รายการอ้างอิง

ภาษาไทย

ขจร โรจนเมธินทร์. เทคนิคการเขียนโปรแกรม Turbo C++ สำหรับวินโดวส์ บริษัท ซีเอ็ดยูเคชัน จำกัด, 2537.

จิรพัฒน์ จันทร์เจิดศักดิ์ และ วีระ นพนิราพาธ. การเขียนโปรแกรมบน Microsoft Windows บริษัท ซีเอ็ดยูเคชัน จำกัด, 2536.

ดร.สมชาย ประสิทธิ์จูตระกูล และ ดร.วิทยา วัชระวิทยากุล. การออกแบบและพัฒนาระบบจินตภาพอัลกอริธึม , 2535.

บุญเลิศ เอี่ยมทัศนา. การเขียนโปรแกรมบนวินโดวส์ด้วยเทอร์โบปาสคาล บริษัท ซีเอ็ดยูเคชัน จำกัด, 2537.

นเรศ เลิศพลังสันติ และ ธงชัย วัตราเศรษฐ์. Animation ศาสตร์แห่งจินตนาการ วารสารคอมพิวเตอร์รีวิว ฉบับที่ 98 เดือน ตุลาคม 2535.

มโน และ มงคล มงคลธนานนท์. คัมภีร์โปรแกรมเมอร์ฉบับ ปีเตอร์ นอร์ตัน บริษัท อินฟอร์เมติก บิซิเนส พับลิเคชัน จำกัด, 2538.

สมพัฒน์ รุ่งตะวันเรืองศรี. เรียนรู้คอมพิวเตอร์กราฟิกส์ 2 มิติด้วยภาษา C บริษัท ซีเอ็ดยูเคชัน จำกัด, 2535.

ภาษาอังกฤษ

Adams, Lee. C for Windows Animation Programming. Windcrest/McGraw-Hill, 1993.

Barkakati, Nabajyoti. Programming Windows Games with Borland C++ SAMS Publishing, 1993.

Brown, Marc H. Algorithm Animation , The MIT Press, 1988.

_____ . Exploring algorithms using BALSA-II. COMPUTER, Vol.21 No.5 May 1988, pp. 14-36.

Comer,Douglas / Fossum, Timoty. Operating System Design Vol.I the XINU Approach Prentice Hall, 1988.

Conger, James L. Windows API BIBLE. Waite Group press, 1992.

_____. Windows Programming Primer Plus Waite Group Press, 1992.

รายการอ้างอิง (ต่อ)

Deitel, H.M. Operating Systems Second edition, Addison-Wesley, 1990.

Klien, Mike. Windows Programmer ' s Guide To DLLs and Memory management SAMS Publishing, 1992.

Lampton, Christopher. Flights of Fantasy Waite Group Press, 1993.

Mayers, Brian and Doner, Chris. Programmer 's Introduction to Windows 3.1 Sybex Inc., 1992.

Microsoft Corporation. Microsoft Windows Multimedia Programmer 's Workbook Microsoft Press, 1991.

___. Microsoft Windows Multimedia Authoring and Tools Guide Microsoft Press, 1991.

Rector, Brent E. Developing Windows 3.1 Applications with Microsoft C/C++ 2nd Edition, SAMS Publishing, 1992.

Sedgewick, Robert. Algorithms in C++ Addison-Wesley, 1992.

Stasko, John T. Tango : A framework and system for algorithm animation COMPUTER, Vol.23 No.9 Sept. 1990, pp.27-39.

Syck, Gary. Object Windows How-To Waite Group Press, 1993.

Tenenbaum, Langsam, Augenstein. Data Structures Using C Prentice-Hall, 1990.

Vince, John. 3-D Computer Animation Addison-Wesley, 1992.

Wilken and Honekamp. Windows System Programming Abacus, 1991.

Young, Douglas A. X Window Systems Programming And Application with Xt Prentice-Hall,1989.

ภาคผนวก

ภาคผนวก ก

ตัวอย่างการใช้คลังโปรแกรม

โปรแกรมต่อไปนี้เป็นตัวอย่างการนำ TAM ไปใช้ในการพัฒนาระบบการทำภาพเคลื่อนไหว ซึ่งจะเป็นแนวทางในการนำไปประยุกต์ใช้ได้ต่อไป

โปรแกรมต่อไปนี้ประกอบไปด้วย Sample.C Sample.H Sample.DEF Sample.RC

แฟ้ม Sample.H

```
// Sample.H
#define  IDM_GO       1
#define  IDM_TIMER    2
#define  IDM_18       218
#define  IDM_3        203
#define  IDM_QUIT     999
```

