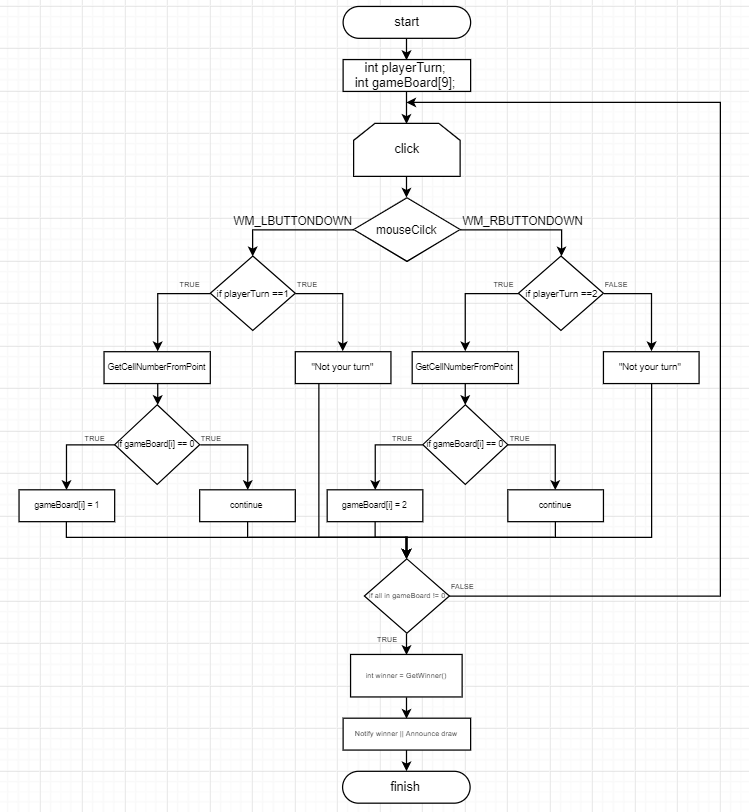


Рисунок 1 – блок-схема игры «Крестики-Нолики»

