Анимация

Лекция 5

Программа «Прыгающий мячик»

- В этой программе демонстрируется применение следующих средств Win32 API:
- Использование узорной кисти (CreatePatternBrush) для фона окна (SetClassLong) и для операции стирания (FillRect) предыдущего изображения мяча.
- Контекст устройства в памяти (CreateCompatibleDC) для размещения в нем DDB- растра с изображением мяча (SelectObject) и последующего его вывода в контекст дисплея (BitBlt).
- Использование региона отсечения (CreateELlipticRgn, SelectClipRgn) для выделения в прямоугольном растре области с изображением мяча, которая затем копируется при помощи функции BitBlt.
- Мировые преобразования (SetWorldTransform) для перемещения и вращения изображения мяча.
- Функции SaveDC и RestoreDC, применяемые для сохранения и восстановления текущего состояния контекста устройства.

```
case WM_CREATE:
    hDC = GetDC(hWnd);
    GetClientRect(hWnd, &rect);
    dX = rect.right / 100.;
    dY = rect.bottom / 50.;
        // Создать таймер (0.1 сек)
    SetTimer(hWnd, 1, 100, NULL);

    hBmpBkgr = LoadBitmap((HINSTANCE)GetWindowLong(hWnd,
    GWL_HINSTANCE),MAKEINTRESOURCE(IDB_STONE));

    hBkBrush = CreatePatternBrush(hBmpBkgr);
    SetClassLong(hWnd, GCL_HBRBACKGROUND, (LONG)hBkBrush);

    hBmpBall = LoadBitmap((HINSTANCE)GetWindowLong(hWnd,
    GWL_HINSTANCE),MAKEINTRESOURCE(IDB_BALL));

    GetObject(hBmpBall, sizeof(bm), (LPSTR)&bm);
    SetGraphicsMode(hDC, GM_ADVANCED);
    break;
```

```
case WM_TIMER:
    GetClientRect(hWnd, &rect);
    // Стираем прежнюю картинку мяча
    SetRect(&rBall, (int)x, (int)y, (int)x + bm.bmWidth, (int)y + bm.bmHeight);
    FillRect(hDC, &rBall, hBkBrush);

    // Новая позиция мяча
    x += dX;
    y += dY;
    alpha += 10;
    if (alpha > 360)  alpha = 0;

    // Если мяч достиг края окна, направление его движения изменяется
    if(x + bm.bmWidth > rect.right || x < 0)
    dX = -dX;
    if(y + bm.bmHeight > rect.bottom || y < 0)
    dY = -dY;
    DrawBall(hWnd, hDC, hBmpBall, bm, x, y, alpha);
    break;
```

```
case WM_DESTROY:
    KillTimer(hWnd, 1);
    ReleaseDC(hWnd, hDC);
    PostQuitMessage(0);
    break;
```

```
void DrawBall(HWND hwnd, HDC hdc, HBITMAP hBmp,
    BITMAP bm, FLOAT x, FLOAT y, int alpha) {
    XFORM xform;
    HRGN hRgn;

    // Подготовка к выводу мяча
    HDC hBallMemDC = CreateCompatibleDC(hdc);
    SelectObject(hBallMemDC, hBmp);

    // Создаем регион отсечения
    hRgn = CreateEllipticRgn(x, y, x + bm.bmWidth, y +
    bm.bmHeight);
    SelectClipRgn(hdc, hRgn);
```

```
// Мировые преобразования для перемещения и
   //вращения мяча
   xform.eM11 = (FLOAT) cos(alpha * 2 * Pi / 360);
   //вращение
   xform.eM12 = (FLOAT) sin(alpha * 2 * Pi / 360);
   //вращение
   xform.eM21 = (FLOAT) -sin(alpha * 2 * Pi / 360);
   //вращение
   xform.eM22 = (FLOAT) cos(alpha * 2 * Pi / 360);
   //вращение
   xform.eDx  = x + bm.bmWidth / 2.;
   //смещение по оси x
   xform.eDy  = y + bm.bmHeight / 2.;
   //смещение по оси y
```

```
// Вывод мяча
    SaveDC(hdc);
    BOOL ret = SetWorldTransform(hdc, &xform);
    BitBlt(hdc, -bm.bmWidth/2, -bm.bmHeight/2,
                    bm.bmWidth, bm.bmHeight, hBallMemDC,
    0, 0, SRCCOPY);
    RestoreDC(hdc, -1);

    SelectClipRgn(hdc, NULL);
    DeleteObject(hRgn);
    DeleteDC(hBallMemDC);
```

Полный текст файла

- лекция 5 код1.docx

Вызов функции SetGraphicsMode для переключения контекста устройства в графический режим (GM_ADVANCED).

