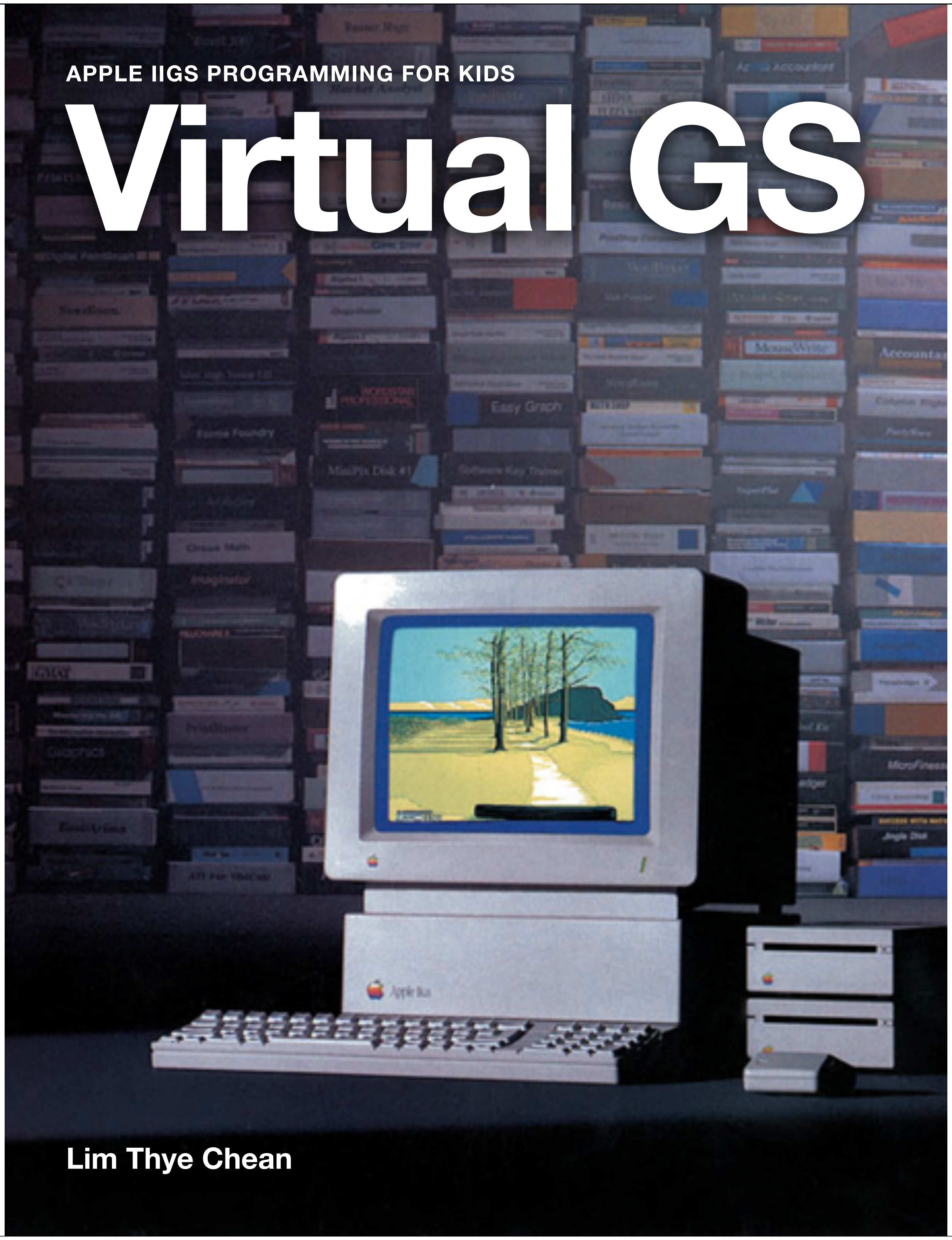
APPLE IIGS PROGRAMMING FOR KIDS

Virtual GS



Lim Thye Chean

iBooks Author

Chapter 1

Virtual GS

To bring back the fun and excitement of Apple IIGS programming for all the young children!

1

Welcome

Apple IIGS is one of the best computers Apple had ever produced, and through emulation it works perfectly in all modern machines. It is very simple to use and allow the young generation to do graphics and animation programming easily.

Welcome to the world of Apple IIGS!

2

Apple IIGS

Apple IIGS arrived on 15th September 1986 and it is the final computer in Apple II series. It is a 16 bit computer and has both excellent graphics and music capabilities at that time.

You can now build up your own Apple IIGS using any Mac or PC. All you need is just the emulator, an Apple IIGS ROM image and a copy of System 6 operating system disk, and you will be ready!

There are a few good Apple IIGS emulators:

• Mac OS X: Sweet16

• Mac OS 9: Bernie][The Rescue

• Windows/Linux: KEGS

Have fun!

Image Credit: Apple Inc.

3

Graphics Modes

The time is 1986, where the PC has EGA graphics and the Mac is still in monochrome, Apple IIGS has excellent color palette of 4096 colors! Together, Apple IIGS, Amiga and Atari ST ruled the multimedia world.

Apple IIGS has 2 graphic modes - 320 mode and 640 mode. 320 mode refers to the graphic resolution 320x200, with 16 colors per line. 640 mode refers to the graphic resolution 640x200 (but we won’t be covering 640 mode here). These resolutions look primitive nowadays but they were amazing at that time.

The 16 colors in the color palette can be changed to any color at anytime, allowing for some interesting effects like color cycling.

4

Winners do use Tools

If you have seen an Apple IIGS in action, you might have seen the amazing FTA demos in action! FTA team has done all the cool demos on a rather slow computer without using programming libraries called the Apple IIGS toolbox.

As the emulated Apple IIGS is often much faster than the real thing (a new Mac running emulation can run as fast as 30 to 100MHz Apple IIGS speed), you can now do the same thing using simple toolbox calls.

FTA
Free Tools Association
Presents :
NUCLEUS
" It's More Fun To Compute ! "
Featuring :
'El Mathos', PAD, The Gog's, SPK and Deny.

Chapter 2

Introduction

There are many programming software for the Apple IIGS, including one that is build into the ROM called Applesoft BASIC.

However, Applesoft BASIC does not take advantage of the Apple IIGS features like Super Hires modes. So we will look into other more modern languages.

1

Programming Languages

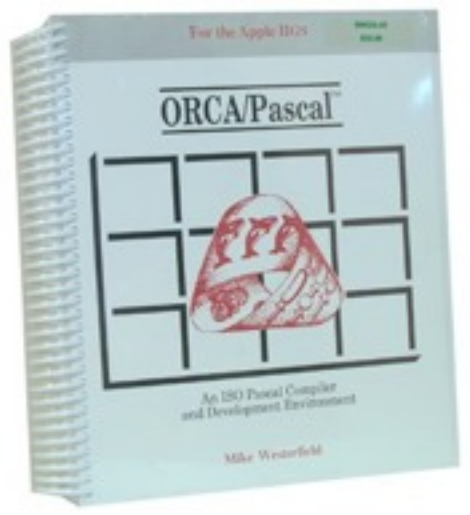
GSoft BASIC is the modern version of the Applesoft BASIC interpretor that support Apple IIGS features. Orca/Pascal supports Object Oriented Programming.

Complete Pascal (formally TML Pascal II) is probably the best solution for development. It has a modern language, support Apple IIGS features, good desktop editor, much faster performance, and allowing programmer to do full screen graphic programming without much technical knowledge.

We will be using Complete Pascal for all our programming examples. We will also be writing in Orca/Pascal and GSoft BASIC for comparison purposes. In many GSoft BASIC programs, you will need to press CTRL-C to stop the program.

You can download the development systems which includes Complete Pascal, GSoft BASIC and all the sample codes from Virtual GS web site (http://virtualgs.larwe.com). Orca/Pascal can be purchased from Syndicomm (http://store.syndicomm.com).

You can learn the Pascal language with books or Learn Pascal (http://www.taoyue.com/tutorials/pascal/contents.html) tutorial. This book will not teach Pascal or BASIC programming.



2

Programming the Apple IIGS

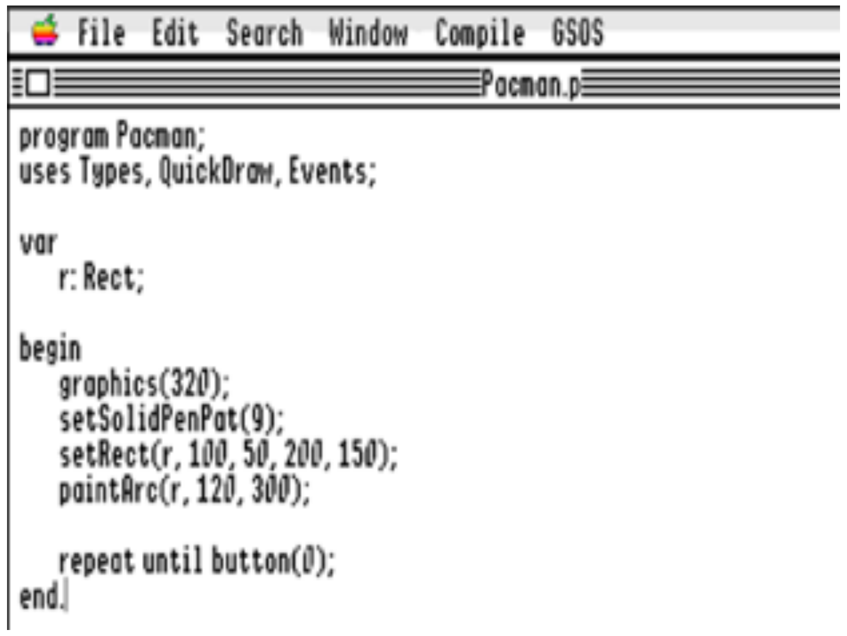
Once you get everything setup, please type in the following program, compile to memory, and what did you see? This is as easy as it can be!

```
program Pacman;
uses Types, QuickDraw, Events;
var r: Rect;

begin
  graphics(320);
  clearScreen(black);
  setSolidPenPat(9);
  setRect(r, 100, 50, 200, 150);
  paintArc(r, 120, 300);
  repeat until button(0);
end.
```

You should always start the program by graphics(320) for 320 mode or graphics(640) for 640 mode Super Hires.