แฟ้ม Sample.C

```
// Program    : Sample.C  Sample.H  Sample.DEF  Sample.RC   Tam.LIB
// Author     : Palakool
// Date       : 25 March 1995
#include "tam.h"
#include "sample.h"

PICMASK  Earth[] = {
{"Earth1" ,0}, {"Earth2" ,0},   {"Earth3" ,0},  {"Earth4" ,0},  {"Earth5" ,0}, {"Earth6" ,0},
{"Earth7" ,0}, {"Earth8" ,0},   {"Earth9" ,0},  {"Earth10",0},  {"Earth11",0}, {"Earth12",0},
NULL
};
```

```
FRAME EarthFrame[] = {
{"Earth1" ,0}, {"Earth2" ,0}, {"Earth3" ,0}, {"Earth4" ,0}, {"Earth5" ,0}, {"Earth6" ,0},
{"Earth7" ,0}, {"Earth8" ,0}, {"Earth9" ,0}, {"Earth10",0}, {"Earth11",0}, {"Earth12",0},
NULL
};

LPSTR    wClass     = "SampleClass";
LPSTR    wName      = "Sample Sprite";
LPSTR    MenuName = "SampleMenu";
HANDLE        hInst;
HBMTAB        StarTab,   EarthTab;
HSEQUENCE   hSequence, hSeqReverse;

LONG  FAR PASCAL  WinProc  (HWND, unsigned, WORD, LONG);
LONG  FAR PASCAL  EarthProc(HWND, unsigned, WORD, LONG);
LONG  FAR PASCAL  StarProc (HWND, unsigned, WORD, LONG);

int PASCAL WinMain (HANDLE  hInstance,     HANDLE  hPrevInstance,
                    LPSTR     lpszCmdLine,  int   nCmdShow)
{
   HWND        hWnd;
   MSG         msg;
   WNDCLASS   wc;

   hInst = hInstance;
   if (!hPrevInstance) {
      wc.lpszClassName = wClass;
      wc.style         = CS_HREDRAW | CS_VREDRAW;
      wc.lpfnWndProc   = (WNDPROC ) WinProc;
      wc.hInstance     = hInstance;
      wc.hIcon         = NULL;
```

```
        wc.hCursor        = LoadCursor(NULL, IDC_ARROW);
        wc.hbrBackground  = GetStockObject(WHITE_BRUSH);
        wc.lpszMenuName = MenuName;
        wc.cbClsExtra     = 0;
        wc.cbWndExtra     = 0;
        if (!RegisterClass(&wc))
            return FALSE;
    }

    hWnd = CreateWindow (wClass,
                         wName,
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL );

    ShowWindow(hWnd,nCmdShow);
    while (GetMessage(&msg, NULL, NULL, NULL)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    };
    return msg.wParam;
}

LONG FAR PASCAL WinProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
    static int        nTimer=0, nGo=FALSE;
    static HSPRITE    sEarth;
    static HENVIRON   hEnv;
    static HBITMAP    hStar[35], hBmpDef;
    HCURSOR           hCurWait, hCurOld;
```

```
RECT            r;
PAINTSTRUCT     ps;
int             i, ang0;
HDC             hDC, hmDC;
HBRUSH          hBrh, hBrhDef;
POINT           pStar[10] = { 0 ,20,20,20,25,0,30,20,49,20,30,30,49,49,25,37,0 ,49,20,30 };
POINT           pRotStar[10];
int             FrameID[12] = {30,31,32,33,34,0,1,2,3,4,5,6};

switch(msg) {
  case WM_CREATE :
        SetCapture(hWnd);
        hCurWait = LoadCursor(NULL, IDC_WAIT);
        hCurOld  = SetCursor(hCurWait);
        hDC      = GetDC(NULL);
        hmDC     = CreateCompatibleDC(hDC);
        hBmpDef  = SelectObject(hmDC, NULL);
        hBrh     = CreateSolidBrush(RGB(200, 0, 100));
        hBrhDef  = SelectObject(hmDC, hBrh);
        for (i=0,ang0=0;i<35;i++, ang0+=10) {
           hStar[i] = CreateCompatibleBitmap(hDC, 50,50);
           SelectObject(hmDC, hStar[i]);
           amRotatePolygon((LPPOINT) &pStar, (LPPOINT) &pRotStar, 10, 25,25, ang0);
           PatBlt(hmDC, 0,0,50,50,BLACKNESS);
           Polygon(hmDC, (LPPOINT) &pRotStar, 10);
        }
        SelectObject(hmDC, hBrhDef);
        SelectObject(hmDC, hBmpDef);
        DeleteObject(hBrh);
        DeleteDC(hmDC);
        ReleaseDC(hWnd, hDC);
```

```
    StarTab      = amRegisterCompatibleBitmap((HBITMAP far*) &hStar[0]);
    hSequence    = amCreateSequence(StarTab,  0,0);
    hSeqReverse = amCreateSequence(StarTab,(int far *)&FrameID,12);
    EarthTab     = amRegisterBitmap(hInst,  (LPPICMASK) Earth);

    hEnv = amInitEnv(hWnd);
    SetCursor(hCurOld);
    ReleaseCapture();
    break;
case WM_COMMAND :
    switch(wParam) {
    case IDM_GO :
        if (nGo == TRUE)    break;
        sEarth = amCreateSprite("Earth",
                                EarthTab,
                                EarthFrame,
                                hEnv, 0,0,
                                100,100,hWnd,
                                (WNDPROC) EarthProc);
        nGo = TRUE;
        if(nTimer) KillTimer(hWnd, nTimer);
        nTimer = SetTimer(hWnd, 0,330, NULL);
        break;
    case IDM_18 :
        if (nTimer) KillTimer(nTimer,hWnd);
        nTimer = SetTimer(hWnd,0,55,NULL);
        break;
    case IDM_3 :
        if (nTimer) KillTimer(nTimer,hWnd);
        nTimer = SetTimer(hWnd, 0,330,NULL);
        break;
```

```
        case IDM_QUIT :
            DestroyWindow(hWnd);
            break;
        }
        break;
    case WM_SIZE :
        hDC = amGetAnimateDC(hEnv);
        GetClientRect(hWnd, &r);
        amNightSky(hDC, r.left, r.top, r.right, r.bottom);
        amGetBkGdImage(hEnv);
        amReleaseAnimateDC(hEnv, hDC);
        break;
    case WM_PAINT :
        BeginPaint(hWnd, &ps);
        amRefreshAll(NULL, hEnv);
        EndPaint(hWnd, &ps);
        break;
    case WM_TIMER :
        amBroadCastMessage(hEnv, msg, wParam, lParam);
        amAnimate(NULL , hEnv, NORMAL);
        break;
    case WM_DESTROY :
        KillTimer(hWnd, nTimer);
        amDeleteSequence(hSequence);
        amDeleteSequence(hSeqReverse);
        amClearAllSprite(hEnv);
        amUnRegisterBitmap(StarTab);
        amUnRegisterBitmap(EarthTab);
        amClearEnv(hEnv);
        PostQuitMessage(0);
        break;
```

```
    default :
        return DefWindowProc(hWnd, msg, wParam, lParam);
  }
  return FALSE;
}

LONG FAR PASCAL EarthProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
  static HSPRITE    sEarth, sStar,sNew;
  static HENVIRON   hEnv;
  int               nFrame;

  switch(msg) {
    case SPR_MSG_CREATE :
        sEarth = wParam;
        hEnv   = (WORD) lParam;
        sStar  = amCreateSprite("Star",
                        NULL, NULL,  hEnv, 0,0, 50,100, hWnd, (WNDPROC) StarProc);
        amSelectSequence(sStar, hSequence);
        sNew   = amCreateSprite("DEFAULT",
                        NULL, NULL, hEnv, 0,0,200,100, hWnd, (WNDPROC) NULL);
        amSelectSequence(sNew, hSeqReverse);
        amSetSpriteProp (sNew, SPR_SETCURFRAME, 7);
        break;
    case WM_TIMER :
        amSpriteFrame(sEarth, FRM_LOOP);
        amSpriteFrame(sNew, FRM_AUTOREVERSE);
        amGetEllipseNextXY(sEarth, 250,100, 150,30, 1);
        amSpriteMove(sNew, MOV_AUTOREVERSE,1,1);
        amUpdateSprite(sEarth);
        amUpdateSprite(sNew);
```

```
            break;

        default :
            return DefWindowProc(hWnd, msg, wParam, lParam);
    }
    return FALSE;
}

LONG FAR PASCAL StarProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
    static HSPRITE    hSpr,hParent;
    static HENVIRON   hEnv;
    COORD             p,c;
    HDC               hDC;

    switch(msg) {
        case SPR_MSG_CREATE :
            hSpr    = wParam;
            hParent = amGetSpriteHandle(GetParent(hWnd));
            hEnv    = (WORD) lParam;
            break;
        case WM_TIMER :
            amSpriteFrame(hSpr, FRM_LOOP);
            amGetSpriteXYZ (hParent,(LPCOORD) &p);
            amGetEllipseNextXY(hSpr, p.x+50,p.y+50, 150,40, 5);
            amGetSpriteXYZ(hSpr,    (LPCOORD) &c);
            amGetSpriteXYZ(hParent, (LPCOORD) &p);
            if (c.y < p.y+50) {
               amSetSpriteProp(hSpr,    SPR_CURZ_SET, 0);
               amSetSpriteProp(hParent, SPR_CURZ_SET, 1);
            }
```

```
            else {
                amSetSpriteProp(hSpr,    SPR_CURZ_SET, 1);
                amSetSpriteProp(hParent, SPR_CURZ_SET, 0);
            }

            amReOrderSprite(hEnv);
            amUpdateSprite(hSpr);
            break;
        default :
            return DefWindowProc(hWnd, msg, wParam, lParam);
    }
    return FALSE;
}
```

แฟ้ม Sample.DEF

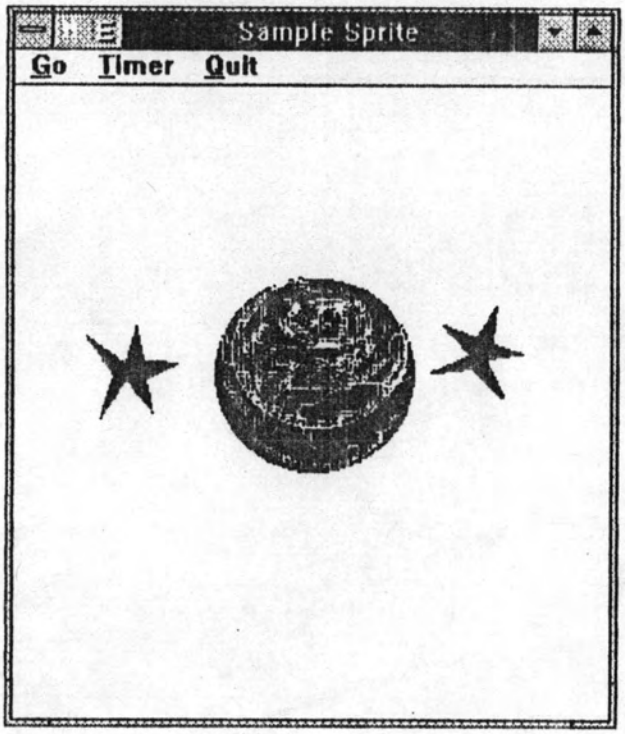
```
NAME        Sample
EXETYPE     windows
CODE        preload  moveable
DATA        preload  moveable MULTIPLE
HEAPSIZE    8192
STACKSIZE   8192
EXPORTS     WinProc  @1
            StarProc  @2
            EarthProc  @3
```