Этот режим нужен для использования мировой системы координат и мировых преобразований. В теле функции DrawBall мировые преобразования реализуются вызовом функции SetWorld-Transform, а изменяющиеся значения полей структуры xform обеспечивают эффект перемещения и вращения мяча. Эти преобразования должны использоваться *только при выводе изображения мяча.*

Поэтому перед вызовом SetWorldTransform мы запоминаем текущее состояние контекста устройства с помощью Save DC, а затем восстанавливаем его вызовом RestoreDC.

Основные события разворачиваются в блоке обработки сообщения WM_TIMER.

- Все работает замечательно, за исключением одной детали. Дело в том, что вращающийся мячик мерцает.
- Причиной этого является быстрое последовательное выполнение двух операций с контекстом дисплея: стирание прежнего изображения мяча (FillRect) и вывод нового изображения (BitBlt).
- Вы можете убедиться в этом, закомментировав вызов функции FillRect. После этого изображение перестанет мерцать, но вместо летающего мячика получится что-то вроде червя, прогрызающего тоннель в камне.

Двойная буферизация

- Неприятное мерцание изображения в анимационном приложении можно устранить, если сформировать очередную фазу картинки в *виртуальном контексте устройства.*
- Для этого используется контекст в памяти, совместимый с контекстом дисплея.
- В нашем случае очередная фаза содержит две операции:
- а) стереть предшествующее изображение мяча;
- б) нарисовать новое изображение мяча.
- После этого содержимое совместимого контекста копируется в контекст дисплея.

Двойная буферизация

- Двойная буферизация в программе реализована на основе контекста в памяти hMemDcFrame, совместимого с контекстом дисплея.
- Виртуальный контекст создается вызовом функции CreateCompatibleDC, после чего в него выбирается при помощи функции SelectObject растр hBmpFrame, также совместимый с контекстом дисплея и имеющий размеры клиентской области окна приложения.
- Инициализация контекста hMemDcFrame для копирования в него изображения фона осуществляется в блоке обработки сообщения WM_TIMER.
- Чтобы инициализация была однократной, мы используем счетчик count и вызываем функцию BitBlt только при нулевом значении счетчика.

- Также нужно позаботиться о новой инициализации контекста hMemDcFrame в случае изменения размеров окна. Для этого добавлен код обработки сообщения WM_SIZE.
- Стирание прежней картинки мяча в виртуальном контексте происходит в блоке обработки сообщения WM_TIMER.
- После вычисления новых координат мяча вызывается функция DrawBall, и ей передается виртуальный контекст в качестве параметра hMemFrameDC.
- В теле функции DrawBall завершается формирование очередной фазы картинки, когда вызывается функция BitBlt для вывода изображения мяча в виртуальный контекст.
- Только после этого вся картинка копируется при помощи второго вызова BitBlt из hMemFrameDC в контекст дисплея.