The 320 mode has 320x200 resolution with 16 colors palette, while 640 mode has 640x200 resolution with 4 colors palette. Each screen can have a total of 16 palettes from the choice of 4096 colors.

You can then use different libraries in the Apple IIGS toolbox to perform different functions. In many cases, you will also need to use the Types library as those libraries are dependent on it.



```
program Pacman;
uses Types, QuickDraw, Events;

var
   r: Rect;

begin
   graphics(320);
   setSolidPenPat(9);
   setRect(r, 100, 50, 200, 150);
   paintArc(r, 120, 300);

   repeat until button(0);
end.
```

3

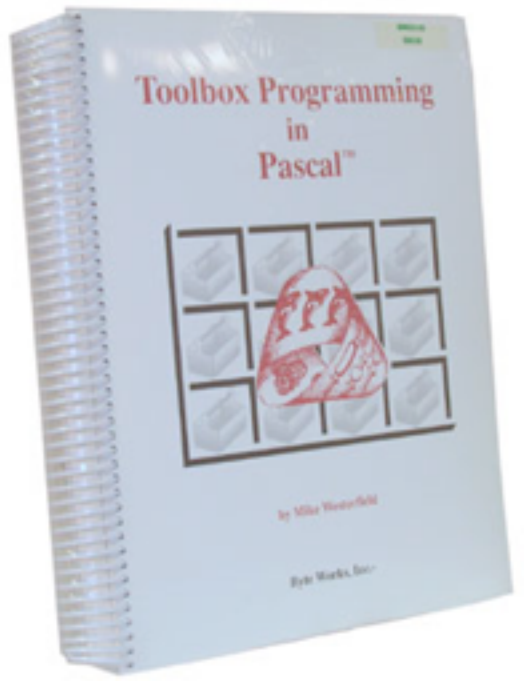
Apple IIGS Toolbox

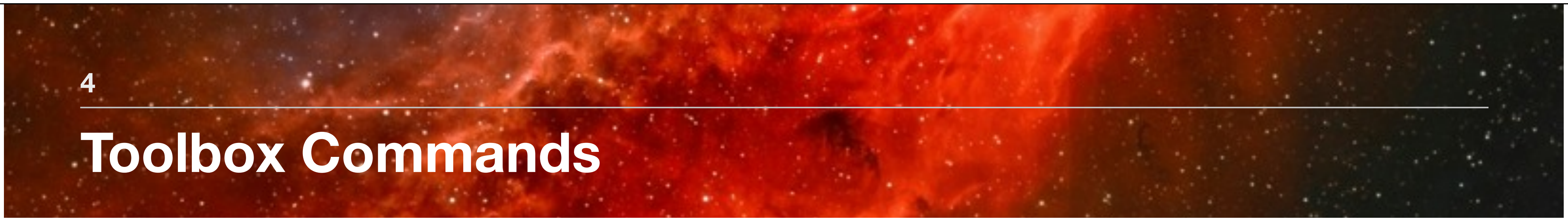
The Apple IIGS toolbox contains many libraries and we are only describing a few that will be useful for our programming purpose.

QuickDraw library is the standard graphics library. It has many commands that can help you draw many type of shapes. You can also change the color palette for each line and the color within the palette to use many colors.

Events library contains commands for you to read mouse and keyboard status.

MiscTool Library contains some miscellaneous commands like beeping the speaker or get the system tick count (number of 60th-second intervals since the system started).





4

Toolbox Commands

Here are some Toolbox commands that we will be using in the sample codes.

QuickDraw

Clear Screen	clearScreen(color)
Hide Mouse Pointer	hideCursor
Write Text	drawString(text)
Set Foreground Text Color	setForeColor(color)
Set Background Text Color	setBackColor(color)
Set Foreground Pen Color	setSolidPenPat(color)
Set Background Pen Color	setSolidBackPat(color)
Set Pen Size	setPenSize(width, height)
Set to Default Pen	penNormal
Move Pen	move(x, y)
Move Pen to	moveTo(x, y)
Draw Line	line(x, y)
Draw Line to	lineTo(x, y)
Set Rectangle Size	setRect(rect, x1, y1, x2, y2)
Draw Rectangle	frameRect(rect)
Paint Rectangle	paintRect(rect)
Erase Rectangle	eraseRect(rect)
Draw Round Rectangle	frameRRect(rect, oval width, oval height)
Paint Round Rectangle	paintRRect(rect, oval width, oval height)
Erase Round Rectangle	eraseRRect(rect, oval width, oval height)
Draw Oval	frameOval(rect)
Paint Oval	paintOval(rect)
Erase Oval	eraseOval(rect)
Draw Arc	frameArc(rect, start angle, arc angle)
Paint Arc	paintArc(rect, start angle, arc angle)
Erase Arc	eraseArc(rect, start angle, arc angle)
Start Polygon	openPoly
End Polygon	closePoly
Check Point In Rectangle	ptInRect(point, rect)
Get Random Number	random
Set Random Seed	setRandSeed(seed)
Set Palette (320 mode)	setSCB(line, palette)
Get Color In Palette	getColorEntry(palette, color)
Set Color In Palatte	setColorEntry(palette, color, color value)

Events

Mouse Button Down	button(0)
Mouse Button Still Down	stillDown(0)
Get Mouse Location	getMouse(point)
Get Next Event	getNextEvent(mask, event)

MiscTools

System Beep	sysBeep
Get System Tick	getTick

5

320 Mode

We will be primary using 320 mode in our sample codes. This mean the screen has the resolution of 320 horizontal lines and 200 vertical lines. Whenever you use a color in 320 mode, you will be referencing the color from 0 to 15.

Here is the color table of the standard palette.

0	Black
1	Dark Grey
2	Brown
3	Purple
4	Blue
5	Dark Green
6	Orange
7	Red
8	Flesh
9	Yellow
10	Green
11	Light Blue
12	Lilac
13	Periwinkle Blue
14	Light Grey
15	White

To reference a point in the 320 mode, you will count from top left with the origin point *0, 0* (which means x position is 0, and y position is 0) in the top left corner. A point *50, 100* will means 50 point to the right of the origin, and 100 to the bottom of the origin.

We won't be discussing about the 640 mode as it is more complex and will not be used in the sample codes.

Pictures taken from the Apple IIGS 320 mode

Chapter 3

Fun & Games

Programming is supposed to be fun. It is supposed to be as easy as sitting down, type a few commands, and run to see the result. This is where you come to learn all the fun programming!

In the next chapter, we will follow one young programmer on his programming journey!

1

Programming is Fun

Each sample code will attempt to teach you something, from simple to hard, and you will have a chance to learn a simple paint program and a space game!

All the programs are developed and tested on Sweet16 emulator running on Mac OS X using Complete Pascal. They should work on other emulators or the real Apple IIGS. However, the codes might be too slow for the original or even the accelerated Apple IIGS.

iBooks Author

2

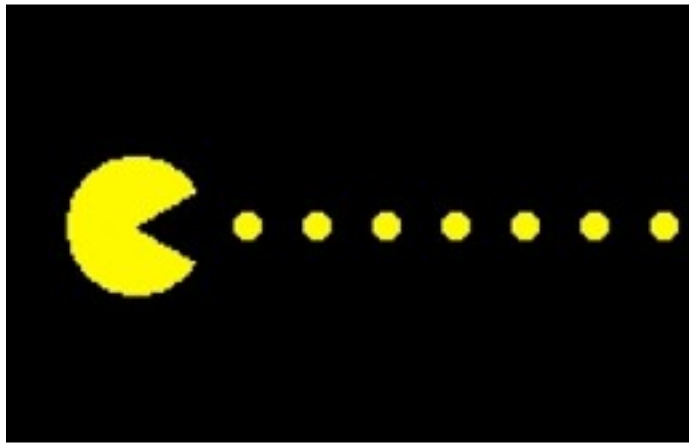
Demo

Beautiful stars and line arts.

