

Drag and Drop

Introduction

A common requirement in a GUI environment and in many games is the ability to use the mouse to drag items about the screen. In this section we'll see how this can be done in DarkBASIC Pro.

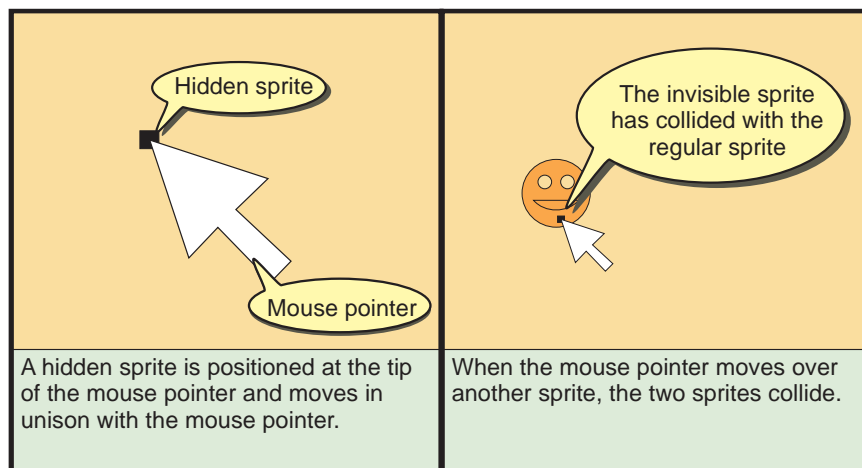
The examples shown below use sprites, but there is no reason why other items such as images and pictures cannot be dragged using the techniques given.

Detecting When the Mouse is Over a Sprite

As we saw in Hands On DarkBASIC Pro volume 1, a simple way of detecting when the mouse is positioned over a sprite is to create an invisible one-pixel sprite which is positioned at the mouse pointer tip and then check for the invisible sprite hitting the other sprite (see FIG-Sup08).

Fig-Sup08

Using an Invisible Sprite to Detect the Mouse-Over Event



We start by creating a 1x1 black pixel image (*black.bmp*) and loading this into an image:

```
#CONSTANT BlackImg 1
LOAD IMAGE "black.bmp",BlackImg
```

The image is then linked to the invisible sprite:

```
#CONSTANT InvisibleSpr 1
SPRITE InvisibleSpr,0,0,BlackImg
```

To make sure that the sprite is always positioned at the mouse tip, the line

```
SPRITE InvisibleSpr, MOUSEX(), MOUSEY(), BlackImg
```

must be repeatedly executed.

These lines are brought together in the example program given in LISTING-Sup008.

LISTING-Sup008

Linking a Sprite to the Mouse Pointer

```
REM *** Constants ***
REM *** Images ***
#CONSTANT BlackImg      1
REM *** Sprites ***
#CONSTANT InvisibleSpr  1

REM *** Load image ***
LOAD IMAGE "black.bmp",BlackImg

REM *** Load sprite ***
SPRITE InvisibleSpr,0,0,BlackImg

REM *** Main loop ***
DO
  SPRITE InvisibleSpr,MOUSEX(),MOUSEY(),BlackImg
LOOP

REM *** End program ***
END
```

Activity Sup019

Type in the program given above.

To check that the code actually works, change the image used from *black.bmp* to *red.bmp* which is both visible and larger.

Once you have seen that the technique works, return to using the *black.bmp* image.

We can check for the mouse moving over a regular sprite by adding a new sprite to our program:

```
#CONSTANT SmileyImg      2
#CONSTANT SmileySpr      2
LOAD IMAGE "smiley.bmp",SmileyImg
SPRITE SmileySpr,400,300,SmileyImg
```

and changing the main loop to check for a collision between the sprites:

```
DO
  SPRITE InvisibleSpr,MOUSEX(),MOUSEY(),BlackImg
  IF SPRITE HIT(InvisibleSpr, SmileySpr)
    PRINT "HIT"
  ENDIF
LOOP
```

Activity Sup020

Modify your existing program to display the word "HIT" when the mouse pointer moves over the smiley sprite.

