Icons and Blocks

This Chapter deals with the practical handling of rectangular units of graphic images.

Background screen graphics

It is common for modern arcade games to feature hundreds of different background screens, over which the
animated action takes place. Similarly, practical programs like kitchen-planners may need to display scores
of varied settings.

Backgrounds can be constructed from a set of simple graphic blocks, to be arranged
and re-arranged as you wish, in varied patterns Each background screen can now be stored as a simple list of
component blocks. These blocks are sometimes known as "tiles", and AOZ provides two sets of alternative tiles: Icons, which are held in their own memory bank, and Blocks, which are held as temporary data.

Icons

An Icon is an individual image, specifically designed to act as a component of a background screen picture. All Icons are stored in their own AOZ memory bank, which is bank 2, and this Icon Bank will be saved along with your program listing automatically.

Once an Icon is drawn it has a fixed location and cannot be moved to another part of the screen.
TODO - "Update Amiga conventions below to AOZ"
Icons are displayed using the Amiga's Blitter chip, which is also responsible for the display of Robs. However, because Icons are essentially static Objects, they are normally drawn in replace mode. This means that any existing graphics at the relevant screen location will be completely erased by the Icon.

Here is a complete list of the Icon commands.

GET ICON

instruction: create an Icon

```
Get Icon Icon number,x1,y1 To x2,y2
Get Icon screen number,lcon number,x1,y1 To x2,y2
```

The GET ICON command grabs an image from the screen and loads it into an Icon.

Specify the Icon number, and then give the coordinates of the rectangle that is to be grabbed, from the top left-hand corner to the bottom righthand corner. If the Icon whose number you specify does not already exist, it will be created in Bank 2.

If the memory bank has not been reserved, this will also be done automatically.
An optional screen number can also be given, immediately after the GET ICON instruction, and this will select the screen to be used as the source of the Icon's image. If this screen number is omitted, the image is taken from the current screen.

GET ICON PALETTE

instruction: Load Icon colours into current screen

```
Get Icon Palette
```

This instruction is usually employed to initialise a screen, after Icons have been loaded from disc.

GET ICON PALETTE grabs the colours of the Icon images stored in Bank 2, and loads them in to the current screen.

PASTE ICON

instruction: Draw an Icon

```
Paste Icon x,y,number
```

Use the PASTE ICON command to draw the specified Icon number already stored in Bank 2, on screen. The screen position is defined by graphic coordinates, and can be anywhere you like. Icon images will be clipped in the normal way, if they exceed the standard limitations. Here is a simple example:

```
Flash Off : Load Iff "AMOSPro_Examples:Iff/logo.iff"
Z=0
For A=0 To 304 Step 16
    Inc Z
    Get Block Z,A,1,16,199
Next A
Cls 0
For A=0 To 304 Step 16
    Put Block Z,A,0
    Dec Z
    Wait Vbl
Next A
```

If the DOUBLE BUFFER system in engaged, a copy of the Icon will be drawn into both the logical and physical screens, and because this takes a little time, you are advised to add a call to AUTOBACK 0 before drawing Icons on screen. This restricts the Icon to the current logical screen, and then the entire background may be copied to the physical screen, using SCREEN COPY, which is a much faster process.

DEL ICON

instruction: Delete Icons

```
Del Icon number
Del Icon first number To last number
```

DEL ICON erases the Icon whose number is specified from Bank 2. A second Icon number may also be given, in which case, all Icons from the first number to the second number will be deleted. When the final Icon in the bank has been deleted, the whole bank will be removed from memory.

INS ICON

instruction: Insert a blank Icon image into the Icon bank

```
Ins Icon number
Ins Icon first To last
```

The INS ICON instruction operates in exactly the same way as INS BOB, which is explained in Chapter XX.X (7.2).

MAKE ICON MASK

instruction: Set colour zero to transparent

```
Make Icon Mask
Make Icon Mask number
```

Normally, any Icons that are drawn on screen completely replace the existing background image, and the Icon appears in a rectangular box filled with colour zero. If you prefer to overlay Icons on top of the current graphics, a mask must be created. This is achieved by the MAKE ICON MASK command, and sets colour zero to transparent.