แฟ้ม Sample.RC

```
#include <windows.h>
#include "sample.h"
Earth1  BITMAP  E01.bmp
Earth2  BITMAP  E02.bmp
Earth3  BITMAP  E03.bmp
```

```
Earth4  BITMAP  E04.bmp
Earth5  BITMAP  E05.bmp
Earth6  BITMAP  E06.bmp
Earth7  BITMAP  E07.bmp
Earth8  BITMAP  E08.bmp
Earth9  BITMAP  E09.bmp
Earth10 BITMAP  E10.bmp
Earth11 BITMAP  E11.bmp
Earth12 BITMAP  E12.bmp
SampleMenu  MENU
{
    MENUITEM  "&Go",   IDM_GO
    POPUP     "&Timer"
    {
        MENUITEM "18 Frame/Sec.", IDM_18
        MENUITEM " 3 Frame/Sec.", IDM_3
    }
    MENUITEM  "&Quit", IDM_QUIT
}
```

เมื่อทำการแปลโปรแกรมแล้วจะได้ผลดังนี้


Sample Sprite
Go Timer Quit

ภาคผนวก ข

ฟังก์ชันสนับสนุนการจัดการภาพเคลื่อนไหว

BOOL amCheckCollision (HSPRITE hSprite, HENVIRON hEnviron);

วัตถุประสงค์ : ตรวจสอบการชนกันของสไปรต์

Return Value : TRUE = ถ้าชนกัน FASLE = ไม่ชน

HBMTAB amRegisterBitmap(HANDLE hInstance, LPPICMASK pm);

วัตถุประสงค์ : สร้างคลังภาพจากชื่อที่ระบุใน array ของโครงสร้าง PICMASK

Return Value : ค่าแฮนเดิลของคลังภาพ นำไปใช้ในการสร้างลำดับภาพของสไปรต์ (Sequence) ต่อไป

VOID amUnRegisterBitmap(HBMTAB hBmTab);

วัตถุประสงค์ : ยกเลิกคลังภาพที่สร้างขึ้นมาจากฟังก์ชัน amRegisterBitmap() และ amRegisterCompatibleBitmap()

Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amDeleteSprite(HENVIRON hEnviron, HSPRITE hSprite);

วัตถุประสงค์ : ยกเลิกสไปรต์ ที่สร้างขึ้นจากฟังก์ชัน amCreateSprite()

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : ต้องใช้ amDeleteSprite() เมื่อไม่ต้องการสไปรต์เสมอ เพื่อคืนทรัพยากรให้ระบบ

VOID amRefreshBackground(HDC hDC, HENVIRON hEnviron);

วัตถุประสงค์ : วาดเฉพาะภาพพื้นหลังใหม่

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : สไปรต์ที่อยู่บนภาพพื้นหลังจะถูกลบไปด้วย

VOID amRefreshAll(HDC hDC, HENVIRON hEnviron);

วัตถุประสงค์ : วาดภาพพื้นหลังและสไปรต์ทุกตัวใหม่

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : วาดเฉพาะสไปรต์ที่มีสถานภาพ active หรือ need update หรือ visible เท่านั้น

VOID amGetBkGdImage(HENVIRON hEnviron);

วัตถุประสงค์ : บันทึกภาพพื้นหลังที่กำลังปรากฏอยู่บนวินโดว์เก็บไว้ในสภาพแวดล้อมเพื่อใช้งาน

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : ถ้าไม่ใช้ amLoadBkGdImage() ให้ใช้ amGetBkGdImage() เพื่อเก็บบันทึกภาพพื้นหลัง

VOID amDeleteBkGdImage(HENVIRON hEnviron);

วัตถุประสงค์ : ยกเลิกภาพพื้นหลังที่อยู่ในสภาพแวดล้อม

Return Value : ไม่ส่งค่าใดๆกลับมา

BOOL amLoadBkGdImage(HWND hWnd, LPSTR bkName, HENVIRON hEnviron, char bkSize);

วัตถุประสงค์ : อ่านแฟ้มข้อมูลภาพเพื่อนำมาใช้เป็นภาพพื้นหลังในวินโดว์

Parameter : bkSize ถ้ากำหนดเป็น TRUE จะได้ภาพพื้นหลังตามขนาดที่แท้จริงของภาพ ถ้ากำหนดเป็น FALSE จะทำการขยายหรือหดภาพให้พอดีกับขนาดของวินโดว์

Return Value : TRUE เมื่อ อ่านแฟ้มข้อมูลภาพได้สำเร็จ

หมายเหตุ : ถ้าไม่ใช้ amLoadBkGdImage() ให้ใช้ amGetBkGdImage() เพื่อเก็บบันทึกภาพพื้นหลัง

VOID amSpriteMove(HSPRITE hspite, int type, int dx, int dy);

วัตถุประสงค์ : ย้ายตำแหน่งของสไปรต์ จากตำแหน่งปัจจุบันไปตามแนวนอน dx จุดภาพ และแนวตั้ง dy จุดภาพ ตามชนิดการเคลื่อนที่ที่ระบุใน int type

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : type คือวิธีการเคลื่อนที่ของสไปรต์ ได้แก่ MOV_UP , MOV_DOWN, MOV_LEFT, MOV_RIGHT, MOV_AUTOREVERSE (ถอยกลับเมื่อชนกรอบวินโดว์)

int amSpriteFrame(HSPRITE hsp, int type);

วัตถุประสงค์ : เปลี่ยนภาพของสไปรต์ ตามชนิดที่ระบุใน int type

Reutrn Value : หมายเลขบอกภาพปัจจุบัน (ปัจจุบันเป็นภาพที่เท่าไร)

หมายเหตุ : type คือวิธีเปลี่ยนภาพ ได้แก่

FRM_FORWARD เลื่อนภาพต่อไป 1 ภาพ (เดินหน้า)

FRM_BACKWARD เลื่อนภาพถอยหลัง 1 ภาพ

FRM_LOOP	เลื่อนภาพ (เดินหน้า / ถอยหลัง) เมื่อถึงภาพสุดท้ายให้เริ่มต้นใหม่
FRM_AUTOREVERSE	เลื่อนภาพ (เดินหน้า / ถอยหลัง) เมื่อถึงภาพสุดท้ายให้เลื่อนภาพกลับ
FRM_STOP	หยุดการเลื่อนภาพ
FRM_CURRENT	สอบถามหมายเลขของภาพปัจจุบัน
FRM_FIRST	เลื่อนไปที่ภาพแรก
FRM_LAST	เลื่อนไปที่ภาพสุดท้าย

VOID amSpriteAT(HSPRITE hSprite, LPPOINT lpPoint);

วัตถุประสงค์ : ถามตำแหน่งปัจจุบันของสไปรต์ ว่าอยู่ที่ x,y เท่าไร

Return Value : ไม่ส่งค่าใดๆกลับมา แต่จะได้ค่าของตำแหน่งปัจจุบันของสไปรต์ในโครงสร้าง POINT

หมายเหตุ : ตรวจสอบตำแหน่งของสไปรต์จาก POINT

ตัวอย่าง :

```
POINT Point;
amSpriteAT(hSprite, (LPPOINT) &Point);
TextOut(hDC, 10,10, buf, wsprintf(buf, "CurX = %3d, CurY = %3d", Point.x, Point.y);
```

VOID amSetSpriteXY(HSPRITE hSprite, int x, int y);

วัตถุประสงค์ : กำหนดตำแหน่งให้สไปรต์ไปอยู่ที่ x,y

Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amSetSpriteXYZ(HENVIRON hEnv, HSPRITE hSprite, int newX, int newY, int newZ);

วัตถุประสงค์ : เปลี่ยนตำแหน่งและระดับการเคลื่อนที่ของสไปรต์ใหม่

Reutrn Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : ฟังก์ชันนี้จะทำการ Set ค่า ReOrderFlag ให้เป็น TRUE ถ้ามีการเปลี่ยนระดับการเคลื่อนที่ของสไปรต์

HENVIRON amInitEnv(HWND hDispWnd);

วัตถุประสงค์ : ใช้ในการสร้างสภาพแวดล้อมของระบบ hDispWnd คือแฮนเดิลของวินโดว์ที่มีภาพเคลื่อนไหวของสไปรต์

