PIX-ART

Frank Dietrich
Zsuzsanna Molnar
Electronic Visualization Laboratory
University of Illinois at Chicago Circle
Departments of Art & Information Engineering
Box 4348
Chicago, Il 60680

INTRODUCTION

Approaching a computer graphics system like the Bally Arcade, the user should understand the basic contradiction inherent in this machine--while it offers rich options conceptually, it also produces "poor graphics", at least compared to state-of the art computer graphics. So you shouldn't expect this small computer to generate close to real-life images with smooth shading, hidden surfaces, simulated highlighting and the like. Instead, you will have to "buy the pixels", those blocky constituents of every raster system which most people try to hide. Once you become acquainted with the needlepoint effect all the Bally images show, you can go ahead appreciating the smart design of a personal computer system which is a major step towards a computer-literate society rather than a computer controlled one.

The manufacturer of the Arcade, Bally Corporation, gained its experience building interactive and easy to use slot and pinball machines played by millions of people, regardless of their age or previous knowledge. Continuing their orientation towards mass entertainment, Bally started designing a microprocessor controlled home video game in 1975. The result is the Bally Arcade, whose extreme low cost and TV orientation has placed computers in over 100,000 homes.

The system includes the computer, a calculator keypad, four hand controls, an interface for storing programs on an audio-cassette recorder, a modulator to hook up to any TV and a ROM with the Bally Basic language. Its price of $350 makes it an affordable personal and educational tool. Like a portable typewriter, the Arcade offers anyone the opportunity to create their own reality instead of simply consuming it.

An entire community of Bally users exists in Chicago, who could otherwise never afford, because of finances or little computing background, to work with computer graphics. Computer art classes are being taught at both the University of Illinois and the Art Institute of Chicago using the Bally. Since you see what you are programming, a small graphics computer like this is especially educational for beginners.

HARDWARE

The electronic heart of the Arcade, the CPU(Central Processing Unit), consists of a Z-80 microprocessor and three chips --data, address, and I/O(Input and Output), that were designed for the unit. These custom chips allow the Arcade to produce sounds and pictures displayed on TV. They also allow the Bally to produce a recordable NTSC video signal, a feature appreciated by video communicators and not yet provided by most home computers.

The data chip sends out pixel information, looks up its color out of 256 choices, and provides the TV sync. The address chip then sequences the information and sends it to the proper location in the frame buffer. The I/O chip scans all inputs enabling interactive use of the hand controls and keypad. It also contains the circuitry to make music.

The memory of the system is also located in the CPU. 8K of ROM(Read Only Memory) is set up permanently to run the video games. 4K of RAM(Read Write Memory) with a resolution of 160*88 pixels, holds the frame buffer, which stores one frame of the TV screen in memory. The screen is divided into a grid of discrete picture elements called pixels. Each pixel corresponds to a particular place in memory. The frame buffer lets images be moved quickly on the screen, producing fast action video games in 8 colors.

ELECTRONIC TOY TURNS CREATIVE COMPUTER

With the addition of a programming language the small entertainment box is transformed from an electronic toy to a creative tool. Jay Fenton, the programmer of the Arcade, took TINYBASIC, a simple language designed to teach computing and in use on most home computers, and enhanced it with sensible graphics commands to create a flexible user-oriented language, BALLY BASIC.

The change begins with the insertion of the 4K BALLY BASIC ROM. The Basic divides the RAM formerly used entirely as a screen buffer for the game into three sections: screen, program and fixed variable storage. The Bally Arcade now offers 1800 bytes of memory space to store programs.
Hand controls, formerly used as paddles in the games now can be scanned under program control. The keypad,converted with a plastic overlay which lists the 16 main BALLY BASIC commands, lets you enter a command with one keystroke. As a trade-off in this expansion, BALLY BASIC loses the videogames´s ability to use 8 colors on the screen at once. Only 2 colors at a time, foreground and background color, remain .
All of these transformations are accomplished without hardware modification of the standard videogame. So with the minimal cost of a 4K ROM cassette you have turned your game into a programmable computer.