Dragging the Sprite

Moving the regular sprite is little different from moving the invisible sprite. The main difference being that the regular sprite should only move when the mouse pointer is over the sprite and the mouse button is being pressed.

A sprite's origin is normally at its top-left corner; this can cause problems when dragging a sprite, so it's best if we move the origin to the centre of the sprite with the line:

```
OFFSET SPRITE SmileySpr, SPRITE WIDTH(SmileySpr)/2,
↳SPRITE HEIGHT(SmileySpr)/2
```

Now all we need to do is change the IF statement in the main loop to read:

```
IF SPRITE HIT(InvisibleSpr, SmileySpr) AND (MOUSECLICK() = 1)
  SPRITE SmileySpr, MOUSEX(), MOUSEY(),SmileySpr
ENDIF
```

Activity Sup021

Make the changes described above and run the program to check that the sprite can now be dragged.

The Jump Problem

Although the code works, there is a slight problem when we first click on the sprite to be dragged.

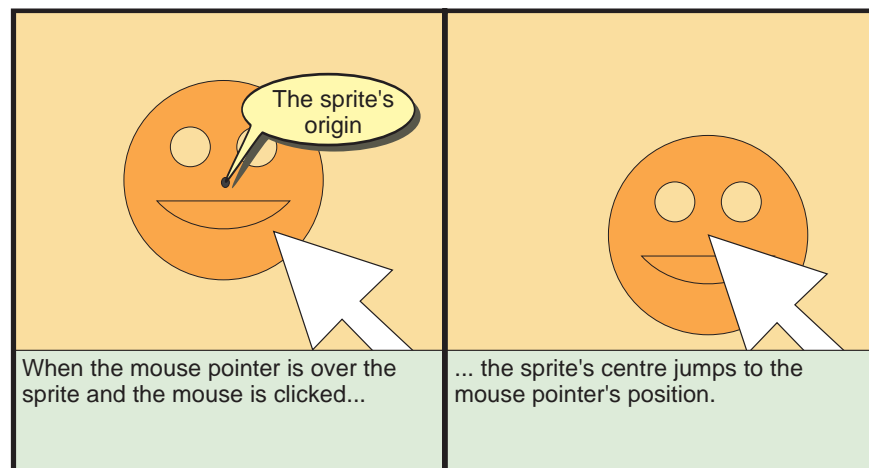
Activity Sup022

Place the mouse pointer in the lower right area of the smiley sprite and click the mouse button without any further movement of the mouse pointer.

What happens to the sprite when the mouse click occurs?

What happens when we first click the mouse within the sprite's area is shown in FIG-Sup09.

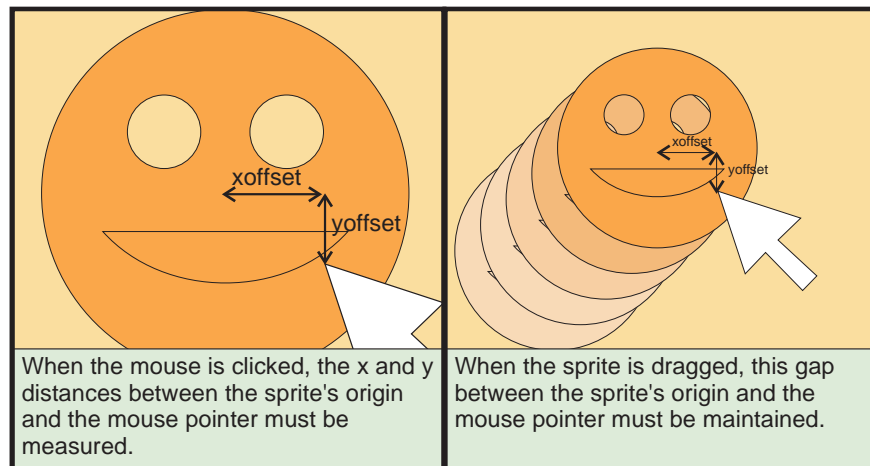
FIG-Sup09
The Sprite Jump



We don't want this jump to happen; the sprite should only move once the mouse pointer itself is moved. We can solve this problem by discovering how far the mouse pointer is from the centre of the sprite when the mouse is first clicked, and then maintain this difference as the sprite is dragged (see FIG-Sup10).