Полный код лекция 5 код2.docx

```
case WM_CREATE:
    hDC = GetDC(hWnd);
    GetClientRect(hWnd, &rect);
    dX = rect.right / 100.;
    dY = rect.bottom / 50.;

    // Создать таймер (0.1 сек)
    SetTimer(hWnd, 1, 100, NULL);

    hBmpBkgr = LoadBitmap((HINSTANCE)GetWindowLong(hWnd,
          GWL_HINSTANCE), MAKEINTRESOURCE(IDB_STONE));
    hBkBrush = CreatePatternBrush(hBmpBkgr);
    SetClassLong(hWnd, GCL_HBRBACKGROUND, (LONG)hBkBrush);
```

```
hBmpBall = LoadBitmap((HINSTANCE)GetWindowLong(hWnd,
        GWL_HINSTANCE),  MAKEINTRESOURCE(IDB_BALL));
GetObject(hBmpBall, sizeof(bm), (LPSTR)&bm);
    hMemDcFrame = CreateCompatibleDC(hDC);
    hBmpFrame = CreateCompatibleBitmap(hDC, rect.right,
    rect.bottom);
    SelectObject(hMemDcFrame, hBmpFrame);
    SetGraphicsMode(hMemDcFrame, GM_ADVANCED);
break;


case WM_SIZE:
    GetClientRect(hWnd, &rect);
    hBmpFrame = CreateCompatibleBitmap(hDC, rect.right,
    rect.bottom);
    DeleteObject(SelectObject(hMemDcFrame, hBmpFrame));
    // Копирование фона в hMemDcFrame
    BitBlt(hMemDcFrame, 0, 0, rect.right, rect.bottom, hDC, 0, 0,
    SRCCOPY);
    break;
```

```
case WM_TIMER:
    GetClientRect(hWnd, &rect);
    if (!count)
{    // Копирование фона в hMemDcFrame
    BitBlt(hMemDcFrame, 0, 0, rect.right, rect.bottom, hDC, 0,
    0, SRCCOPY);
    count++;
}
    // Стираем прежнюю картинку мяча
    SetRect(&rBall, x, y, x + bm.bmWidth, y + bm.bmHeight);
    FillRect(hMemDcFrame, &rBall, hBkBrush);
```

```
// Новая позиция мяча
    x += dX;
    y += dY;
    alpha += 10;
    if (alpha > 360)  alpha = 0;

        // Если мяч достиг края окна, направление его //движения
    изменяется
    if(x + bm.bmWidth > rect.right || x < 0)          dX = -dX;
    if(y + bm.bmHeight > rect.bottom || y < 0)     dY = -dY;

    DrawBall(hWnd, hDC, hMemDcFrame, hBmpBall, bm, (int)x, (int)y, alpha);
    break;

case WM_DESTROY:
    KillTimer(hWnd, 1);
    ReleaseDC(hWnd, hDC);
    DeleteDC(hMemDcFrame);
    PostQuitMessage(0);
    break;
```

```
void DrawBall(HWND hwnd, HDC hdc, HDC hMemFrameDC,
    HBITMAP hBmp,  BITMAP bm, FLOAT x, FLOAT y, int alpha) {
    XFORM xform;
    HRGN hRgn;

    // Подготовка к выводу мяча
    HDC hMemDcBall = CreateCompatibleDC(hdc);
    SelectObject(hMemDcBall, hBmp);

    // Создаем регион отсечения
    hRgn = CreateEllipticRgn(x, y, x + bm.bmWidth, y +
    bm.bmHeight);
    SelectClipRgn(hMemFrameDC, hRgn);
```

```
// Мировые преобразования для перемещения и вращения мяча
    xform.eM11 = (FLOAT) cos(alpha * 2 * Pi / 360);  //вращение
    xform.eM12 = (FLOAT) sin(alpha * 2 * Pi / 360);  //вращение
    xform.eM21 = (FLOAT) -sin(alpha * 2 * Pi / 360); //вращение
    xform.eM22 = (FLOAT) cos(alpha * 2 * Pi / 360);  //вращение
    xform.eDx  = x + bm.bmWidth / 2;       //смещение по оси x
    xform.eDy  = y + bm.bmHeight / 2;      //смещение по оси y

    // Вывод мяча в контекст hMemFrameDC
    SaveDC(hMemFrameDC);
    BOOL ret = SetWorldTransform(hMemFrameDC, &xform);
    BitBlt(hMemFrameDC, -bm.bmWidth/2, -bm.bmHeight/2,
           bm.bmWidth, bm.bmHeight, hMemDcBall, 0, 0, SRCCOPY);
    RestoreDC(hMemFrameDC, -1);
```

```
// Копирование изображения из hMemFrameDC в hdc
    RECT rect;
    GetClientRect(hwnd, &rect);
    BitBlt(hdc, 0, 0, rect.right, rect.bottom,
    hMemFrameDC, 0, 0, SRCCOPY);

    SelectClipRgn(hMemFrameDC, NULL);
    DeleteObject(hRgn);
    DeleteDC(hMemDcBall);
}
```