This program teaches simple QuickDraw commands.

```
program Demo;
uses Types, QuickDraw, Events;

var
    i, col: Integer;

begin
    graphics(320);
    hideCursor;

    while true do begin
        clearScreen(black);

        for col := 1 to 15 do begin
            setSolidPenPat(col);

            for i := 0 to 31 do begin

                { Draw star }

                moveTo(random mod 320, random mod 200);
                line(0, 0);

                { Draw lines }

                moveTo(i * 10, 0);
                lineTo(0, 200 - i * 7);
                moveTo(320 - i * 10, 199);
                lineTo(319, i * 7);
            end;
            if button(0) then halt;
        end;
    end;
end.
```

Pacman


Pacman munching!

This program teaches simple animation technique, including using system tick to maintain animation speed and reduce flickering.

```
program Pacman;
uses Types, QuickDraw, Events, MiscTool;

var
     i, j: Integer;
     tick: LongInt;
     r: Rect;

begin
     graphics(320);
     clearScreen(black);
     hideCursor;
     setSolidBackPat(black);
     setSolidPenPat(9);
     tick := getTick;

     { Draw dots }

     for i := 1 to 15 do begin
          setRect(r, i * 20, 96, i * 20 + 8, 104);
          paintOval(r);
     end;

     { Draw Pacman }

     setRect(r, 0, 80, 40, 120);
     paintArc(r, 120, 300);

     for i := 1 to 80 do begin

          { Erase Pacman }

          setRect(r, i * 4 - 4, 80, i * 4 + 36, 120);
          eraseOval(r);

          { Move Pacman }

          setRect(r, i * 4, 80, i * 4 + 40, 120);
          if odd(i) then paintArc(r, 100, 340)
          else paintArc(r, 120, 300);

          { Digest food }

          repeat until getTick - tick > 3;
          tick := getTick;
     if button(0) then exit;
     end;
end.
```

The is the same program written in GSoft BASIC:

```
DIM R AS RECT

HGR
HCOLOR= 9
SETSOLIDBACKPAT (0)
TICK = GETTICK

! Draw dots

FOR I = 1 TO 15
     SETRECT (R, I * 20, 96, I * 20 + 8, 104)
     PAINTOVAL (R)
NEXT

! Draw Pacman

SETRECT (R, 0, 80, 40, 120)
PAINTARC (R, 120, 300)

FOR I = 1 TO 80

     ! Erase Pacman

     SETRECT (R, I * 4 - 4, 80, I * 4 + 36, 120)
     ERASERECT (R)

     ! Move Pacman

     SETRECT (R, I * 4, 80, I * 4 + 40, 120)
     IF I - INT (I / 2) * 2 > 0 THEN
          PAINTARC (R, 100, 340)
     ELSE
          PAINTARC (R, 120, 300)
     END IF

     DO UNTIL GETTICK - TICK > 3
     LOOP
     TICK = GETTICK
NEXT
```

4

Happy Mac

Little Mac bouncing happily, click on it and it will beep and bounce away!

This program teaches animation technique with more complex shape and mouse handling.

```
program Mac;
uses Types, QuickDraw, Events, MiscTool;

var
     i, x, y, sx, sy, color: Integer;
     tick: LongInt;
     r, mac: Rect;
     p: Point;

begin

     { Setup }

     tick := getTick;
     x := random mod 280 + 10;
     y := random mod 150 + 10;
     sx := 1;
     sy := 1;

     graphics(320);
     clearScreen(black);
     setSolidBackPat(black);

     while true do begin

          { Erase Mac }
          setRect(r, x, y, x + 25, y + 30);
          eraseRect(r);

          { Handle mouse click }

          if button(0) then begin
               getMouse(p);

               if ptInRect(p, mac) then begin
                    sysBeep;
                    x := random mod 280 + 10;
                    y := random mod 150 + 10;
                    sx := -sx;
                    sy := -sy;

                    while button(0) do;
               end
               else exit;
          end;

          { Draw Mac }

          x := x + sx;
          y := y + sy;

          setRect(mac, x, y, x + 25, y + 30);
          setSolidPenPat(15);
          paintRect(mac);
          setRect(r, x + 2, y + 3, x + 23, y + 18);
          setSolidPenPat(11);
          paintRRect(r, 8, 8);
          setSolidPenPat(black);
          setRect(r, x + 5, y + 5, x + 20, y + 15);
          frameArc(r, 90, 180);
          moveTo(x + 8, y + 6);
          line(0, 0);
          moveTo(x + 17, y + 6);
          line(0, 0);
          moveTo(x + 15, y + 23);
          line(7, 0);

          { Bounce }

          if (x < 0) or (x > 295) then sx := -sx;
          if (y < 0) or (y > 170) then sy := -sy;

          repeat until getTick > tick;
          tick := tick + 1;
     end;
end.
```

The is the same program written in GSoft BASIC:

```
DIM R AS RECT

TICK = GETTICK
X = RND (1) * 280 + 10
Y = RND (1) * 150 + 10
SX = 1
SY = 1

HGR
SETSOLIDBACKPAT (0)

DO

     ! Erase Mac

     SETRECT (R, X, Y, X + 25, Y + 30)
     ERASERECT (R)

     ! Move Mac

     X = X + SX
     Y = Y + SY

     ! Draw Mac

     SETRECT (R, X, Y, X + 25, Y + 30)
     SETSOLIDPENPAT (15)
     PAINTRECT (R)
     SETRECT (R, X + 2, Y + 3, X + 23, Y + 18)
     SETSOLIDPENPAT (11)
     PAINTRRECT (R, 8, 8)
     SETSOLIDPENPAT (0)
     SETRECT (R, X + 5, Y + 5, X + 20, Y + 15)
     FRAMEARC (R, 90, 180)
     HPLOT X + 8, Y + 6
     HPLOT X + 17, Y + 6
     HPLOT X + 15, Y + 23 TO X + 22, Y + 23

     ! Bounce

     IF X < 0 OR X > 295 THEN SX = - SX
     IF Y < 0 OR Y > 170 THEN SY = - SY

     DO UNTIL GETTICK > TICK
     LOOP
     TICK = TICK + 1
LOOP
```

5

Stars

Spaceship flying in star field.

This program teaches the use of array and record to create multiple objects. It uses array to generate many stars and animate them all at once at different speed. It also teaches how to use polygon to draw a shape.

```
program Stars;
uses Types, QuickDraw, Events, MiscTool;

type
     Star = Record
          x, y, speed: Integer;
     end;

var
     i: Integer;
     tick: LongInt;
     r: Rect;
     poly: Handle;
     stars: Array[0..19] of Star;

begin
     graphics(320);
     clearScreen(black);
     hideCursor;
     tick := getTick;

     { Draw spaceship }

     setSolidPenPat(7);
     setRect(r, 0, 93, 25, 97);
     paintOval(r);
```

```
poly := openPoly;
moveTo(10, 91);
lineTo(30, 95);
lineTo(10, 99);
closePoly;
setSolidPenPat(4);
paintPoly(poly);

setSolidPenPat(5);
moveTo(8, 91);
line(8, 0);
moveTo(8, 98);
line(8, 0);

{ Setup stars }

for i := 0 to 19 do begin
     stars[i].x := random mod 320;
     stars[i].y := i * 10;
     stars[i].speed := random mod 6 + 1;
end;

repeat
     for i := 0 to 19 do begin

          { Erase stars }

          setSolidPenPat(black);
          moveTo(stars[i].x, stars[i].y);
          line(0, 0);

          { Move stars }

          stars[i].x := stars[i].x - stars[i].speed;
          if stars[i].x < 0 then begin
               stars[i].x := 319;
               stars[i].speed := random mod 6 + 1;
          end;

          { Draw stars }

          setSolidPenPat(i mod 15 + 1);
          moveTo(stars[i].x, stars[i].y);
          line(0, 0);
     end;

     { Draw fire }

     if random > 0 then setSolidPenPat(9)
     else setSolidPenPat(7);

     moveTo(5, 94);
     line(4, 0);

     repeat until getTick > tick;
     tick := tick + 1;
until button(0);
end.
```