All Icons in Bank 2 will be affected by this instruction, unless an optional Icon number is given, in which case only that Icon will be masked.

NO ICON MASK

instruction: Remove colour zero mask from Icon

```
No Icon Mask number
```

This command performs exactly the same task as the NO MASK instruction, explained in Chapter 7.2, except that it is used with Icons instead of Bobs.

Screen Blocks

Unlike Icons, graphic Blocks are not saved along with your programs, and the following BLOCK Instructions are used to hold and manipulate temporary graphics data. Blocks are extremely useful for setting up items such as dialogue boxes, by saving background pictures before new graphics are displayed. They can be used to create "tiles" for all sorts of entertainment programs, such as visual puzzles, as well as practical programs like identi-kits and architectural planners.

GET BLOCK

instruction: Grab a screen Block into memory

```
Get Block number,x,y,width,height
Get Block number,x,y,width,height,mask
```

The GET BLOCK command is used to grab a rectangular area from the graphics on the current screen. First specify a Block number from 1 up to 65535, then set the coordinates of the top left-hand corner of the rectangle to be grabbed, followed by the number of pixels making up the width and height of the Block.

An optional mask code can be added after these parameters. If this code is set to zero, the Block will destroy and replace any graphics that used to occupy its position on screen. If the mask code is set to 1, the block is given a
background mask, and colour zero becomes transparent.

PUT BLOCK

instruction: Copy Block onto screen

```
Put Block number
Put Block number,x,y
Put Block number,x,y,bit-planes
Put Block number,x,y,bit-planes,blitter mode
```

To re-draw a Block at its original coordinates on the current screen, simply add the Block's identification number after the PUT BLOCK command.

If you want to draw the Block at a new position, then add the new x,y-coordinates for the left-hand corner, after the Block number.

On the Amiga, the screen is divided into segments known as "bit-planes", and Blocks are normally displayed using all the available screen bit-planes, which is a bit-pattern of %111111. Resetting these bit-planes can create numerous special effects, and various settings are dealt with at the beginning of Chapter XX.X (6.2).

AOZ emulates the Amiga bitplane in emulation mode.

DEL BLOCK

instruction: delete a screen Block

```
Del Block
Del Block number
```

To delete all new screen Blocks, the DEL BLOCK command is used. The memory these Blocks used is returned to the main program automatically. If you only want to get rid of a single Block, follow the command with that Block's identification number.

HREV BLOCK

instruction: Flip a Block horizontally

```
Hrev Block number
```

This command reverses any numbered Block, by flipping it over its own horizontal axis.

VREV BLOCK

instruction: Flip a Block vertically

```
Vrev Block number
```

Similarly, VREV BLOCK is used to flip a block over its own vertical axis.

Compacted blocks

If you need reminding about the screen compaction memory-saving techniques, please refer to SPACK and PACK, which are fully explained at the end of Chapter 6.2. The compaction system used for the following commands is designed for speed as opposed to efficiency. They save less memory than SPACK and PACK, but they are a lot
faster!

GET CBLOCK

instruction: Save and compact a screen Block

```
Get Cblock number,x,y,width,height
```

The GET CBLOCK command is used to save and compact a rectangular area of graphics from the screen. These Blocks are often used to grab the area underneath dialogue boxes, so that after the dialogue has been completed, the screen can be rapidly restored to its original state.

Specify the Block number from 1 to 65535, followed by the x,y-coordinates of its top left-hand corner. Then define the Block by giving its width and height, in pixels. Note that the x- coordinate, and the width of the Block will be
rounded to the nearest multiple of eight pixels.

PUT CBLOCK

instruction: Display a compacted Block

```
Put Cblock number
Put Cblock number,x,y
```

This command places the Block whose number is specified at its original screen coordinates. Optional target coordinates can be added, in which case the Block will be unpacked and then drawn at the new position. Any new x-coordinate will also be rounded to the nearest 8-pixel boundary.

DEL CBLOCK

instruction: Delete compacted Blocks

```
Del Cblock
Del Cblock number
```

The DEL CBLOCK instruction erases all compacted Blocks from memory, unless an individual Block number is specified, in which case only that Block will be erased.