FIG-Sup10

Maintaining the Offset



The code required to record the x and y offsets from the sprite origin to the mouse pointer is shown below:

```
IF SPRITE HIT(InvisibleSpr, SmileySpr) AND (MOUSECLICK()=1)
  xoffset = SPRITE X(SmileySpr) - MOUSEX()
  yoffset = SPRITE Y(SmileySpr) - MOUSEY()
  WHILE MOUSECLICK() = 1
    SPRITE SmileySpr, MOUSEX()+xoffset, MOUSEY()+yoffset,
    ↪SmileySpr
  ENDWHILE
ENDIF
```

Activity Sup023

Modify your code to incorporate these changes to the main loop and check that the program worked correctly.

Handling Several Sprites

It is quite likely that your screen will contain several sprites, so when we perform dragging we first need to detect which sprite has been selected with a line such as:

```
spriteselectd = SPRITE HIT(InvisibleSpr, 0)
```

The variable *spriteselectd* is then used in all subsequent sprite-related statements:

```
IF (spriteselectd <> 0) AND (MOUSECLICK()=1)
  xoffset = SPRITE X(spriteselectd) - MOUSEX()
  yoffset = SPRITE Y(spriteselectd) - MOUSEY()
  WHILE MOUSECLICK() = 1
    SPRITE spriteselectd, MOUSEX()+xoffset, MOUSEY()+
    ↪yoffset,spriteselectd
  ENDWHILE
ENDIF
```

This ensures that the correct sprite is being dragged.

Activity Sup024

Modify your last program so that it contains three sprites (*SmileySpr*, *BoltSpr*, and *StarSpr*) which use the images *smiley.bmp*, *bolt.bmp* and *star.bmp*.

Change the main loop so that any of the three sprites may be dragged.

Solutions

Activity Sup019

No solution required.

Activity Sup020

```
REM *** Constants ***
REM *** Images ***
#CONSTANT BlackImg      1
#CONSTANT SmileyImg     2
REM *** Sprites ***
#CONSTANT InvisibleSpr  1
#CONSTANT SmileySpr     2
REM *** Load image ***
LOAD IMAGE "black.bmp",BlackImg
LOAD IMAGE "smiley.bmp",SmileyImg
REM *** Load sprite ***
SPRITE InvisibleSpr,0,0,BlackImg
SPRITE SmileySpr,400,300,SmileyImg
REM *** Main loop ***
DO
  SPRITE InvisibleSpr,MOUSEX(),MOUSEY(),
  ↵BlackImg
  IF SPRITE HIT(InvisibleSpr, SmileySpr)
    PRINT "HIT"
  ENDIF
LOOP
REM *** End program ***
END
```

Activity Sup021

```
REM *** Constants ***
REM *** Images ***
#CONSTANT BlackImg      1
#CONSTANT SmileyImg     2
REM *** Sprites ***
#CONSTANT InvisibleSpr  1
#CONSTANT SmileySpr     2
REM *** Load image ***
LOAD IMAGE "black.bmp",BlackImg
LOAD IMAGE "smiley.bmp",SmileyImg
REM *** Load sprite ***
SPRITE InvisibleSpr,0,0,BlackImg
SPRITE SmileySpr,400,300,SmileyImg
REM *** Move sprite's origin to centre ***
OFFSET SPRITE SmileySpr,
↵SPRITE WIDTH(SmileySpr)/2,
↵SPRITE HEIGHT(SmileySpr)/2
REM *** Main loop ***
DO
  SPRITE InvisibleSpr,MOUSEX(),MOUSEY(),
  ↵BlackImg
  IF (SPRITE HIT(InvisibleSpr, SmileySpr))
    ↵AND (MOUSECLICK() = 1)
      SPRITE SmileySpr, MOUSEX(), MOUSEY(),
      ↵SmileySpr
    ENDIF
  LOOP
REM *** End program ***
END
```

Activity Sup022

The sprite jumps to a new position such that its origin (centre) is now at the tip of the mouse pointer.