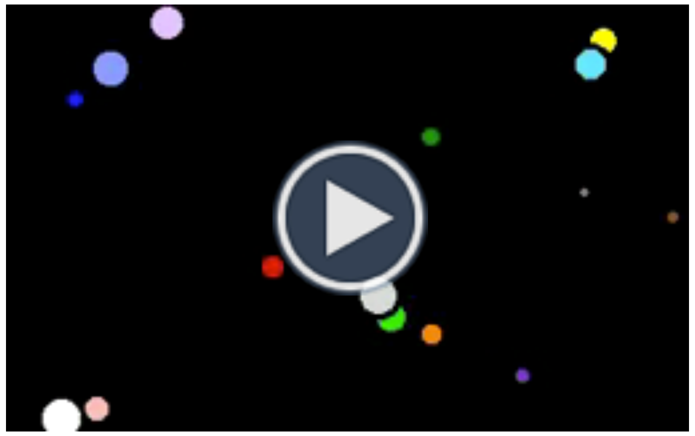
This is the Orca/Pascal version, using Object Pascal to demonstrate object oriented programming.

```
program Stars;
uses Common, QuickDrawII, EventMgr, MscToolSet;

type
     Star = Object
          x, y, speed: Integer;
          procedure setup;
          procedure move;
     end;

var
     i: Integer;
     tick: LongInt;
     r: Rect;
     poly: polyHandle;
     stars: Array[0..19] of Star;

procedure Star.setup;
begin
     x := randomInteger mod 320;
     y := i * 10;
     speed := randomInteger mod 6 + 1;
end;

procedure Star.move;
begin
     setSolidPenPat(black);
     moveTo(x, y);
     line(0, 0);
     x := x - speed;
     if x < 0 then begin
          x := 319;
          speed := randomInteger mod 6 + 1;
     end;

     setSolidPenPat(i mod 15 + 1);
     moveTo(x, y);
     line(0, 0);
end;

begin
     startDesk(320);
     clearScreen(black);
     hideCursor;
     tick := getTick;

     { Draw spaceship }

     setSolidPenPat(7);
     setRect(r, 0, 93, 25, 97);
     paintOval(r);

     poly := openPoly;
     moveTo(10, 91);
     lineTo(30, 95);
     lineTo(10, 99);
     closePoly;
     setSolidPenPat(4);
     paintPoly(poly);
     setSolidPenPat(5);
     moveTo(8, 91);
     line(8, 0);
     moveTo(8, 98);
     line(8, 0);

     { Setup stars }

     for i := 0 to 19 do begin
          new(stars[i]);
          stars[i].setup;
     end;

     repeat

          { Move stars }

          for i := 0 to 19 do
               stars[i].move;

          { Draw fire }

          if random > 0 then setSolidPenPat(9)
          else setSolidPenPat(7);

          moveTo(5, 94);
          line(4, 0);

          repeat until getTick > tick;
          tick := tick + 1;
     until button(0);

     for i := 0 to 19 do
          dispose(stars[i]);

     endDesk;
end.
```

This is the star field routine written in GSoft BASIC:

```
TYPE STAR
    X AS INTEGER
    Y AS INTEGER
    SPEED AS INTEGER
END TYPE

DIM STARS(15) AS STAR

! Setup stars

HGR
FOR I = 1 TO 15
    STARS(I).X = RND (1) * 320
    STARS(I).Y = I * 12
    STARS(I).SPEED = RND (1) * 6 + 1
NEXT

DO
    FOR I = 1 TO 15

        ! Erase star

        HCOLOR= 0
        HPLOT STARS(I).X, STARS(I).Y

        ! Draw star

        STARS(I).X = STARS(I).X - STARS(I).SPEED
        IF STARS(I).X < 0 THEN STARS(I).X = 319
        HCOLOR= I
        HPLOT STARS(I).X, STARS(I).Y
    NEXT
LOOP
```

Bouncing Balls



Many bouncing balls.

This program teaches the use of array to create multiple objects.

```
program Balls;
uses Types, QuickDraw, Events, MiscTool;

type
     Ball = Record
          x, y, sx, sy: Integer
     end;

var
     i: Integer;
     tick: LongInt;
     r: Rect;
     b: Ball;
     balls: Array[1..15] of Ball;

begin
     graphics(320);
     clearScreen(black);
     hideCursor;
     tick := getTick;
     setPenSize(3, 3);

     { Setup balls }

     for i := 1 to 15 do begin
          balls[i].x := random mod 300 + 10;
```

```
               balls[i].y := random mod 180 + 10;

               if random > 0 then
                    balls[i].sx := random mod 2 + 1
               else
                    balls[i].sx := -(random mod 2 + 1);

               if random > 0 then
                    balls[i].sy := random mod 2 + 1
               else
                    balls[i].sy := -(random mod 2 + 1);
          end;

          repeat
               for i := 1 to 15 do begin
                    b := balls[i];

                    { Draw balls }

                    setRect(r, b.x, b.y, b.x + 5 + i,
                    b.y + 5 + i);
                    setSolidPenPat(i);
                    paintOval(r);
                    setRect(r, b.x - 2, b.y - 2, b.x + 7 + i,
                    b.y + 7 + i);
                    setSolidPenPat(black);
                    frameOval(r);

                    { Move balls }

                    balls[i].x := b.x + b.sx;
                    balls[i].y := b.y + b.sy;

                    if (balls[i].x < 0) or (balls[i].x > 310)
                      then balls[i].sx := -b.sx;

                    if (balls[i].y < 0) or (balls[i].y > 190)
                      then balls[i].sy := -b.sy;
               end;

               repeat until getTick > tick;
               tick := tick + 1;
          until button(0);
     end.
```

The is the same program written in GSoft BASIC:

```
TYPE BALL
     X AS INTEGER
     Y AS INTEGER
     SX AS INTEGER
     SY AS INTEGER
END TYPE

DIM R AS RECT
DIM B AS BALL
DIM BALLS(15) AS BALL

HGR
SETPENSIZE (3, 3)

FOR I = 1 TO 15
     BALLS(I).X = RND (1) * 300 + 10
     BALLS(I).Y = RND (1) * 180 + 10

     IF RND (1) > 0.5 THEN
          BALLS(I).SX = INT ( RND (1) * 2) + 1
     ELSE
          BALLS(I).SX = - INT ( RND (1) * 2) - 1
     END IF

     IF RND (1) > 0.5 THEN
          BALLS(I).SY = INT ( RND (1) * 2) + 1
     ELSE
          BALLS(I).SY = - INT ( RND (1) * 2) - 1
     END IF
NEXT

DO
     FOR I = 1 TO 15
          B = BALLS(I)
          SETRECT (R, B.X, B.Y, B.X + 5 + I, B.Y + 5 + I)
          SETSOLIDPENPAT (I)
          PAINTOVAL (R)
          SETRECT (R, B.X - 2, B.Y - 2, B.X + 7 + I,
          B.Y + 7 + I)
          SETSOLIDPENPAT (0)
          FRAMEOVAL (R)

          BALLS(I).X = B.X + B.SX
          BALLS(I).Y = B.Y + B.SY

          IF BALLS(I).X < 0 OR BALLS(I).X > 310 THEN
               BALLS(I).SX = - B.SX
          IF BALLS(I).Y < 0 OR BALLS(I).Y > 190 THEN
               BALLS(I).SY = - B.SY
     NEXT
LOOP
```

Firework

This program teaches random number generation.

The random number will always be the same unless it started on a random seed - the current system tick is a good choice.

```
program Firework;
uses Types, QuickDraw, Events, MiscTool;

type
     Fire = Record
          x, y, xdir, ydir: Integer;
     end;

var
     i, j, x, y, color: Integer;
     tick: LongInt;
     fires: Array[0..30] of Fire;

begin
     graphics(320);
     clearScreen(black);
     hideCursor;
     tick := getTick;
     setRandSeed(tick);

     while true do begin

          { Setup firework }

          setPenSize(1, 1);
          color := random mod 15 + 1;
          x := random mod 320;
          y := random mod 100 + 50;

          for i := 0 to 30 do begin
               fires[i].x := x;
               fires[i].y := y;
               fires[i].xdir := random mod 8 - 4;
               fires[i].ydir := random mod 8 - 4;
          end;

          { Shoot fire }

          for i := 1 to 30 do begin
               setSolidPenPat(black);
               moveTo((i - 1) * x div 30,
               (i - 1) * y div 30);
               line(0, 0);

               setSolidPenPat(color);
               moveTo(i * x div 30, i * y div 30);
               line(0, 0);

               repeat until getTick - tick > 1;
               tick := getTick;
          end;

          { Draw firework }

          for i := 1 to 30 do begin
               for j := 0 to 30 do begin
                    setPenSize(i div 15 + 1, i div 15 + 1);
                    setSolidPenPat(black);
                    moveTo(fires[j].x, fires[j].y);
                    line(0, 0);

                    fires[j].x := fires[j].x + fires[j].xdir;
                    fires[j].y := fires[j].y + fires[j].ydir;
                    setSolidPenPat(color);
                    moveTo(fires[j].x, fires[j].y);
                    line(0, 0);

                    if button(0) then exit;
               end;

               repeat until getTick > tick;
               tick := tick + 1;
          end;

          clearScreen(black);
     end;
end.
```

This is the Orca/Pascal version, written using Object Pascal to demonstrate object oriented programming.