Return Value : ส่งค่าแฮนเดิลของสภาพแวดล้อมของระบบมาให้

หมายเหตุ : ต้องเรียกฟังก์ชันนี้ก่อนฟังก์ชันอื่นใน TAM เสมอ มิฉะนั้นระบบจะเกิด Protection Fault

และหยุดทำงาน

ตัวอย่าง :

```
HWND      hWnd;
HENVIRON  hEnv;
hWnd  = CreateWindow( .....);
hEnv  = amInitEnv(hWnd);
```

VOID amClearEnv(HENVIRON hEnviron);

วัตถุประสงค์ : เรียกฟังก์ชันนี้เมื่อต้องการยกเลิกสภาพแวดล้อมต่างๆ รวมทั้งยกเลิกทรัพยากรที่ TAM สร้างขึ้นมา amClearEnv() จะต้องถูกเรียกเป็นฟังก์ชันสุดท้ายของ TAM เสมอ

ตัวอย่าง :

```
HENVIRON  hEnv;
case   WM_CREATE :
       hEnv = amInitEnv(hWnd);
...
case   WM_DESTROY :
       amClearEnv(hEnv);
       PostQuitMessage(0);
       break;
```

Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amClearAllSprite(HENVIRON hEnviron);

วัตถุประสงค์ : ยกเลิกสไปรต์ทุกตัวในสภาพแวดล้อม

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : ใช้เมื่อต้องการยกเลิกสไปรต์ทุกตัว มีประสิทธิภาพกว่า amDeleteSprite ทีละตัว

LPSPRITE amGetSprite(HWND hWnd);

วัตถุประสงค์ : ใช้ขอพอยเตอร์ชี้โครงสร้างข้อมูลของสไปรต์

Return Value : พอยเตอร์ชี้โครงสร้างข้อมูลของสไปรต์

หมายเหตุ : ใช้ฟังก์ชันนี้เฉพาะในฟังก์ชันประจำตัวของสไปรต์ ทำให้สามารถแก้ไขข้อมูลสไปรต์ได้ แต่ไม่รับประกันความถูกต้องเนื่องจากจะต้องเป็นความรับผิดชอบของผู้เขียนโปรแกรมเอง

ตัวอย่าง :

```
LONG FAR PASCAL SpriteProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
    LPSPRITE lpSprite;
    ...
        case WM_USER :
            lpSprite = amGetSprite(hWnd);
            lpSprite->LastX = lpSprite->CurX;
            lpSprite->CurX +=10;
            amReleaseSprite(hWnd);
            break;
    ...
}
```

int amReleaseSprite(HWND hWnd);

วัตถุประสงค์ : ยกเลิกการใช้ amGetSprite();

Return Value : 0 ถ้าคืนสำเร็จ ถ้าไม่สำเร็จจะส่งค่า address ของสไปรต์ที่คืนไม่สำเร็จ

หมายเหตุ : เมื่อใช้ amGetSprite () ต้องเรียก amReleaseSprite() คู่กัน

HBMTAB amRegisterCompatibleBitmap(HBITMAP far *hbmPic);

วัตถุประสงค์ : ใช้สร้างคลังภาพที่ไม่ใช่แฟ้มข้อมูลภาพ (เป็นภาพที่เกิดจากโปรแกรมสร้างเอง)

Return Value : แฮนเดิลของคลังภาพที่สร้างได้

HSEQUENCE amCreateSequence(HBMTAB hBmTab, LPVOID hFrame, char type);

วัตถุประสงค์ : สร้างลำดับภาพเพื่อใช้กับสไปรต์

Return Value : แฮนเดิลของลำดับภาพ

หมายเหตุ : hBmTab เป็นคลังภาพที่สร้างจาก amRegisterBitmap หรือ amRegisterCompatibleBitmap

hFrame เป็นชื่อภาพในคลังที่ต้องการ ถ้าเป็น NULL หมายถึงต้องการทุกภาพ

type เป็น 0 เมื่อ hFrame เป็นแถวลำดับของรายชื่อของภาพ

ถ้า hFrame เป็นแถวลำดับของหมายเลขของภาพ type จะเป็นจำนวนของภาพ

ที่ต้องการ

HSEQUENCE amSelectSequence(HSPRITE hSprite, HSEQUENCE hSeq);

วัตถุประสงค์ : เลือกลำดับภาพให้สไปรต์

Return Value : ส่งค่าลำดับภาพเดิมของสไปรต์กลับมา

ตัวอย่าง :

```
HSEQUENCE   hSeq;
. .
hSprite = amCreateSprite( . . );
hSeq    = amCreateSequence(hBmTab, NULL, 0);
amSelectSequence (hSprite, hSeq),
```

VOID amDeleteSequence(HSEQUENCE hSeq);

วัตถุประสงค์ : ยกเลิกลำดับภาพ

Return Value : ไม่ส่งค่าใดๆกลับมา

HSPRITE amCreateSprite(LPSTR SprClass, HBMTAB hBmTab, LPFRAME lpFrame,
HENVIRON hEnv, int Priority, char Status,
int x, int y, HWND hWnd, WNDPROC Proc);

วัตถุประสงค์ : สร้างสไปรต์ขึ้นมา

Return Value : ค่าแฮนเดิลของสไปรต์ที่สร้างได้

ตัวอย่าง :

```
HSPRITE   hSprite;
HENVIRON  hEnv;

hEnv  = amInitEnv(hWnd);
hSprite = amCreateSprite("BirdClass",
                         BirdTab,
                         BirdFrame,
                         hEnv,  0, SPR_DEFAULT,
                         200,100,  hWnd, BirdProc);
```

VOID amBroadCastMessage(HENVIRON hEnv, unsigned msg, WORD wParam, LONG lParam);

วัตถุประสงค์ : ส่งข้อความไปให้สไปรต์ทุกตัวในสภาพแวดล้อม

Return Value : ไม่มีการส่งค่ากลับมา

หมายเหตุ : ใช้แทน SendMessage หรือ PostMessage

VOID amSetSpriteProp(HSPRITE hSprite, int Item, int Value);

วัตถุประสงค์ : กำหนดค่าคุณสมบัติให้แก่สไปรต์

Return Value : ไม่มีการส่งค่าใดๆกลับมา

หมายเหตุ : Item เป็นค่าระบุเขตข้อมูลของสไปรต์ที่ต้องการกำหนดค่า

Value เป็นค่าที่ต้องการกำหนดค่าต่างๆของ Item ได้แก่

SPR_SETCURFRAME = ใช้เปลี่ยนภาพในลำดับภาพของสไปรต์ ต้องการภาพที่เท่าไรก็บอกใน Value

SPR_CURZ_SET = ใช้เปลี่ยนระดับชั้นการเคลื่อนที่ของสไปรต์ ระบุระดับการเคลื่อนที่ใน Value

SPR_CURZ_INC = เพิ่มระดับชั้นการเคลื่อนที่ของสไปรต์ตามค่าที่กำหนดใน Value

SPR_CURZ_DEC = ลดระดับชั้นการเคลื่อนที่ของสไปรต์ตามค่าที่กำหนดใน Value

SPR_MOVEDIR = ใช้เปลี่ยนทิศทางการเคลื่อนที่ของสไปรต์ตามค่าที่กำหนดใน Value

SPR_BMDIR = ใช้เปลี่ยนทิศทางการแสดงภาพของสไปรต์ตามค่าที่กำหนดใน Value

SPR_PRIORITY = กำหนดค่าระดับความสำคัญในการดำเนินงานของสไปรต์ตามค่าใน Value

SPR_LIFETIME = กำหนดค่าช่วงเวลาชีวิตของสไปรต์

SPR_STATUS = กำหนดสถานภาพของสไปรต์

HSPRITE amGetSpriteHandle(HWND hWnd);

วัตถุประสงค์ : ขอค่าแฮนเดิลของสไปรต์ ใช้ในฟังก์ชันประจำตัวสไปรต์

Return Value : ค่าแฮนเดิลของสไปรต์ที่ได้รับข้อความ

ตัวอย่าง : ให้สไปรต์แสดงภาพแบบ autoreverse และเคลื่อนที่ตามแนวตั้ง 2 จุดภาพ ตามแนวนอน 3 จุดภาพ

```
LONG FAR PASCAL SpriteProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
    HSPRITE hSprite;
    ...
```

```
switch (msg) {
   case WM_TIMER :
        hSprite = amGetSpriteHandle (hWnd);
        amSetSpriteProp(hSprite, SPR_BMDIR, FRM_AUTOREVERSE));
        amSpriteMove(hSprite, 2, 5);
        amUpdateSprite(hSprite);
        break;
   ...
}
```