GRAPHICS COMMANDS

BALLY BASIC takes up the central problem of personal computing, that is providing a sensible yet open minded programming language to non-experts. Two main graphics commands, LINE and BOX , as well as the four optional drawing modes provide its visual power. To define a LINE statement only three arguments are necessary: LINE X,Y,C.
X and Y specify the coordinates the line should be drawn to, from the location of the last line. LINE simulates the method people use when drawing, connecting the end of one line to the next one.

Since it is a raster system the assembly language routine underlying the LINE command has to figure out which discrete pixels to turn on in order to draw as approximate a straight line as possible without bothering the user with this task. The BOX statement functions similarly, with a description of the width and height of the box as additional information: BOX X,Y,A,B,C. Again, the calculation of the field of pixels being filled is done by the assembly language routine, speeding up this procedure as well as enabling the user to concentrate on the design. Since the box command, already used by the video game to paint large areas of the screen, is fairly quick, many users gravitate to the playing field of rectangular shapes.

For both commands four different drawing options are at hand:

1. Draw in foreground color
2. Draw in background color
3. Exclusive-or--checks what value the pixel to be drawn currently has and reverses that
4. Just move the "pen" to the next location without drawing

We will later see how a conscious choice or even change of the drawing mode can effect and enhance the image. All images are plotted on the screen using the cartesian coordinate system with the origin (0,0) at the screen center.

The fixed variables of the system interface with the lower level assembly language giving the user access to options routinely used by the operating system. Jay's philosophy is: "if the system uses it, let the user control it too."
For example the variable PX checks if a pixel is currently set on or off. The video game uses PX to check for object boundaries. Many interactive design programs depend on this feature to read a pattern from the screen and store it.
Other fixed system variables allow control over musical notes and timing, position of text printing, and feedback from the hand controls.

INTERACTIVE PROGRAMS

One approach to using a graphics computer is to simulate pen and paper. A drawing program developed by Dan Sandin lets the user be unconcerned about programming. Lines are drawn or erased just triggering the hand control. Pictures can also be stored or recalled from audio tape by the press of a button on the keypad. Putting a major drawing program into a tiny 1800 byte memory space is a major feat.
Dan's main trick to save space lies in packing the X and Y coordinates of the end points of a line, as well as the information about the drawing mode being used, into a single byte (which is the smallest unit of memory.)

```
        @A = (X+100)*100+Y+50
```

@A is a one-dimensional array provided by Bally BASIC for storage of data. So that the X and Y data stay separated and positive, the X coordinate is multiplied by 100. It changes into a large number which is stored in the high end of the byte. The Y coordinate stays a small number in the low end of the same byte. To redraw the image, this this procedure is reversed: dividing by 100 and subtracting 100 to get the X back and retrieving Y from the remainder, RM-50. If the number changes into a negative one, it tells the drawing program that this line gets moved and not drawn.

Another program giving easy graphic access to the computer was

developed by Patti Harrison and Dan Sadowski. Their program uses mathematical symmetry operations to create intrinsic designs. Hand controls are the drawing input here too. The whole screen becomes a drawing grid on which the user creates a basic design. The X and Y coordinates of the pattern are stored much like the previous drawing program. A menu then lets you choose the scale and ordering of the final design.
These miniatures by Copper Giloth first show one basic cell undergoing two different symmetry transformations, and then two cells with different transformation patterns.

The interplay between one person and the computer has been extended to include groups of people acting together to create computer graphics and music. At an event titled, "A Test of the Interactive Broadcast System", the Arcade turned into a two-way communication system resembling the QUBE cable system in Columbus Ohio. "What we're trying to prove here is that you don't need all that sophisticated technology to do interactive two-way TV," said Dan Sandin. 20 homemade response boxes, acting like hand controls were linked together and wired into the I/O chip of the Arcade. The low cost computer tabulated the results in less than a second and fed them into graphics and music programs. This polling was done by user written assembly language programs which considerably sped up the process of evaluating the responses of the audience.

The programming tasks were to write programs that created pleasing images no matter what the input and moreover to give enough feedback to the audience so they knew what they were controlling. This graphics program by Zsuzsanna Molnar uses four-fold symmetry with boxes and overlays a fifth overlaid box to produce a variety of images. The audience jointly defined the vertical and horizontal growth rate of the boxes. Large numbers created the blockier patterns on the left.[note images a,c]

IMAGE GENERATION

Instead of drawing with the joysticks, of course it is possible to generate images by describing the coordinates of a picture entirely with a program. A number of different approaches to picturemaking are possible whether you start with a design and then program a succession of coordinates or you develop an abstract programming concept which then produces certain images.