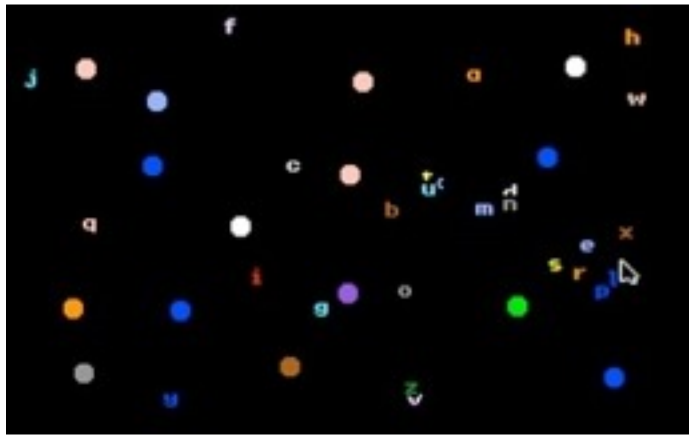
```
program Firework;
uses Common, QuickDrawII, EventMgr, MscToolSet;

type
     Fire = Object
          x, y, xdir, ydir: Integer;
          procedure setup(startX, startY: Integer);
          procedure move;
     end;

var
     i, j, k, x, y, color: Integer;
     tick: LongInt;
     fires: Array[0..30] of Fire;

procedure Fire.setup;
begin
     x := startX;
     y := startY;
     xdir := randomInteger mod 8 - 4;
     ydir := randomInteger mod 8 - 4;
end;

procedure Fire.move;
begin
     setSolidPenPat(black);
     moveTo(x, y);
     line(0, 0);
     x := x + xdir;
     y := y + ydir;
     setSolidPenPat(color);
     moveTo(x, y);
     line(0, 0);
end;

begin
     startDesk(320);
     clearScreen(black);
     hideCursor;
     tick := getTick;
     setRandSeed(tick);

     while true do begin

          { Setup fire }

          setPenSize(1, 1);
          color := randomInteger mod 15 + 1;
          x := randomInteger mod 320;
          y := randomInteger mod 100 + 50;

          for i := 0 to 30 do begin
               new(fires[i]);
               fires[i].setup(x, y);
          end;

          { Draw fire }

          for i := 1 to 30 do begin
               setSolidPenPat(black);
          moveTo((i - 1) * x div 30,
          (i - 1) * y div 30);
          line(0, 0);
          setSolidPenPat(color);
          moveTo(i * x div 30, i * y div 30);
          line(0, 0);

          repeat until getTick - tick > 1;
          tick := getTick;
     end;

     { Draw firework }

     for i := 1 to 30 do begin
          for j := 0 to 30 do begin
               setPenSize(i div 15 + 1, i div 15 + 1);
               fires[j].move;
               if button(0) then begin
                    for k := 0 to 30 do
                         dispose(fires[i]);
                    endDesk;
                    halt(0);
               end;
          end;

          repeat until getTick > tick;
          tick := tick + 1;
     end;

     clearScreen(black);
  end;
end.
```

The is the same program written in GSoft BASIC:

```
TYPE FIRE
     X AS INTEGER
     Y AS INTEGER
     XDIR AS INTEGER
     YDIR AS INTEGER
END TYPE

DIM FIRES(30) AS FIRE

TICK = GETTICK
DO

     ! Setup Firework

     X = RND (1) * 320
     Y = RND (1) * 100 + 50
     COLOR = RND (1) * 15 + 1

     HGR
     FOR I = 1 TO 30
          FIRES(I).X = X
          FIRES(I).Y = Y
          FIRES(I).XDIR = RND (1) * 8 - 4
          FIRES(I).YDIR = RND (1) * 8 - 4
     NEXT

     ! Shoot fire

     FOR I = 1 TO 30
          HCOLOR= 0
          HPLOT (I - 1) * X / 30, (I - 1) * Y / 30
          HCOLOR= COLOR
          HPLOT I * X / 30, I * Y / 30

          DO UNTIL GETTICK - TICK > 1
          LOOP
          TICK = GETTICK
     NEXT

     ! Draw firework

     FOR I = 1 TO 30
          FOR J = 1 TO 30
               HCOLOR= 0
               HPLOT FIRES(J).X, FIRES(J).Y

               FIRES(J).X = FIRES(J).X + FIRES(J).XDIR
               FIRES(J).Y = FIRES(J).Y + FIRES(J).YDIR
               HCOLOR= COLOR
               HPLOT FIRES(J).X, FIRES(J).Y
          NEXT
     NEXT
LOOP
```

8

Crazy Typer



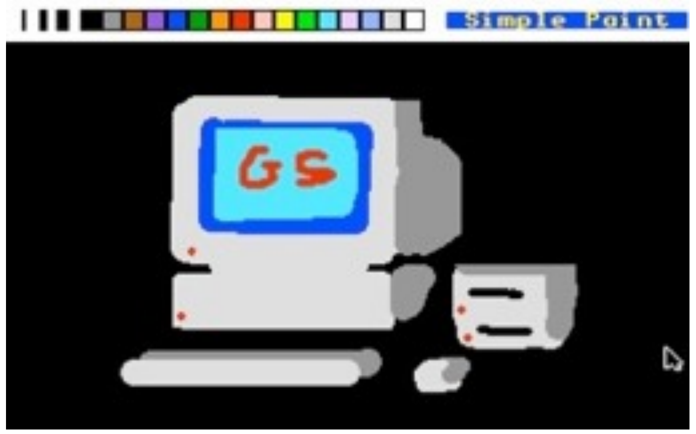
This program shows any keys typed on keyboard in random position and color on screen.

It also shows the current mouse position when clicked. Press the space bar to clear the screen, and press ESC to quit the program. It teaches keyboard and mouse event handling.

```
program CrazyTyper;
uses Types, QuickDraw, Events;

{ Handle key }

procedure handleKey(key: Integer);
begin
     setForeColor(random mod 15 + 1);
     moveTo(random mod 310, random mod 180 + 10);
     drawString(chr(key));
end;

{ Handle mouse }

procedure handleMouse(p: Point);
var
     circle: Rect;

begin
     setSolidPenPat(random mod 15 + 1);
     setRect(circle, p.h - 3, p.v - 3, p.h + 6, p.v + 6);
     paintOval(circle);
end;

var
     i, key, etypes: Integer;
     event: EventRecord;

begin
     graphics(320);
     clearScreen(black);
     setBackColor(black);
     etypes := keyDownMask + mDownMask;

     while true do begin
          if getNextEvent(etypes, event) then begin
               case event.what of

                    { Key down }

                    keyDownEvt: begin
                         key := event.message;

                         case key of
                              27: halt;
                              32: clearScreen(black);
                              otherwise handleKey(key);
                         end;
                    end;

                    { Mouse down }

                    mouseDownEvt: handleMouse(event.where);
               end;
          end;
     end;
end.
```

9

Simple Paint



A simple painting program for kids. You can change the paint brush size and color. If you press the mouse button long enough, the paint brush size will slowly expand to double its size. Press space bar to clear the picture and ESC to quit the program.

This program teaches the use of procedure and event driven programming.

Simple Paint is also available for iPhone, iPad and Android devices.

```
program SimplePaint;
uses Types, QuickDraw, Events;

var
     i, x, y, key, bsize, etypes: Integer;
     mouse: Point;
     r, tbar: Rect;
     bsizes: array[1..3] of Rect;
     colors: array[0..15] of Rect;
     event: EventRecord;

{ Draw screen }

procedure drawScreen;
begin

     { Draw title bar }

     clearScreen(black);
     setSolidPenPat(white);
     setRect(tbar, 0, 0, 319, 20);
     paintRect(tbar);

     { Draw brush sizes }

     setSolidPenPat(black);
```

```
          for i := 1 to 3 do begin
               setRect(bsizes[i], 5 + (i - 1) * 8, 5,
               15 + (i - 1) * 8, 15);
               setPenSize(i * 2, 1);
               moveTo(i * 8, 5);
               line(0, 9);
          end;

          penNormal;

          { Draw color bar }

          for i := 0 to 15 do begin
               setRect(colors[i], 35 + i * 10, 5, 45 + i * 10,
               15);
               setSolidPenPat(i);
               paintRect(colors[i]);
               setSolidPenPat(0);
               frameRect(colors[i]);
          end;

          { Draw title }

          moveTo(205, 13);
          setForeColor(9);
          setBackColor(4);
          drawString(' Simple Paint ');
          setSolidPenPat(white);
     end;

     { Draw dots }
```

```
procedure drawDots(x, y: Integer);
var
     ox, oy, ix, iy: Integer;
     size: Real;

begin
     size := bsize;

     while button(0) do begin
          getMouse(mouse);
          if not PtInRect(mouse, tbar) then begin

               { Paint dot }

               ox := x;
               oy := y;
               x := trunc(mouse.h - bsize div 2);
               y := trunc(mouse.v - bsize div 2);
               setRect(r, x, y, x + bsize, y + bsize);
               paintOval(r);

               { Paint connected dot }

               if (ox <> x) or (oy <> y) then begin
                    size := bsize;
                    ix := (ox + x) div 2;
                    iy := (oy + y) div 2;
                    setRect(r, ix, iy, ix + bsize,
                    iy + bsize);
                    paintOval(r);
               end
               else begin
```

```
                    { Enlarge paint area if pressed }

                    if (size < bsize * 2) then
                         size := size + 0.01;

                    setRect(r, x - trunc(size / 2),
                    y - trunc(size / 2), trunc(x + size),
                    trunc(y + size));

                    paintOval(r);
               end;
          end;
     end;
end;

begin
     graphics(320);
     bsize := 3;
     etypes := keyDownMask + mDownMask;
     drawScreen;

     while true do begin
          if getNextEvent(etypes, event) then begin
               case event.what of

                    { Key down }

                    keyDownEvt: begin
                         key := event.message;
                         if key = 27 then halt
                         else if key = 32 then drawScreen;
                    end;

                    { Mouse down }

                    mouseDownEvt: begin
                    mouse := event.where;

                    { Set brush size }

                    for i := 1 to 3 do
                         if ptInRect(mouse, bsizes[i]) then
                              bsize := i * 3;

                         { Set color }

                         for i := 0 to 15 do
                              if ptInRect(mouse, colors[i])
                                then setSolidPenPat(i);

                         drawDots(mouse.h - bsize div 2,
                         mouse.v - bsize div 2);
                    end;
               end;
          end;
     end;
end.
```

10

Slide Show

A slideshow program to show off the beauty of Apple IIGS.