VOID amMessageToSprite(HSPRITE hSprite, unsigned msg, WORD wParam, LONG lparam);

วัตถุประสงค์ : ส่งข้อความไปยังสไปรต์

Return Value : ไม่มีการส่งค่าใดๆกลับมา

หมายเหตุ : ส่งข้อความไปที่ฟังก์ชันประจำตัวสไปรต์

BOOL amGetCircleNextXY(HSPRITE hSprite, int xC, int yC, int r, int d0);

วัตถุประสงค์ : ขอตำแหน่งต่อไปในการเคลื่อนที่เป็นวงกลมของสไปรต์

Return Value : TRUE ถ้าหาค่าสำเร็จ

หมายเหตุ : xC , yC คือจุดศูนย์กลางการเคลื่อนที่เป็นวงกลมของสไปรต์

r คือ รัศมีการเคลื่อนที่เป็นวงกลมของสไปรต์

d0 คือ การเคลื่อนที่ครั้งละกี่องศาจากจุดที่แล้ว

ตัวอย่าง : ให้สไปรต์เคลื่อนที่เป็นวงกลมรอบจุด (200,200) รัศมีการเคลื่อนที่ 150 จุดภาพ และเคลื่อน
ที่ทีละ 5 องศา

```
LONG FAR PASCAL SpriteProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
    HSPRITE hSprite;
    ...
    switch (msg) {
       case WM_TIMER :
            hSprite = amGetSpriteHandle(hWnd);
```

```
            amGetCircleNextXY(hSprite,  200,200, 150, 5);
            amUpdateSprite(hSprite);
            break;
        }
}
```

BOOL amGetEllipseNextXY(HSPRITE hSprite, int xC, int yC, int rA, int rB, int d0);

วัตถุประสงค์ : ขอตำแหน่งต่อไปในการเคลื่อนที่เป็นวงรีของสไปรต์

Return Value : TRUE ถ้าขอได้สำเร็จ

หมายเหตุ : xC , yC คือจุดศูนย์กลางการเคลื่อนที่เป็นวงรีของสไปรต์

rA, rB คือ รัศมีการเคลื่อนที่แนวตั้งและแนวนอนเป็นวงรีของสไปรต์

d0 คือ การเคลื่อนที่ครั้งละกี่องศาจากจุดที่แล้ว

ตัวอย่าง : ดูจากตัวอย่างของ amGetCircleNextXY()

BOOL amSwapSpritePos(HENVIRON hEnviron, HSPRITE hSprite1, HSPRITE hSprite2);

วัตถุประสงค์ : สลับระดับการเคลื่อนที่ของสไปรต์ 2 ตัว

Return Value : TRUE ถ้าสลับได้

int amGetSpritePos(HENVIRON hEnviron, HSPRITE hSprite);

วัตถุประสงค์ : ขอค่าระดับการเคลื่อนที่ของสไปรต์

Return Value : ค่าระดับการเคลื่อนที่ของสไปรต์ (Current Z)

VOID amUpdateSprite(HSPRITE hSprite);

วัตถุประสงค์ : กำหนดค่าสถานะของสไปรต์ว่าต้องการ update ภาพของสไปรต์บนจอ

Return Value : ไม่มีการส่งค่าใดๆกลับมา

หมายเหตุ : ใช้เมื่อต้องการให้ตัวจักร update สไปรต์บนจอภาพ

ตัวอย่าง : สไปรต์เคลื่อนที่ทางขวาทีละ 1 จุดภาพ

```
LONG FAR PASCAL SpriteProc(HWND hWnd, unsigned msg, WORD wParam, LONG lParam)
{
    static HSPRITE hSprite;
    ...
```

```
        switch (msg) {
            case  SPR_MSG_CREATE  :  // จะส่งข้อความนี้เมื่อมีการสร้างสไปรต์เสร็จแล้ว
                  hSprite = amGetSpriteHandle(hWnd);
                  amSetSpriteProp(hSprite, SPR_MOVEDIR, MOV_RIGHT);
                  break;                    // ไม่มีการ update บนจอภาพ
            case  WM_TIMER :
                  amSpriteMove(hSprite, 1,0);
                  amUpdateSprite(hSprite);   // บอกตัวจักรว่าสไปรต์ตัวนี้ต้องการ update บนจอภาพ
                  break;
        }
}
```

BOOL amNeedUpdateSprite(HSPRITE hSpr);

วัตถุประสงค์ : สอบถามว่าสไปรต์ตัวนี้ร้องขอ update หรือยัง

Return Value : TRUE ถ้ากำลังขอ update อยู่

BOOL amIsActiveSprite(HSPRITE hSpr);

วัตถุประสงค์ : สอบถามว่าสไปรต์ตัวนี้ ยังดำเนินงานอยู่หรือไม่

Return Value : TRUE ถ้ายังดำเนินงานอยู่

VOID amSetSpriteActive(HSPRITE hSprite);

วัตถุประสงค์ : กำหนดสถานภาพของสไปรต์เป็น active (ยังดำเนินงานอยู่)

Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amSetSpriteInActive(HSPRITE hSpr);

วัตถุประสงค์ : กำหนดสถานภาพของสไปรต์ให้หยุดดำเนินงานชั่วคราว (inactive)

Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amSetSpriteInvisible(HSPRITE hSpr);

วัตถุประสงค์ : กำหนดสถาพภาพของสไปรต์ให้เป็น invisible คือบอกตัวจักรว่าไม่ต้องวาดสไปรต์นี้

Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amAnimate(HDC hDC, HANDLE hEnviron, char mode);

วัตถุประสงค์ : ปลุกให้ตัวจักรทำภาพเคลื่อนไหวดำเนินงาน

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : hDC เป็น Display Context หรือ วินโดว์ที่จะจัดการทำภาพเคลื่อนไหว ถ้ากำหนดเป็น NULL ตัวจักรจะดำเนินงานที่วินโดว์หลัก (วินโดว์ที่เรียก amInitEnv (hWnd) ควรเรียก amAnimate ในวินโดว์หลัก เพียงครั้งเดียวตามเหตุการณ์ที่กำหนด เนื่องจาก amAnimate จะจัดการ update สไปรต์ทุกตัวที่มีสถานภาพต้องการ update

mode มีให้เลือกดังนี้

NORMAL ใช้ในกรณีที่มีสไปรต์จำนวนน้อยๆ (ไม่ควรเกิน 20 ตัว)

OPTIMIZED ใช้ในกรณีที่มีสไปรต์จำนวนมาก

FRAMEOUT ใช้เมื่อต้องการให้วาดทั้งจอภาพหรือมีการใช้ข้อความเลื่อนไปบนจอภาพ

HDC amGetAnimateDC(HENVIRON hEnv);

วัตถุประสงค์ : ขอ Display Context หรือ วินโดว์หลักที่ใช้ในการดำเนินการภาพเคลื่อนไหว

Return Value : แฮนเดิลของ Display Context ของวินโดว์ที่จะทำภาพเคลื่อนไหว

หมายเหตุ : มักใช้ในฟังก์ชันประจำตัวสไปรต์ เมื่อใช้ Display Context นี้แล้วต้องคืนให้ระบบด้วย ฟังก์ชัน amReleaseAnimateDC()

VOID amReleaseAnimateDC(HENVIRON hEnv, HDC hDC);

วัตถุประสงค์ : คืน Display Context ที่ขอจากฟังก์ชัน amGetAnimateDC();

Return Value : ไม่ส่งค่าใดๆกลับมา

HWND amGetAnimateWnd(HENVIRON hEnv);

วัตถุประสงค์ : ขอค่าแฮนเดิลของวินโดว์ที่ดำเนินการภาพเคลื่อนไหว

Return Value : ค่าแฮนเดิลของวินโดว์ที่ต้องการ

VOID amSetFrameStatus(HSPRITE hSprite, int nFrameNumber, char mStatus);

