Segments and Flushing

-
Alyce

Segments and Flushing | Graphics and Memory | CLS | FLUSH | FLUSH SEGMENT NAME | SEGMENT | SPRITES | DISCARD | REDRAW | DELSEGMENT | DEMO

Graphics and Memory

Liberty BASIC graphics consume memory. Each graphics command is written to memory to something called a metafile. If many graphics are drawn, memory resources may become low. This can make a program and a system perform sluggishly. It can even cause a program to crash. You can avoid this problem with proper graphics memory management. There are three commands that release memory consumed by graphics: CLS, DISCARD, and DELSEGMENT

CLS

The CLS (CLearScreen) command clears the graphics area and releases all memory consumed by graphics. This will work fine if there is no need to remember previous drawing routines, and if the flickering associated with clearing the screen is acceptable in a program.

FLUSH

The FLUSH command may be a bit confusing. In other areas of life, the word flush may mean to remove things, or to clean things out. We flush the old antifreeze from our radiators so that we can fill them with new antifreeze, just as one example. In Liberty BASIC, FLUSH has a unique meaning. It means "make graphics persist". Graphics drawn on a graphics window or graphicbox will disappear if the window is obscured or minimized. If we issue a FLUSH command, the graphics will persist so that they will still be visible when the window is again made visible.

syntax: print #handle, "flush"

Each time the FLUSH command is issued, a "drawing segment" is created. All drawing operations that are accomplished in between FLUSHes (or from the beginning to the first FLUSH command) are part of a drawing segment. Each of these segments has an ID number. A FLUSH command closes the current drawing segment and increments the ID by 1.

FLUSH SEGMENT NAME

syntax: print #handle, "flush segmentName"

This command ensures that drawn graphics persist. It assigns a **name** to the flushed segment. This assigned name can be used by DELSEGMENT and REDRAW to manipulate the segment.

SEGMENT

You can retrieve the segment number (segment ID) of the current drawing segment with the SEGMENT command. It looks like this:

syntax: print #handle, "segment variableName"

as it might look in a program:

```
print #1, "segment segID"
notice "Drawing segment is ";segID
```

After this command is issued, the segment ID number will be contained in the receiver variable. This is the ID for the segment that is currently open. The first segment begins with the first graphic command and ends with the first FLUSH command. All segments after that are contained between FLUSH commands. To get the ID of a segment, use the SEGMENT command right before the FLUSH command. Segment IDs are used to DISCARD or REDRAW individual segments. The receiver variable can be named as desired, and it is advantageous to use discriptive names for it. For instance, if a circle has been drawn, you might call the segment "circleID".

The segment ID for each segment doesn't change, regardless of any additions or deletions of other segments.

SPRITES

Segments and flushing do not apply to sprite graphics placed on the screen with DRAWSPRITES. If sprite graphics are flushed by using GETBMP, DRAWBMP, FLUSH then this sequence does constitute a drawing segment.

DISCARD

This causes all items drawn since the last flush to be removed from memory immediately. Discard does not force an immediate redraw, so the items that have been discarded will still be displayed until a REDRAW (see REDRAW below).

syntax: print #handle, "discard"

as it might appear in a program:

```
print #1, "discard"
```

REDRAW

This command has multiple forms. The first form will cause the window to redraw all flushed drawn segments. Any deleted segments will not be redrawn (see DELSEGMENT below). Any items drawn since the last flush will not be redrawn either, and will be lost.

syntax: print #handle, "redraw"

as it might appear in a program:

```
print #1, "discard; redraw"
```

The second form of the command allows us to redraw a specific drawing segment.

Syntax: syntax: print #handle, "redraw "; idNum

Let's imagine that we've filled the screen with blue and drawn a pink circle and a green box in the first drawing segment, which we close with a FLUSH command. We issue other drawing commands later in the program, but later still, we went to display that first segment again. Knowing that this is the first segment, we can issue a REDRAW command like this:

```
print #1, "discard; redraw"
```

Any segment may be redrawn in this fashion, as long as it hasn't been deleted with DELSEGMENT (see below), or by the CLS command. We can retrieve the ID of a drawing segment and store it in a variable for use later with the SEGMENT command, which is explained above. If we have placed a particular segment ID into a variable called "myID", then we redraw that segment like this:

```
print #1, "redraw 1"
```

The third form of REDRAW causes a named segment to be redrawn. The name was assigned by the programmer when the FLUSH command was issued.

syntax: print #handle, "redraw "; segmentName

It might look like this in a program:

```
#1 "flush mySegment"
#1 "fill green"
#1 "redraw mySegment"
```

DELSEGMENT

This causes the drawn segment identified to be removed from the window's list of drawn items. The memory that was consumed by the drawn segment is reclaimed by the operating system. When the window is redrawn, the deleted segment will not be included in the redraw. It will not be possible to redraw this segment with the REDRAW command.

The first form deletes a segment according to ID number.

syntax: print #handle, "delsegment n"

As it might appear in a program:

```
'with hard-coded ID number:
print #1, "delsegment 2"

'or with ID number contained in a variable:
print #1, "delsegment ";myID
```

The second form of DELSEGMENT deletes a segment according to the name given it by the programmer when the FLUSH command is issued.

syntax: print #handle, "delsegment segmentName"

DEMO

The following small program illustrates how to flush graphics, how to obtain segment IDs, how to discard grahics, and how to redraw individual segments.

```
nomainwin
button #1.redraw, "Redraw",[newDraw],UL,10,10, 120,24
open "Flush and Redraw Demo" for graphics_nf_nsb as #1
#1 "trapclose [quit]"

#1 "down; fill yellow; size 5"
#1 "color darkgreen;backcolor green"
#1 "place 10 70;boxfilled 200 200"
#1 "place 30 120;\First Segment"
#1 "segment boxID"
#1 "flush"  'end first segment

#1 "fill pink;color yellow"
#1 "backcolor darkcyan"
#1 "place 150 150; circlefilled 100"
#1 "place 100 140;\Second Segment"
#1 "segment circleID"
#1 "flush"  'end second segment

#1 "fill blue; color lightgray"
#1 "backcolor darkred;place 150 150"
#1 "ellipsefilled 250 200"
#1 "place 50 140;\Third Segment, not flushed"
#1 "discard"    'discard these commands from memory

wait

[newDraw]
if currentID=circleID then
    #1 "redraw ";boxID
    currentID=boxID
else
    #1 "redraw ";circleID
    currentID=circleID
end if
wait

[quit]
```

```
close #1:end
```

Segments and Flushing | Graphics and Memory | CLS | FLUSH | FLUSH SEGMENT NAME | SEGMENT | SPRITES | DISCARD | REDRAW | DELSEGMENT | DEMO