Mark McKernin used an Algerian carpet design as the basic figure for his program GEO. He measured its dimensions on graph paper and incorporated these coordinates in the drawing section of his program , carefully emphasizing the sequence of the line draws in order to accomplish the pseudo 3-D effect. Once the image was completed, he continued overlaying it with boxes of decreasing size, thus changing the appearance of the basic pattern.

While Mark's program doesn't have to calculate anything but follows explicitly given LINE statements, Julie Boren sets up a much shorter program, SUN, containing three loops to compute the numerous lines composing this complex image. She scans the whole screen with lines arranged like the beams of the rising sun. Keeping the middle of the bottom line as a fixed starting point for all of the lines, she computes the second point of each line by feeding the loops all of the possible coordinates in a clockwise movement from left side to the upper side and finally to the right

screen boundary.
Adjacent lines have some pixels in common, which means that some lines get drawn several times. Because Julie uses the exclusive-or drawing mode, the final picture visually represents the results of the line drawing algorithm enabled by the LINE statement.
The set of patterns shown here demonstrate the use of various increments used to step through the basic loops. It is worth mentioning that it takes only a couple of seconds to complete one of these pictures.
Dan Sandin needs about 15 minutes to finish a single one of these MOD images, consisting of circles which seem to be mapped onto globes.

```
        MOD 2

        CLEAR
        D=5
        FOR Y=43 to -44 STEP -1
                D=D+2
                        FOR X= -80 to 79
                                M= (X*X+Y*Y) /D /2
                                IF RM = 0 BOX X,Y,1,1,1
                        NEXT X
        NEXT Y
        STOP
```

Some of the important aspects of this compact 10 line program are:
1. Only two nested loops create the basic structure which scans the whole screen
2. The computation of X and Y squared for each pixel and the RM command which returns the remainder of a division, makes it possible to generate circles even in an integer basic.
3. The size of the circles depends on the value of the division factor--D. Their growth is controlled by incrementing D after one vertical line is scanned, thus also generating the 3-D effect. Without incrementing D the result would be a regular pattern of circles of the same size.

CONTROLLED RANDOMNESS

Since the early days of computer graphics the use of random numbers has been a standard method to achieve images which are not predictable in detail and thus "cheat" the usual precision of the machine and its aesthetics of overall visual regularity. The main problem when working with random numbers is to set up a program which generates not totally entropic or "noisy" images, but ones structured by recognizable parameters. One example of our experiments in this area is the NEAR NEIGHBOR program by Dan Sandin. The program consists of three parts: first a coordinate inside the screen boundaries is chosen by the random number generator of the Arcade. Then the program checks whether any of the surrounding pixels of the randomly picked coordinate have already been drawn. This is done by looking up the PX values of the neighbors. If there are no surrounding pixels, the program executes its final task and plots a pixel-wide box at the randomly determined coordinate. Otherwise it returns without drawing and gets another coordinate. Two different versions of the program exist: one checks the four positions along the sides, the other checks the corners of the pixel. Rick Frankel elaborated on this concept by proceeding in two steps. He starts filling the screen with larger boxes and then continues printing small ones. Both versions output distinctly different visual entities.

Another interesting case of controlled randomness has been written by the computer film animator Larry Cuba. His SCROLL programs present the structured concept of adding more loops to an existing basic program to create more complex images.

```
SCROLL #1

S = 4 + RND(4)
C= RND (S-1)
FOR A = -72 to 70 STEP S
        BOX A,-39,C,8,1
NEXT A
PRINT
```

The bottom line of the screen is always cleared by the PRINT statement which causes the image to scroll 8 pixels up the screen. All boxes are now being drawn into that line with exactly the same height, since the Y coordinate and the height of the boxes are kept constant. Only the step S of the loop controlling the scan along the X-axis varies randomly and is correlated at the same time to the width of the boxes, so there is at least a space of one pixel left between two boxes.

```
SCROLL #2

S= 4 + RND(4)
C= RND (S-1)
FOR A = -72 to 79 STEP S
        T = RND (3) +1
        FOR B = -43 to -36 STEP T
                BOX A,B,C,1,1
        NEXT B
NEXT A
PRINT
```