วัตถุประสงค์ : กำหนดค่าสถานภาพการวาดของภาพในลำดับภาพของสไปรต์

Return Value : ไม่ส่งค่าใดๆกลับมา

หมายเหตุ : nFrameNumber เป็นหมายเลขของภาพว่าต้องการกำหนดให้กับภาพที่เท่าไร

mStatus เป็นสถานภาพการวาดภาพได้แก่

FRM_NORMAL วาดตามปรกติตามภาพเดิม
FRM_HORZ_DIR วาดภาพแบบกลับด้านตามแนวนอน
FRM_VERT_DIR วาดภาพแบบกลับด้านตามแนวตั้ง (กลับหัว)
FRM_MIX_DIR วาดภาพแบบกลับด้านตามแนวตั้งและแนวนอน

int amGetFrameStatus(HSPRITE hSprite, int nFrameNumber);
วัตถุประสงค์ : สอบถามสถานภาพการวาดภาพของสไปรต์ตามหมายเลขของภาพนั้น
Return Value : สถานภาพการวาด ตามที่อธิบายในฟังก์ชัน amSetFrameStatus()

HBMTAB amMakeMirrorBitmap(HBMTAB hBmTab, char mStatus);
วัตถุประสงค์ : สร้างคลังภาพเช่นเดียวกับ amRegisterBitmap() แต่วาดภาพตามสถานภาพการวาดที่ระบุใน mStatus แล้วจึงเก็บในคลังภาพ ทำให้การวาดภาพบนวินโดว์มีความเร็วขึ้น เนื่องจากใช้ BitBlt() โดยตรง
Return Value : ส่งค่าแฮนเดิลของคลังภาพที่สร้างได้กลับมาให้

VOID amReOrderSprite(HENVIRON hEnviron);
วัตถุประสงค์ : บอกให้ตัวจักรทำการเรียงลำดับของสไปรต์ในสภาพแวดล้อมก่อนวาดไปยังจอภาพ
Return Value : ไม่ส่งค่าใดๆกลับมา

VOID amSpriteDeltaXYZ(HENVIRON hEnviron, HSPRITE hSprite, int dX, int dY, int dZ);
วัตถุประสงค์ : เปลี่ยนตำแหน่งของสไปรต์ทั้งแนวนอน แนวตั้ง และระดับการเคลื่อนที่ด้วยฟังก์ชันเดียว
Return Value : ไม่ส่งค่าใดๆกลับมา
หมายเหตุ : ถ้าในโปรแกรมต้องการเปลี่ยนค่าทั้งสามพร้อมกัน ใช้ฟังก์ชันนี้จะเร็วกว่า amSetSpritProp()

VOID amGetSpriteXYZ(HSPRITE hSprite, LPCOORD coord);
วัตถุประสงค์ : สอบถามตำแหน่งตามแนวนอน แนวตั้ง และระดับการเคลื่อนที่ของสไปรต์
Return Value : ไม่ส่งค่ากลับมา แต่ผลลัพธ์ที่ขอจะเก็บไว้ในโครงสร้าง COORD ซึ่งมีสมาชิกดังนี้

```
typedef struct tagCOORD {
```

```
    int     x;      // ตำแหน่งปัจจุบันตามแนวนอน
    int     y;      // ตำแหน่งปัจจุบันตามแนวตั้ง
    int     z;      // ระดับการเคลื่อนที่ในปัจจุบันของสไปรต์
} COORD;
```

VOID amGetSpriteProp(HSPRITE hSprite, int nRequest, VOID FAR *Value);

วัตถุประสงค์ : สอบถามข้อมูลเกี่ยวกับตัวสไปรต์

Return Value : ไม่ส่งค่ากลับมา แต่ผลลัพธ์จะอยู่ใน Value ตามชนิดของข้อมูลที่ขอ (ระบุใน nRequest)

หมายเหตุ : nRequest คือคำร้องขอข้อมูล ได้แก่

SPR_REQ_XYZ	สอบถามตำแหน่งปัจจุบันของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของโครงสร้าง COORD
SPR_REQ_STATUS	สอบถามสถานภาพของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ char
SPR_REQ_PRIORITY	สอบถามระดับความสำคัญในการดำเนินงานของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ int
SPR_REQ_CURPRIORITY	สอบถามระดับความสำคัญในขณะนั้นของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ int
SPR_REQ_MAXFRAME	สอบถามจำนวนภาพในลำดับภาพของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ int
SPR_REQ_CURFRAME	สอบถามหมายเลขของภาพปัจจุบันในลำดับภาพของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ int
SPR_REQ_BMSIZE	สอบถามขนาดของสไปรต์ (ขนาดของภาพของสไปรต์)
	Value จะต้องเป็นพอยเตอร์ของ long
	โดยความกว้างอยู่ที่ LOWORD, ความสูงอยู่ที่ HIWORD
SPR_REQ_LASTBMSIZE	สอบถามขนาดของสไปรต์ที่ผ่านมาแล้ว (ขนาดของภาพของสไปรต์)
	Value จะต้องเป็นพอยเตอร์ของ long
	โดยความกว้างอยู่ที่ LOWORD, ความสูงอยู่ที่ HIWORD
SPR_REQ_MOVEDIR	สอบถามทิศทางการเคลื่อนที่ของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ char
SPR_REQ_HDISPWND	สอบถามค่าแฮนเดิลของวินโดว์ที่ทำภาพเคลื่อนไหว
	Value จะต้องเป็นพอยเตอร์ของ HANDLE

SPR_REQ_COLLSTATUS	สอบถามสถานภาพการชนกันของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ char
SPR_REQ_HBMTAB	สอบถามค่าแฮนเดิลของคลังภาพที่สไปรต์กำลังใช้งานอยู่
	Value จะต้องเป็นพอยเตอร์ของ HANDLE
SPR_REQ_HFRAMELIST	สอบถามค่าแฮนเดิลของลำดับภาพที่สไปรต์ใช้อยู่
	Value จะต้องเป็นพอยเตอร์ของ HANDLE
SPR_REQ_BMDIR	สอบถามทิศทางการเลื่อนภาพในลำดับภาพของสไปรต์
	Value จะต้องเป็นพอยเตอร์ของ char

BOOL amSpriteMoveTo(HSPRITE hSprite, int x, int y, int nStep);

วัตถุประสงค์ : กำหนดเส้นทางเดินให้สไปรต์

Return Value : TRUE ถ้ายังเดินไม่ถึงจุดหมาย (x,y)

FALSE ถ้าเดินถึงจุดหมายแล้ว

หมายเหตุ : x, y คือจุดหมายที่สไปรต์เคลื่อนที่ไป

nStep คือระยะที่สไปรต์จะเคลื่อนที่ต่อหนึ่งครั้ง

HSPRPATH amCreatePath(LPPOINT lpPoint, int nPoint);

วัตถุประสงค์ : สร้างเส้นทางเดิน (path) ให้สไปรต์

Return Value : ค่าแฮนเดิลของเส้นทางเดินของสไปรต์

หมายเหตุ : lpPoint คือแถวลำดับโครงสร้าง POINT ที่เก็บตำแหน่งที่สไปรต์จะเคลื่อนที่ผ่าน (vertices)

nPoint คือจำนวนตำแหน่งที่สไปรต์เคลื่อนที่ผ่าน (จำนวนสมาชิกใน lpPoint)

BOOL amDeletePath(HSPRPATH hPath);

วัตถุประสงค์ : ยกเลิกเส้นทางเดินของสไปรต์ที่สร้างจากฟังก์ชัน amCreatePath()

Return Value : TRUE ถ้ายกเลิกค่า hPath ได้

HSPRPATH amSelectPath(HSPRITE hSprite, HSPRPATH hPath);

วัตถุประสงค์ : เลือกเส้นทางเดินให้สไปรต์

Return Value : ค่าแฮนเดิลของเส้นทางเดินเก่า

BOOL amSpriteMoveOnPath(HSPRITE hSprite, int nStep);