This program shows how to load a 320x200 Super Hires picture on screen, then fade off. The loadImage procedure can be used as a *black box* (this means you do not have to understand how it works) in any program to load a Super Hires image.

The fading of the screen is done by reducing the intensity of red, green, blue components of all the colors in the palette.

```
program Slideshow;
uses Types, QuickDraw, Events, GSOS, CTIUtils, MiscTool;

const
     MAX = 7;

var
     tick: LongInt;
     title: Integer;
     titles: array[0..MAX] of String;

{ Load Image }

procedure loadImage(path: string);
var
     openRec: OpenRecGS;
     ioRec: IORecGS;
     closeRec: RefNumRecGS;
     pathNameGS: GSstring255;

begin

     { Open image }

     p2GSstring(path, pathNameGS);
     openRec.pCount := 2;
     openRec.pathName := @pathNameGS;
     openGS(openRec);

     { Read into memory }

     if _ToolErr = 0 then begin
          ioRec.pCount := 4;
          ioRec.refNum := openRec.refNum;
          ioRec.dataBuffer := ptr($E12000);
          ioRec.requestCount := 32768;
          readGS(ioRec);

          closeRec.pCount := 1;
          closeRec.refNum := openRec.refNum;
          closeGS(closeRec);
     end;
end;

{ Fade screen }

procedure fade;
var
     i, j, k, color, cbit: Integer;

begin
     cbit := 256;

     for i := 1 to 3 do begin
          for j := 0 to 15 do begin
               for k := 0 to 15 do begin
                    color := getColorEntry(0, k) - cbit;
                    if color < 0 then color := 0;
                    setColorEntry(0, k, color);
               end;

               repeat until getTick > tick;
               tick := getTick;
          end;

          cbit := cbit div 16;
     end;
end;

begin
     graphics(320);
     hideCursor;

     { Setup }

     title := 0;
     titles[0] := 'Pam.pic';
     titles[1] := 'Girl.pic';
     titles[2] := ‘RedHood.pic’;
     titles[3] := 'Clown.pic';
     titles[4] := 'Fruits.pic';
     titles[5] := 'Flower.pic';
     titles[6] := 'Taj.pic';
     titles[7] := 'Dragon.pic';

     repeat
          tick := getTick;
          loadImage(titles[title]);
```

```
            repeat
                 if button(0) then begin
                      fade;
                      exit;
                 end;
            until getTick - tick > 200;
            tick := getTick;

            inc(title);
            if title > MAX then title := 0;

            fade;
       until button(0);
  end.
```

11

Forever GS

Apple IIGS Forever! Now you can write your own FTA demo!

This program teaches psuedo 3D effect, multiple color palettes and color cycling.

This program uses FTA Tool219 to play music, please first install Tool219 to the Apple IIGS System folder. You can get Tool219 from the Virtual GS disk.

```
program ForeverGS;
uses Types, QuickDraw, Events, MiscTool, ST;

type
     Star = record
          x, y, sx, sy: Real;
     end;

var
     i, color, dir: Integer;
     tick: LongInt;
     s: Star;
     stars: Array[0..20] of Star;

procedure playMusic(path: String);
begin
     loadOneTool(219, 0);
     stStartup(57005);
     stLoadOneMusic(path);
     stPlayMusic(true);
end;

procedure stopMusic;
begin
     stShutDown;
end;
```

```
begin

     { Setup }

     graphics(320);
     clearScreen(black);
     playMusic('Music/Toolbox');
     setPenMode(modeXOR);
     hideCursor;
     tick := getTick;
     dir := -1;

     { Draw background }

     setColorEntry(0, 15, 0);

     for i := 1 to 15 do
          setColorEntry(1, i, i);

     for i := 125 to 199 do
          setSCB(i, 1);

     setPenSize(1, 5);

     for i := 1 to 15 do begin
          setSolidPenPat(i);
          moveTo(0, 120 + i * 5);
          line(320, 0);
     end;

     setPenSize(1, 1);
```

```
setBackColor(black);
setForeColor(15);
moveTo(85, 124);
write('Apple IIGS Forever!');

{ Setup stars }

for i := 0 to 20 do begin
     stars[i].x := 160;
     stars[i].y := 60;
     stars[i].sx := (random mod 41) / 10 - 2;
     stars[i].sy := (random mod 41) / 10 - 2;
end;

repeat

     { Color cycle title }

     color := getColorEntry(0, 15);
     if (color = 0) or (color = $F00) then
          dir := -dir;

     setColorEntry(0, 15, color + $100 * dir);

     { Color cycle background }

     color := getColorEntry(1, 15);

     for i := 15 downto 1 do
          setColorEntry(1, i, getColorEntry(1, i - 1));

     setColorEntry(1, 1, color);
```

```
{ Move stars }

for i := 0 to 20 do begin
     s := stars[i];
     setSolidPenPat(i mod 14 + 1);
     moveTo(round(s.x), round(s.y));
     line(0, 0);

     stars[i].x := s.x + s.sx;
     stars[i].y := s.y + s.sy;

     s := stars[i];
     if (s.x < 0) or (s.x > 319) or (s.y < 0) or
       (s.y > 119) then begin
          stars[i].x := 160;
          stars[i].y := 60;
          stars[i].sx := (random mod 41) / 10 - 2;
          stars[i].sy := (random mod 41) / 10 - 2;
     end
     else begin

          { Accelerate stars movement }

          stars[i].sx := s.sx * 1.1;
          stars[i].sy := s.sy * 1.1;
     end;

     s := stars[i];
     moveTo(round(s.x), round(s.y));
     line(0, 0);
end;
```

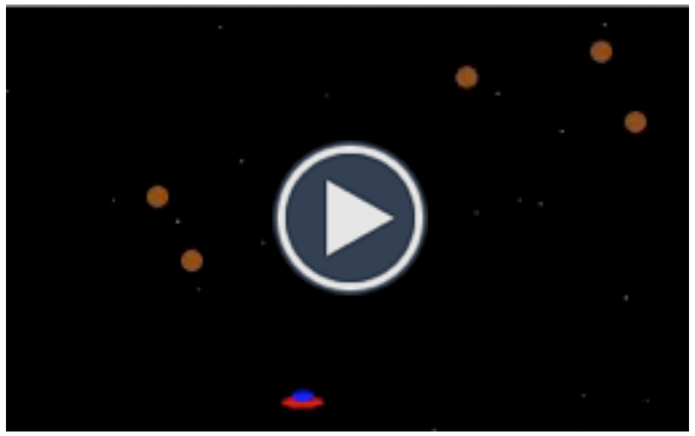
```
          repeat until getTick - tick > 1;
          tick := getTick;
     until button(0);

     stopMusic;
end.
```

This is the 3D star field routine written in GSoft BASIC:

```
TYPE STAR
     X
     Y
     SX
     SY
END TYPE

DIM S AS STAR
DIM STARS(20) AS STAR

HGR
SETPENMODE (MODEXOR)

! Write message

SETBACKCOLOR (0)
SETFORECOLOR (7)
MOVETO (85, 100)
PRINT “Apple IIGS Forever!”

! Setup stars

FOR I = 0 TO 20
     STARS(I).X = 160
     STARS(I).Y = 100
     STARS(I).SX = RND (1) * 4 - 2
     STARS(I).SY = RND (1) * 4 - 2
NEXT

DO
     ! Move Stars

     FOR I = 0 TO 20
          S = STARS(I)
          HCOLOR= (I / 21) * 15 + 1
          HPLOT S.X, S.Y

          STARS(I).X = S.X + S.SX
          STARS(I).Y = S.Y + S.SY
          S = STARS(I)

          IF (S.X < 0) OR (S.X > 319) OR (S.Y < 0)
            OR (S.Y > 199) THEN
               STARS(I).X = 160
               STARS(I).Y = 100
               STARS(I).SX = RND (1) * 4 - 2
               STARS(I).SY = RND (1) * 4 - 2
          ELSE

               ! Accelerate stars movement

               STARS(I).SX = S.SX * 1.1
               STARS(I).SY = S.SY * 1.1
          END IF

          HPLOT STARS(I).X, STARS(I).Y
     NEXT
LOOP
```

12

Space Game



Shoots down the asteroids as they approaches your spaceship! The nearer the asteroid, the higher the score!

This game demonstrates many concepts including array, record, procedure, QuickDraw commands, mouse event and collision detection technique.