Activity Sup023

```
REM *** Constants ***
REM *** Images ***
#CONSTANT BlackImg      1
#CONSTANT SmileyImg     2
REM *** Sprites ***
#CONSTANT InvisibleSpr  1
#CONSTANT SmileySpr     2
REM *** Load images ***
LOAD IMAGE "black.bmp",BlackImg
LOAD IMAGE "smiley.bmp",SmileyImg
REM *** Load mouse tip sprite ***
SPRITE InvisibleSpr,0,0,BlackImg
REM *** Load regular sprite ***
SPRITE SmileySpr,400,300,SmileyImg
OFFSET SPRITE SmileySpr,
↵SPRITE WIDTH(SmileySpr)/2,
↵SPRITE HEIGHT(SmileySpr)/2
REM *** Main loop ***
DO
  SPRITE InvisibleSpr,MOUSEX(),MOUSEY(),
  ↵BlackImg
  IF SPRITE HIT(InvisibleSpr,SmileySpr)
    ↵AND (MOUSECLICK()=1)
      xoffset = SPRITE X(SmileySpr)-MOUSEX()
      yoffset = SPRITE Y(SmileySpr)-MOUSEY()
      WHILE MOUSECLICK() = 1
        SPRITE SmileySpr, MOUSEX()+xoffset,
        ↵MOUSEY()+yoffset,SmileySpr
      ENDWHILE
    ENDIF
  LOOP
REM *** End program ***
END
```

Activity Sup024

```
REM *** Constants ***
REM *** Images ***
#CONSTANT BlackImg      1
#CONSTANT SmileyImg     2
#CONSTANT BoltImg       3
#CONSTANT StarImg       4
REM *** Sprites ***
#CONSTANT InvisibleSpr  1
#CONSTANT SmileySpr     2
#CONSTANT BoltSpr       3
#CONSTANT StarSpr       4
REM *** Load images ***
LOAD IMAGE "black.bmp",BlackImg
LOAD IMAGE "smiley.bmp",SmileyImg
LOAD IMAGE "bolt.bmp",BoltImg
LOAD IMAGE "star.bmp",StarImg
REM *** Load mouse tip sprite ***
SPRITE InvisibleSpr,0,0,BlackImg
REM *** Load regular sprites ***
SPRITE SmileySpr,200,300,SmileyImg
OFFSET SPRITE SmileySpr, SPRITE
↵WIDTH(SmileySpr)/2, SPRITE
↵HEIGHT(SmileySpr)/2
SPRITE BoltSpr,300,300,BoltImg
OFFSET SPRITE BoltSpr,
↵SPRITE WIDTH(BoltSpr)/2,
↵SPRITE HEIGHT(BoltSpr)/2
SPRITE StarSpr,400,300,StarImg
OFFSET SPRITE StarSpr,
↵SPRITE WIDTH(StarSpr)/2,
↵SPRITE HEIGHT(StarSpr)/2
REM *** Main loop ***
DO
  SPRITE InvisibleSpr,MOUSEX(),MOUSEY(),
  ↵BlackImg
```

```
spriteSelected = SPRITE HIT(InvisibleSpr, 0)
IF (spriteSelected<> 0) AND (MOUSECLICK()=1)
  xoffset = SPRITE X(spriteSelected)-MOUSEX()
  yoffset = SPRITE Y(spriteSelected)-MOUSEY()
  WHILE MOUSECLICK() = 1
    SPRITE spriteSelected, MOUSEX()+xoffset,
    MOUSEY()+yoffset,spriteSelected
  ENDWHILE
ENDIF
LOOP
REM *** End program ***
END
```