วัตถุประสงค์ : สั่งให้สไปรต์เคลื่อนที่ไปตามเส้นทางที่เลือกให้แล้วด้วยฟังก์ชัน amSelectPath()
Return Value : TRUE กำลังเดินอยู่ FALSE ถึงจุดหมายแล้ว

VOID amFrameAnimate(HENVIRON hEnv);
วัตถุประสงค์ : ตัวจักรทำภาพเคลื่อนไหวชนิด update ทั้งวินโดว์
Return Value : ไม่ส่งค่าใดๆกลับมา
หมายเหตุ : ที่ฟังก์ชันประจำตัวสไปรต์จะได้รับข้อความ AM_FRAMEANIMATE ดังนี้
msg = AM_FRAMEANIMATE
wParam = hSprite
lParam = hEnv / hDC (hEnv = LOWORD(lParam); hDC = HIWORD(lParam);)

VOID amRotatePolygon(LPPOINT op, LPPOINT np, int num, int cX, int cY, int ang0);
วัตถุประสงค์ : หมุนตำแหน่ง Coordinate ในแถวลำดับของ POINT ที่ใช้ในการวาดรูปหลายเหลี่ยม
Return Value : ไม่ส่งค่าใดๆกลับมา
หมายเหตุ : op คือ แถวลำดับของ POINT ต้นฉบับ
np คือ แถวลำดับของ POINT ที่เป็นผลลัพธ์จากการหมุน
num คือ จำนวนสมาชิกในแถวลำดับของ POINT (ที่ต้องการหมุน)
cX, cY คือ จุดหมุน
ang0 คือ มุมที่จะหมุน (หน่วยเป็นองศา เช่น 10 องศา)

VOID amRotatePOINT(LPPOINT op, LPPOINT np, int num, int cX, int cY, double ang0);
วัตถุประสงค์ : ทำงานเหมือนฟังก์ชัน amRotatePolygon() แต่มุมที่ใช้มีหน่วยเป็นเรเดียน

VOID amGetCenter(LPPOINT p, int num, LPPOINT cp);
วัตถุประสงค์ : หาจุดกลางของรูปหลายเหลี่ยม
Return Value : ไม่ส่งค่าใดๆกลับมา แต่ผลลัพธ์จะเก็บใน cp
p คือ แถวลำดับของ POINT ที่เป็นรูปหลายเหลี่ยม
num คือ จำนวนสมาชิกในแถวลำดับ POINT
cp คือ ผลลัพธ์ที่หาได้ โดย cp.x, cp.y เป็นจุดกลางของรูปหลายเหลี่ยมใน p

ข้อความควบคุมการทำงานในระบบ

AM_FRAMEANIMATE

วัตถุประสงค์ : ตัวจักรทำภาพเคลื่อนไหวแบบวาดใหม่ amFrameAnimate() เป็นผู้ส่งให้กับสไปรต์ทุกตัว

หมายเหตุ : มีข้อมูลที่ส่งมาด้วย

wParam เก็บ hSprite ของสไปรต์ที่ได้รับข้อความนี้

lParam ที่ LOWORD(lParam) เก็บ hEnviron แฮนเดิลของสภาพแวดล้อม

ที่ HIWORD (lParam) เก็บ hDC แฮนเดิลของ Display Context ของ OffScreen

สามารถใช้ hDC ที่ส่งมานี้วาดภาพหรือข้อความลงไปได้เลย

SPR_MSG_CREATE

วัตถุประสงค์ : คลังโปรแกรม TAM จะส่งมาให้ฟังก์ชันประจำตัวสไปรต์เมื่อได้ให้กำเนิดสไปรต์เรียบร้อยแล้ว

หมายเหตุ : มีข้อมูลที่ส่งมาด้วย

wParam เก็บ hSprite แฮนเดิลของสไปรต์ตัวที่เกิดขึ้น

lParam เก็บ hEnviron แฮนเดิลของสภาพแวดล้อม

SPR_MSG_OUTFRAME

วัตถุประสงค์ : ส่งให้ฟังก์ชันประจำตัวสไปรต์ทราบว่าขณะนี้สไปรต์ออกนอกวินโดว์

หมายเหตุ : ข้อมูลที่ส่งมาด้วย

wParam เก็บ hSprite แฮนเดิลของสไปรต์

lParam ที่ LOWORD (lParam) เก็บ ตำแหน่งแนวนอน (Curx) ของสไปรต์ขณะนั้น

HIWORD (lParam) เก็บ ตำแหน่งแนวตั้ง (Cury) ของสไปรต์ขณะนั้น

SPR_MSG_SETPROP :

วัตถุประสงค์ : วินโดว์แม่ส่งให้สไปรต์ เพื่อกำหนดคุณสมบัติของสไปรต์

หมายเหตุ : ใครก็ได้สามารถส่งข้อความนี้ให้แก่สไปรต์โดยระบุแฮนเดิลของสไปรต์ให้ถูกต้อง

wParam เก็บ hSprite

lParam ที่ LOWORD (lParam) เก็บหมายเลขบอกคุณสมบัติที่ต้องการกำหนดค่า

HIWORD (lParam) เก็บค่าของคุณสมบัติที่ต้องการ

SPR_MSG_SETPOS

วัตถุประสงค์ : วินโดว์แม่ส่งให้สไปรต์ เพื่อกำหนดตำแหน่งแนวนอนและแนวตั้ง

หมายเหตุ : ใครก็ได้สามารถส่งข้อความนี้ให้แก่สไปรต์โดยระบุแฮนเดิลของสไปรต์ให้ถูกต้อง

wParam เก็บ hSprite

lParam ที่ LOWORD (lParam) เก็บตำแหน่งแนวนอน (New X Position)

HIWORD (lParam) เก็บตำแหน่งแนวตั้ง (New Y Position)

SPR_MSG_SETPLANE

วัตถุประสงค์ : วินโดว์แม่ส่งให้สไปรต์ เพื่อกำหนดระดับการเคลื่อนที่ของสไปรต์

หมายเหตุ : ใครก็ได้สามารถส่งข้อความนี้ให้แก่สไปรต์โดยระบุแฮนเดิลของสไปรต์ให้ถูกต้อง

wParam เก็บ hSprite

lParam ที่ LOWORD (lParam) เก็บแฮนเดิลของสภาพแวดล้อม hEnviron

HIWORD (lParam) เก็บระดับการเคลื่อนที่ของสไปรต์ (New Z Level)

SPR_MSG_MOVE

วัตถุประสงค์ : วินโดว์แม่ส่งให้สไปรต์ เพื่อสั่งให้สไปรต์เคลื่อนที่ตามแนวนอนและแนวตั้งไปอีกกี่จุด ภาพ และสั่งให้มีการเปลี่ยนภาพได้ด้วย

หมายเหตุ : ใครก็ได้สามารถส่งข้อความนี้ให้แก่สไปรต์โดยระบุแฮนเดิลของสไปรต์ให้ถูกต้อง

wParam เก็บวิธีการเปลี่ยนภาพเช่น FRM_FORWARD, FRM_LOOP เป็นต้น

lParam ที่ LOWORD (lParam) เก็บระยะการเคลื่อนที่ในแนวนอน (dX)

HIWORD (lParam) เก็บระยะการเคลื่อนที่ในแนวตั้ง (dY)

SPR_MSG_SETFRAME

วัตถุประสงค์ : วินโดว์แม่ส่งให้สไปรต์ เพื่อทำการเปลี่ยนภาพ

หมายเหตุ : ใครก็ได้สามารถส่งข้อความนี้ให้แก่สไปรต์โดยระบุแฮนเดิลของสไปรต์ให้ถูกต้อง

wParam เก็บ hSprite

lParam เก็บวิธีการเปลี่ยนภาพ

SPR_MSG_LEFTFRAME , SPR_MSG_RIGHTFRAME ,

SPR_MSG_TOPFRAME , SPR_MSG_BOTTOMFRAME

วัตถุประสงค์ : ระบบส่งให้สไปรต์เพื่อให้ทราบว่าขณะนี้สไปรต์มีการชนที่ขอบวินโดว์ด้านใด

หมายเหตุ :

wParam เก็บ hSprite

lParam ที่ LOWORD (lParam) เก็บตำแหน่งแนวนอนของสไปรต์ที่ชนวินโดว์ (Cur X)