Space Game has been ported to iPhone, Google Android and Java ME phones as Space War.

```
program Space;
uses Types, QuickDraw, Events, MiscTool;

const
     MAX_ROCKS = 5;

type
     Object = Record
          x, y, speed: Integer;
     end;

var
     i, j, k, x, y, pos, score: Integer;
     tick: longInt;
     mouse: Point;
     r: Rect;
     rock: Object;
     stars: Array[0..31] of Object;
     rocks: Array[1..MAX_ROCKS] of Object;
     gameOver: Boolean;

{ Draw spaceship }

procedure drawShip;
begin
     setRect(r, pos, 180, pos + 20, 188);
     eraseRect(r);
     getMouse(mouse);
     pos := mouse.h;
     if pos > 300 then pos := 300;

     setSolidPenPat(7);
     setRect(r, pos, 183, pos + 20, 188);
     paintOval(r);
     setSolidPenPat(4);
     setRect(r, pos + 5, 180, pos + 15, 185);
     paintOval(r);
end;

{ Draw stars }

procedure drawStars;
var
     i: Integer;

begin
     for i := 0 to 31 do begin
          setSolidPenPat(black);
          moveTo(stars[i].x, stars[i].y);
          line(0, 0);

          stars[i].y := stars[i].y + stars[i].speed;
          if stars[i].y > 199 then begin
               stars[i].y := 0;
               stars[i].speed := random mod 5 + 1;
          end;

          setSolidPenPat(i mod 15 + 1);
          moveTo(stars[i].x, stars[i].y);
          line(0, 0);
     end;
end;

{ Draw rocks }

procedure drawRocks;
var
     i: Integer;

begin
     for i := 1 to MAX_ROCKS do begin
          rock := rocks[i];
          setRect(r, rock.x, rock.y, rock.x + 10,
          rock.y + 10);
          eraseOval(r);

          rocks[i].y := rock.y + rock.speed;
          if rocks[i].y > 190 then begin
               rocks[i].x := random mod 310;
               rocks[i].y := 0;
               rocks[i].speed := random mod 6 + 3;
          end;

          rock := rocks[i];
          setSolidPenPat(2);
          setRect(r, rock.x, rock.y, rock.x + 10,
          rock.y + 10);
          paintOval(r);
     end;
end;
```

```
begin

     { Setup  }

     graphics(320);
     clearScreen(black);
     setBackColor(black);
     setForeColor(9);
     setSolidBackPat(black);
     hideCursor;

     score := 0;
     tick := getTick;
     gameOver := false;
     getMouse(mouse);
     pos := mouse.h;

     for i := 1 to MAX_ROCKS do begin
          rocks[i].x := random mod 310;
          rocks[i].y := random mod 100;
          rocks[i].speed := random mod 3 + 1;
     end;

     for i := 0 to 31 do begin
          stars[i].x := i * 10;
          stars[i].y := random mod 200;
          stars[i].speed := random mod 5 + 1;
     end;

     repeat
          drawStars;
          drawRocks;
          drawShip;

          repeat until getTick > tick;
          tick := tick + 1;

          { Check collision }

          for i := 1 to MAX_ROCKS do begin
               rock := rocks[i];
               if (rock.y > 170) and (rock.y < 190) and
                 (rock.x > pos) and (rock.x < pos + 20) then
                 begin

               { Explosion }

               x := rock.x;
               y := rock.y;

               for j := 1 to 15 do begin
                    k := j * 5;
                    drawStars;
                    drawRocks;
                    setSolidPenPat(j);
                    setRect(r, x - k, y - k, x + 5 + k,
                    y + 5 + k);
                    paintOval(r);

                    repeat until getTick - tick > 3;
                    tick := getTick;
               end;
          sysBeep;
               gameOver := true;
          end;
     end;

     { Fire laser }

     if button(0) then begin
          setSolidPenPat(white);
          moveTo(pos + 10, 0);
          lineTo(pos + 10, 179);

          for i := 1 to MAX_ROCKS do begin
               drawStars;
               rock := rocks[i];

               if (rock.x > pos) and (rock.x < pos + 10) then
               begin

               { Destroy rock }

                    score := score + rocks[i].y div 10;
                    setRect(r, rock.x, rock.y, rock.x + 10,
                    rock.y + 10);
                    eraseOval(r);
                    rocks[i].x := random mod 310;
                    rocks[i].y := 0;
                    rocks[i].speed := random mod 6 + 3;
                    leave;
               end;
          end;
```

```
                    setSolidPenPat(black);
                    moveTo(pos + 10, 0);
                    lineTo(pos + 10, 179);
               end;
          until gameOver;

          moveTo(50, 90);
          setForeColor(9);
          write('Game Over. Your Score is ', score, '.');
          repeat until button(0);
     end.
```

This is the Orca/Pascal version, written using Object Pascal syntax to demonstrate object oriented programming.

```
program Space;
uses Common, QuickDrawII, EventMgr, MscToolSet;

const
     MAX_ROCKS = 5;

type
     Star = Object
          x, y, speed, color: Integer;
          procedure setup;
          procedure draw;
     end;

     Rock = Object
          x, y, speed: Integer;
          procedure setup;
          procedure draw;
          procedure reset;
          function hasCollided(pos: Integer): Boolean;
          function hasDestroyed(pos: Integer): Boolean;
     end;

var
     i, j, x, y, pos, score: Integer;
     tick: longInt;
     mouse: Point;
     stars: Array[0..31] of Star;
     rocks: Array[1..MAX_ROCKS] of Rock;
     gameOver: Boolean;
     str: String;

{ Star }

procedure Star.setup;
begin
     x := randomInteger mod 320;
     y := randomInteger mod 200;
     speed := randomInteger mod 5 + 1;
     color := randomInteger mod 15 + 1;
end;

procedure Star.draw;
begin
     setSolidPenPat(black);
     moveTo(x, y);
     line(0, 0);

     y := y + speed;
     if y > 199 then begin
          y := 0;
          speed := randomInteger mod 5 + 1;
     end;

     setSolidPenPat(color);
     moveTo(x, y);
     line(0, 0);
end;

{ Rock }

procedure Rock.setup;
```

```
begin
     x := randomInteger mod 310;
     y := randomInteger mod 100;
     speed := randomInteger mod 3 + 1;
end;

procedure Rock.draw;
var
     r: Rect;

begin
     setRect(r, x, y, x + 10, y + 10);
     eraseOval(r);

     y := y + speed;
     if y > 190 then begin
          x := randomInteger mod 310;
          y := 0;
          speed := randomInteger mod 6 + 3;
     end;

     setSolidPenPat(2);
     setRect(r, x, y, x + 10, y + 10);
     paintOval(r);
end;

procedure Rock.reset;
var
     r: Rect;

begin
     setRect(r, x, y, x + 10, y + 10);
     eraseOval(r);
     x := randomInteger mod 310;
     y := 0;
     speed := randomInteger mod 6 + 3;
end;

function Rock.hasCollided;
begin
     if (y > 170) and (y < 190) and (x > pos) and
       (x < pos + 20) then hasCollided := true
     else hasCollided := false;
end;

function Rock.hasDestroyed;
begin
     if (x > pos) and (x < pos + 10) then
          hasDestroyed := true
     else hasDestroyed := false;
end;

{ Draw spaceship }

procedure drawShip;
var
     r: Rect;

begin
     setRect(r, pos, 180, pos + 20, 188);
     eraseRect(r);

     getMouse(mouse);
     pos := mouse.h;
```

```
    if pos > 300 then pos := 300;

    setSolidPenPat(7);
    setRect(r, pos, 183, pos + 20, 188);
    paintOval(r);
    setSolidPenPat(4);
    setRect(r, pos + 5, 180, pos + 15, 185);
    paintOval(r);
end;

{ Draw stars }

procedure drawStars;
var
    i: Integer;

begin
    for i := 0 to 31 do
        stars[i].draw;
end;

{ Draw rocks }

procedure drawRocks;
var
    i: Integer;

begin
    for i := 1 to MAX_ROCKS do
        rocks[i].draw;
end;

{ Draw explosion }

procedure drawExplosion(x, y, radius, color: Integer);
var
    r: Rect;

begin
    setSolidPenPat(color);
    setRect(r, x - radius, y - radius, x + 5 + radius,
      y + 5 + radius);
    paintOval(r);
end;

{ Draw laser }

procedure drawLaser(pos, color: Integer);
begin
    setSolidPenPat(color);
    moveTo(pos + 10, 0);
    lineTo(pos + 10, 179);
end;

{ Main }

begin

    { Setup  }

    startDesk(320);
    clearScreen(black);
    setSolidBackPat(black);
    hideCursor;
```

```
score := 0;
tick := getTick;
gameOver := false;
getMouse(mouse);
pos := mouse.h;

for i := 0 to 31 do begin
     new(stars[i]);
     stars[i].setup;
end;

for i := 1 to MAX_ROCKS do begin
     new(rocks[i]);
     rocks[i].setup;
end;

repeat
     drawStars;
     drawRocks;
     drawShip;
     repeat until getTick > tick;
     tick := tick + 1;

     { Check collision }

     for i := 1 to MAX_ROCKS do begin
          if rocks[i].hasCollided(pos) then begin

               { Explosion }

               x := rocks[i].x;
               y := rocks[i].y;

               for j := 1 to 15 do begin
                    drawStars;
                    drawRocks;
                    drawExplosion(x, y, j * 5, j);

                    repeat until getTick - tick > 3;
                    tick := getTick;
               end;

               sysBeep;
               gameOver := true;
          end;
     end;

     { Fire laser }

     if button(0) then begin
          drawLaser(pos, white);

          for i := 1 to MAX_ROCKS do begin
               drawStars;

               if rocks[i].hasDestroyed(pos) then begin
                    score := score + rocks[i].y div 10;
                    rocks[i].reset;
               end;
          end;

          drawLaser(pos, black);
     end;
until gameOver;
```

```
     { Game over }

     moveTo(60, 90);
     setBackColor(black);
     setForeColor(9);
     str := concat('Game Over. Your Score is ',
       cnvis(score), '.');
     drawString(str);
     repeat until button(0);

     for i := 0 to 31 do
          dispose(stars[i]);

     for i := 1 to MAX_ROCKS do
          dispose(rocks[i]);

     endDesk;
end.
```