The inclusion of a nested loop using B, also being randomly determined, and slightly different arguments for the BOX command in an otherwise unchanged program are responsible for the visual difference of SCROLL #2. Each time an X coordinate has been evaluated, the program enters a second loop, where it randomly chooses Y coordinates all lying within the range of the bottom scroll line. With every new Y value a box will be drawn that is only one pixel high but is as wide as the boxes of SCROLL #1. The final version, SCROLL #3 merges both #1 and #2 into one program, executing first the loop of #1 and then the two nested loops of #2. The outcome looks like a superimposition of the images of the first two programs. Since Larry switches to the exclusive-or drawing mode for the second scan along the X-axis, new patterns emerge from the overlay. All three programs produce highly structured, yet in part random visual output changing through time due to the upward movement of the scroll.

ANIMATION

To animate computer generated images requires a lot of computing power dedicated to calculate the transformations for each object. Most computer animation films are produced by shooting single frames, each of which needs considerable time to be computed. Currently only expensive and specialized computer systems which feature hard-wired matrix computation for the updating of the vector lists are suited for real-time animation of 3-D objects. If we talk about real-time animation done on the Bally Arcade, it is therefore assumed that as a low-cost system it can only handle the gradual change of images in time. So we would consider Larry Cuba's programs, discussed above, as animation. Sue Forner also uses scroll as the major means of animating her jewel-like patterns representing statistically weighted randomness. She encorporates the algorithm RND (RND (80))), which tends to give numbers around 0. Then four-fold symmetry plots the points. Once an image is complete, it is lifted upwards by the scroll and more points are printed toward the right and left screen boundaries. Thus, a rocket-like effect originates.

A very different approach is followed by Steve Wengerski when he lets a bundle of arrows wander across the screen. Each time he draws a new one, he erases the last one by simply drawing over it in the exclusive-or mode. By superimposing two bundles of arrows going in opposite directions, delicate patterns emerge, slowly changing in time. The great variety he achieves out of this fairly-simple set-up is due to the fact that the program switches between the plop and exclusive-or drawing modes.

A similar method is used by Frank Dietrich who overwrites three different versions of one basic tessellation in the frame buffer. The program ARABESQUE consists of only five parts: the drawing subroutine produces a package of eight boxes, a subroutine decrementing the size of the boxes and three routines specifying number, size and coordinates of the sets of the eight boxes. A counter switches the sequences in which the routines are executed so all possible permutations of the basic design are generated. The most interesting animation effect--the movement of replacing one set of boxes by another set and the gradual reduction of a the superimposition of all three sets to a single one--can't be reproduced by these still photographs.

Considering the restricted resources of the system, it is astounding enough that even these limited animation options exist. These examples demonstrate one more time our overall thesis, that this system represents a quite powerful and succeessful design enabling the creative use of a computer originally built as a video game just offering repetitious and in the long run boring cheap thrills. Sometimes the machine still shows its game heritage when it "flakes out", playing its own visual games and leaving the programmer dazzled by the beauty of these pictures s/he could never program.

Note: For the past three years an extension of the Bally

Arcade has been under development by the Electronic Visualization Laboratory, a collaboration between electronic artists and computer engineers. The new system, the UV-1 ZGrass Computer System has more memory,64K, and further expands on the principle of an easy-to use interactive graphic language. An article describing the system will be published in the November 1980 issue of BYTE magazine. Written by Tom DeFanti, it is titled "Language Control Structures for Easy Electronic Visualization."

PIX LIST

---

1 a Amateur TV
  b Buffalo
By Jane Veeder and Phil Morton

---

2 a-d Miniatures
by Copper Giloth

---

3 a-d Z-Boxes
by Zsuzsanna Molnar

---

4 a,b Geo
by Mark McKernin

---

5 a-c SUN
by Julie Boren

---

6 a,b MCD 2
by Dan Sandin ****already sent to you***

---

7 a Near Neighbors, Corner Check
  b Near Neighbors, Side Check
by Rick Frankel

---

8 a SCROLL #1
  b SCROLL #2
  c SCROLL #3
by Larry Cuba

---

9 a Cheap Thrills