HIWORD (lParam) เก็บตำแหน่งแนวตั้งของสไปรต์ที่ชนวินโดว์ (Cur Y)

SPR_MSG_NOCOLLISION

วัตถุประสงค์ : ระบบส่งให้สไปรต์เมื่อมีการร้องของให้ตรวจสอบการชนกัน แต่ไม่มีการชนกับใคร

หมายเหตุ : ถ้าไม่มีการตรวจสอบการชนกันจะไม่ส่งข้อความนี้มาให้

wParam เก็บ hSprite

lParam ไม่เก็บค่าใดๆ

SPR_MSG_COLLISION

วัตถุประสงค์ : ระบบส่งให้สไปรต์เมื่อมีการร้องขอให้ตรวจสอบการชนกันและเกิดการชนกันขึ้น

หมายเหตุ : ถ้าไม่มีการตรวจสอบการชนกันจะไม่ส่งข้อความนี้มาให้

wParam เก็บ จำนวนสไปรต์ที่มีการชนกัน

lParam เก็บแฮนเดิลของหน่วยความจำที่เป็นแถวลำดับของสไปรต์ที่มีการชนกัน
ดังนั้นการตรวจสอบว่าชนกับใครบ้างให้มาดูที่นี่

SPR_DEFAULT

วัตถุประสงค์ : ใช้ในการให้กำเนิดสไปรต์ในฟังก์ชัน amCreateSprite() โดยบอกสถานภาพเริ่มต้นเป็นค่าปริยาย คือ ให้มีการปรากฏตัวของสไปรต์ (active I update)

การกำหนดชนิดของข้อมูลและโครงสร้างต่างๆ

```
typedef  HANDLE    HSPRITE;          // กำหนดแฮนเดิลของสไปรต์
typedef  HANDLE    HBMTAB;           // กำหนดแฮนเดิลของคลังภาพ
typedef  HANDLE    HFRAMELIST;       // กำหนดแฮนเดิลของแถวลำดับของภาพในสไปรต์
typedef  HANDLE    HSPRTAB;          // กำหนดแฮนเดิลของคลังสไปรต์
typedef  HANDLE    HSEQUENCE;        // กำหนดแฮนเดิลของลำดับภาพ
typedef  HANDLE    HENVIRON;         // กำหนดแฮนเดิลของสภาพแวดล้อม
typedef  HANDLE    HSPRPATH;         // กำหนดแฮนเดิลของเส้นทางการเคลื่อนที่ของสไปรต์
```

```
typedef struct   tagBMTAB {              // กำหนดโครงสร้างของคลังภาพ
  HBITMAP   hPic;
  HBITMAP   hMask;
  LPSTR     Name;
  int       bmWidth;
  int       bmHeight;
} BMTAB;

typedef struct   tagCOLLTAB {            // กำหนดโครงสร้างของพื้นที่ตรวจสอบการชนกัน
  LPRECT    lpRect;
  char      nRect;
} COLLTAB;

typedef struct tagCOLLISION {            // กำหนดโครงสร้างของแถวลำดับของพื้นที่การชนกัน
  LPCOLLTAB   lpCollFrameTab;
  char        nCollFrame;
  char        CurCollFrame;
} COLLISION;

typedef struct  tagMove {                // กำหนดโครงสร้างข้อมูลสำคัญที่ใช้ในการเคลื่อนที่ของสไปรต์
      int    dX,dY,ErrorTerm,Counter;
      char   InitFlag;
}MOVE;

typedef struct  tagSprPath {             // กำหนดโครงสร้างของเส้นทางการเคลื่อนที่ของสไปรต์
      int        CurPoint;
      int        MaxPoint;
      LPPOINT    lpPoint;
} SPRPATH;
```

```
typedef  struct  tagSprite {              // กำหนดโครงสร้างของสไปรต์
 UINT          sId;
 int           CurX,  CurY,  CurZ;
 int           LastX, LastY, LastZ;
 MOVE          MoveProp;
 HSPRPATH      hPath;
 int           MaxFrame, CurFrame;
 HBMTAB        hBmTab;
 HFRAMELIST    hFrameList;
 HBITMAP       bmBkGd;
 int           bmWidth,   bmHeight;
 int           LastWidth, LastHeight;
 char          MoveDir;
 char          xDir, yDir;
 char          bmDir;
 char          Size;
 int           Priority;
 int           CurPriority;
 char          Status;
 int           Timer;
 int           LifeTime;
 HWND          hDispWnd;
 HWND          hWnd;
 WNDPROC       SprProc;
 char          CollStatus;
 COLLISION     Collision;
} SPRITE;

typedef struct tagCOORD {          // กำหนดโครงสร้างของตำแหน่งของสไปรต์
 int      x,y,z;
} COORD;
```

```
typedef struct tagSPRTAB {        // กำหนดโครงสร้างของคลังสไปรต์
  HSPRITE       hSprite;
  PTSPRITE      lpSprite;
} SPRTAB;

typedef struct tagSequence {      // กำหนดโครงสร้างของลำดับภาพ
  HBMTAB        hBmTab;
  HFRAMELIST    hFrameList;
  int           nFrame;
} SEQUENCE;
typedef struct tagFrameList {     // กำหนดแถวลำดับของภาพที่สไปรต์ใช้ภายใน
  int    BmTabNumber;
  char  Status;
  char  DeltaWidth;
  char  DeltaHeight;
} FRAMELIST;

typedef struct tagEnvironment{    // กำหนดโครงสร้างของสภาพแวดล้อม
 HWND           hDispWnd;
 HBITMAP        hBkGd;
 int            left,top,right,bottom;
 int            bmWidth, bmHeight;
 HBITMAP        image;
 int            imWidth, imHeight;
 char           imageFile;
 char           BkSize;
 UINT           nRunID;
 HSPRTAB        hSprTab;
 int            SprCounter;
 char           ReOrderFlag;
} ENVIRON;
```

```
typedef  struct  tagPICMASK {     // กำหนดโครงสร้างรายชื่อภาพที่ต้องการสร้างคลังภาพ
  LPSTR          PicName;
  LPSTR          MaskName;
} PICMASK;

typedef struct tagFRAME {         // กำหนดโครงสร้างแถวลำดับรายการเลือกภาพจากคลังภาพ
  LPSTR          Frame;
  char           Status;
} FRAME;

typedef PICMASK          FAR  *   LPPICMASK;
typedef PICMASK          NEAR *   NPPICMASK;
typedef FRAME            FAR  *   LPFRAME;
typedef FRAME            NEAR *   NPFRAME;
typedef int              FAR  *   LPINT;
typedef int              NEAR *   NPINT;
typedef ENVIRON          FAR  *   LPENVIRON;
typedef ENVIRON          NEAR *   NPENVIRON;
typedef BMTAB            FAR  *   LPBMTAB;
typedef BMTAB            NEAR *   NPBMTAB;
typedef SPRTAB           FAR  *   LPSPRTAB;
typedef SPRTAB           NEAR *   NPSPRTAB;
typedef COLLTAB          NEAR *   NPCOLLTAB;
typedef COLLTAB          FAR  *   LPCOLLTAB;
typedef COLLISION        NEAR *   NPCOLLISION;
typedef COLLISION        FAR  *   LPCOLLISION;
typedef SPRPATH          FAR  *   LPSPRPATH;
typedef SPRPATH          NEAR *   NPSPRPATH;
typedef SPRITE           FAR  *   LPSPRITE;
typedef SPRITE           NEAR *   NPSPRITE;
typedef COORD            NEAR *   NPCOORD;
```

```
typedef COORD          FAR  *  LPCOORD;
typedef SEQUENCE       NEAR *  NPSEQUENCE;
typedef SEQUENCE       FAR  *  LPSEQUENCE;
typedef FRAMELIST      NEAR *  NPFRAMELIST;
typedef FRAMELIST      FAR  *  LPFRAMELIST;
```

ประวัติผู้เขียน

นายสมพงษ์ ปาลกูล เกิดวันที่ 5 กันยายน พ.ศ. 2508 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีครุศาสตรบัณฑิต วิชาเอกฟิสิกส์ สาขามัธยมศึกษา คณะครุศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี 2530 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2534