**Результаты выполнения:**

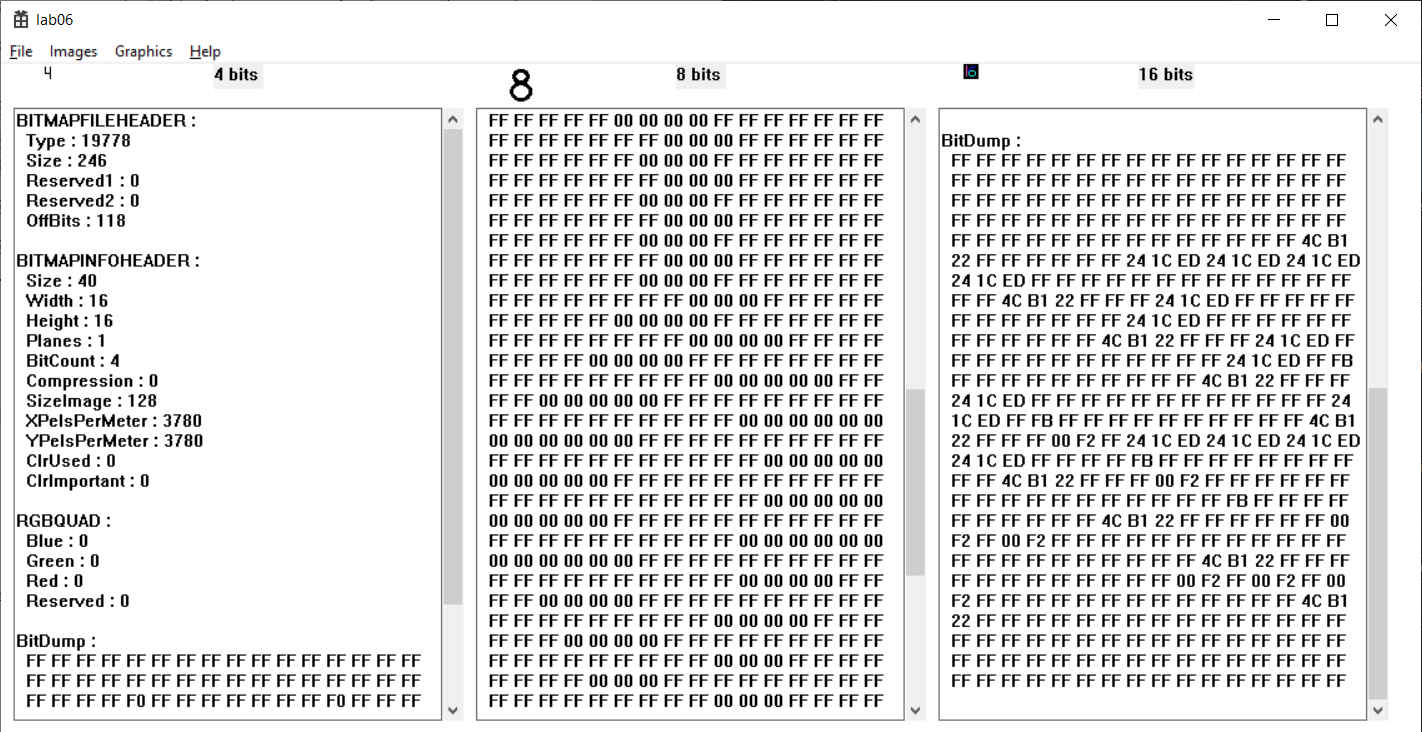


Рисунок 2 – вывод дампа изображений

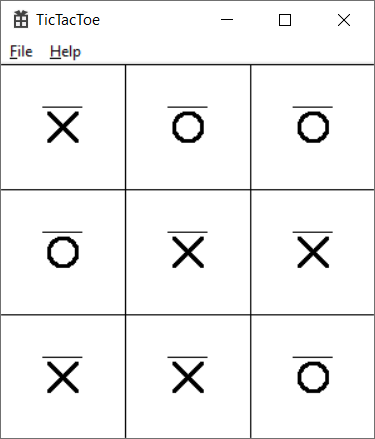


Рисунок 3 – ркзультат игры «Крестики-Нолики»

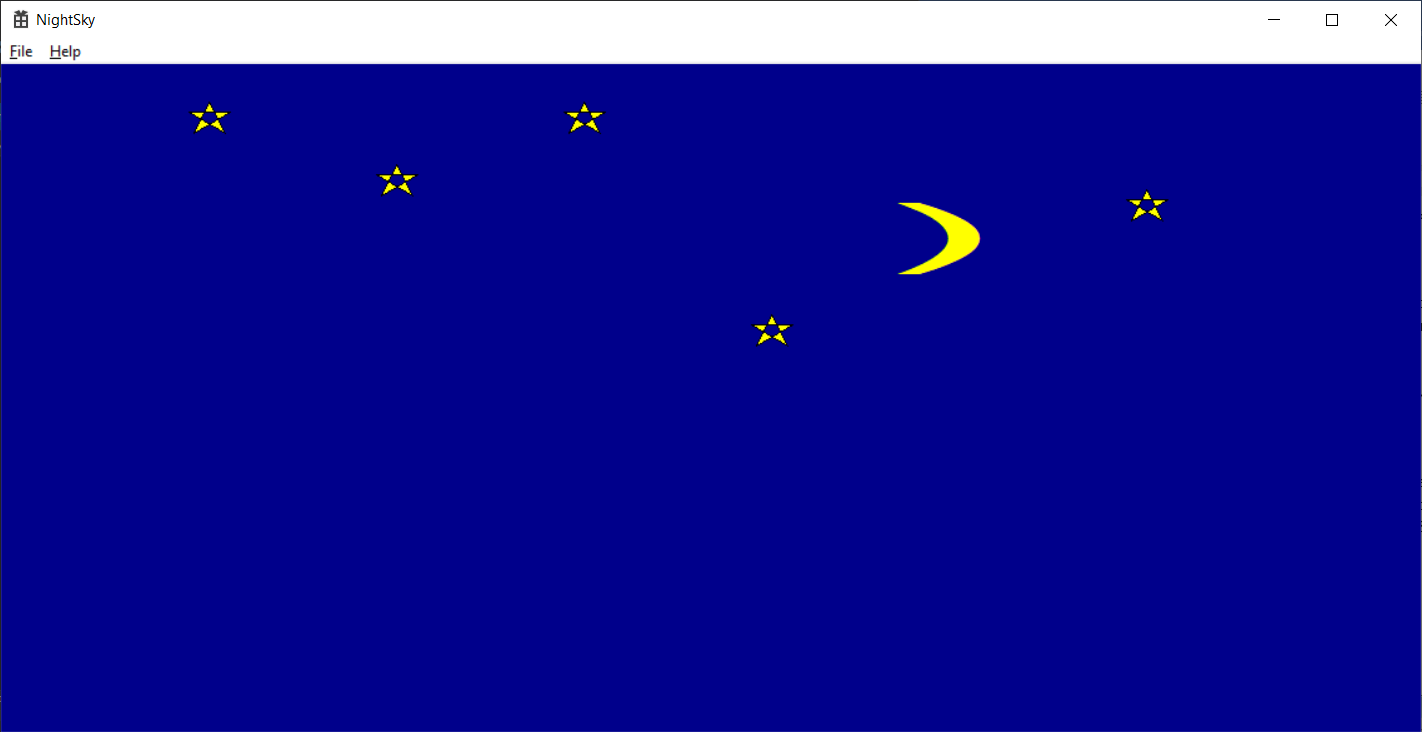


Рисунок 4 – результат программы «Ночное Небо»

**Вывод:**

* 1. Изучены принципы работы с контекстным меню графического устройства;
  2. Ознакомились со структурой LOGFONT;
  3. Научились рисовать фигуры на экране средствами Win32 API;
  4. Изучили структуру BITMAPINFOHEADER.