Chapter 4

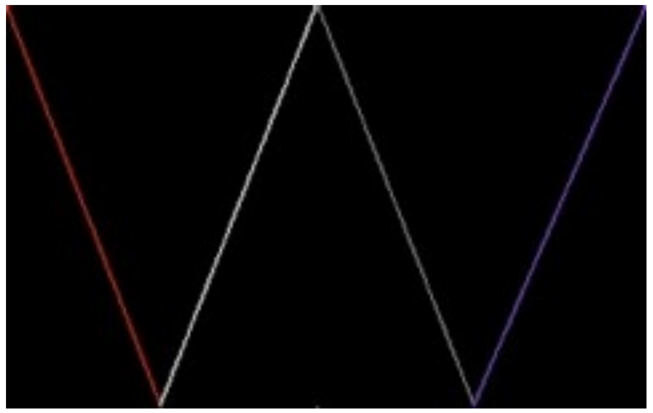
Lim Ding Wen

Lim Ding Wen started programming at 7, and he is able to code in Applesoft BASIC, GSoft BASIC, Complete Pascal, Orca/Pascal and modern languages like Objective C and JavaScript.

Lim Ding Wen became the world youngest iPhone developer when he ported his Apple IIGS program **Doodle Kids** to iPhone at the age of 9.

1

W



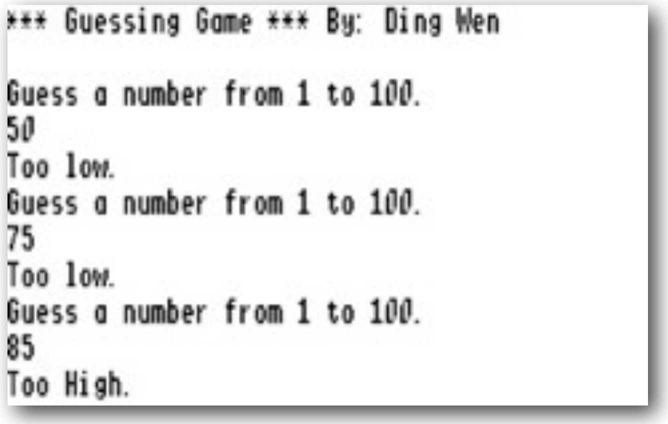
This is Ding Wen first Pascal program developed for Apple IIGS.

Instead of a simple "Hello World", he demonstrates the understanding of Super Hires coordinate system and simple QuickDraw commands.

```
Program W;
Uses Types, QuickDraw;

Begin
     graphics(320);
     hidecursor;
     clearScreen(0) ;

     moveto(0,0);
     setSolidPenPat(7);
     lineTo(77,199);

     setSolidPenPat(15);
     lineTo(155,0);

     setSolidPenPat(1);
     lineTo(232,199);

     setSolidPenPat(3);
     lineTo(319,0);

  readln;
End.
```

2

Guessing Game



```
*** Guessing Game *** By: Ding Wen

Guess a number from 1 to 100.
50
Too low.
Guess a number from 1 to 100.
75
Too low.
Guess a number from 1 to 100.
85
Too High.
```

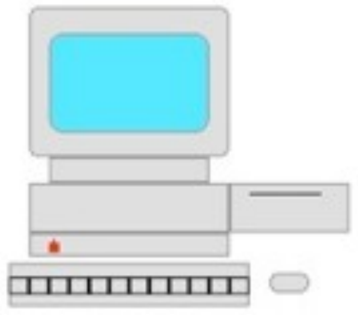
This is a classic guessing game implemented for Apple IIGS.

It shows the understanding of various programming concepts like loop and decision making. It also shows how to use the system tick to do a better random number generation.

```
program Guess;
uses Types, QuickDraw, MiscTool;

var
     ans, rand, loop, score: Integer;

begin

     { Setup }

     graphics(640);
     setRandSeed(getTick);
     hidecursor;
     writeln('*** Guessing Game *** By: Ding Wen');
     writeln;

     { Set varibles }

     loop := 0;
     score := 110;
     rand := random mod 100 + 1;

     while loop <= 10 do begin
          loop := loop + 1;
          score := score - 10;
          { Ask question }

          writeln('Guess a number from 1 to 100.');
          readln(ans);

          { Check answer }

          if ans < rand then
               writeln('Too low.')
          else begin
               if ans > rand then
                    writeln('Too High.')
               else begin
                    if ans = rand then begin
                         writeln('You Win!');
                         writeln('You had ', loop, '
                           tries.');
                         writeln('Good luck next time!');
                         leave;
                    end;
               end;
          end;
     end;

     { Write score }

     writeln('Here is your score:',score, '/100');
     readln;
end.
```

3

Apple IIGS



This program draws an Apple IIGS computer.

It demonstrate the uses of various QuickDraw drawing tools.

```
program AppleIIGS;
uses Types, QuickDraw, Events;

var
     keys, positionX, positionY: integer;
     r: rect;

begin
     positionX := 60;
     positionY := 70;

     graphics(320);
     hideCursor;

     { Draw screen outside }

     setSolidPenPat(14);
     setRect(r, 80, 37, 180, 112);
     paintRRect(r, 10, 10);
     setSolidPenPat(1);
     frameRRect(r, 10, 10);

     { Draw screen }

     setSolidPenPat(11);
     setRect(r, 90, 50, 170, 100);
```

```
paintRRect(r, 15, 15);
setSolidPenPat(1);
frameRRect(r, 15, 15);

{ Draw Apple IIGS neck }

setSolidPenPat(14);
setRect(r, 90, 112, 170, 125);
paintRect(r);
setSolidPenPat(1);
frameRect(r);

{ Draw base }

setSolidPenPat(14);
setRect(r, 80, 125, 180, 150);
paintRect(r);
setSolidPenPat(1);
frameRect(r);
setSolidPenPat(14);
setRect(r, 80, 150, 180, 162);
paintRect(r);
setSolidPenPat(1);
frameRect(r);

{ Draw logo }

setSolidPenPat(7);
setPenSize(5, 5);
moveTo(90, 155);
line(0, 0);
move(0, -1);
move(2, 0);
setPenSize(1, 1);
line(0,-1);

{ Draw keyboard }

setSolidPenPat(14);
setRect(r, 70, 165, 190, 187);
paintRect(r);
setSolidPenPat(1);
frameRect(r);

{ Draw keys }

for keys:=1 to 12 do begin
     positionX := positionX + 10;
     positionY := positionY + 10;
     setSolidPenPat(0);
     setRect(r, positionX, 172, positionY, 181);
     frameRect(r);
end;

{ Draw disk }

setSolidPenPat(14);
setRect(r, 180, 125, 240, 150);
paintRect(r);
setSolidPenPat(0);
moveTo(190, 130);
line(35,0);
setSolidPenPat(1);
frameRect(r);
```

```
     { Draw mouse }

     setRect(r, 200, 170, 220, 180);
     setSolidPenPat(14);
     paintRRect(r, 10, 10);
     setSolidPenPat(1);
     frameRRect(r, 10, 10);

     repeat until button(0);
end.
```

4

Moving Rectangle

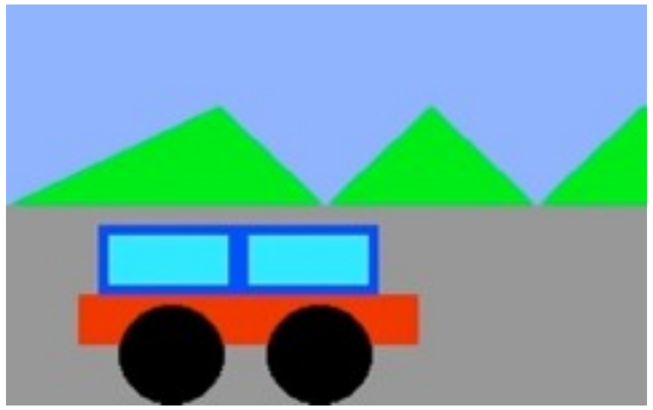
This program draws a moving rectangle.

It demonstrates animation technique.

```
program movingRect;
uses Types, QuickDraw, Events, MiscTool;

var
     x,y,speed: integer;
     tick: longInt;

begin
     x := 0;
     y := 100;
     tick := getTick;

     graphics(320);
     setRandSeed(tick);
     speed := random mod 6 + 1;

     clearScreen(0);
     hideCursor;

     { Draw rectangle }

     setPenSize(5,3);
     setSolidPenPat(7);
     moveTo(x, y);
     line(0, 0);
     repeat

          { Erase rectangle }

          setSolidPenPat(0);
          moveTo(x, y);
          line(0, 0);

          x := x + speed;

          { Draw rectangle }

          setSolidPenPat(7);
          moveTo(x, y);
          line(0, 0);

          if x > 320 then begin
               x := 0;
               sysBeep;
               speed := random mod 6 + 1;
               y := random mod 197;
          end;

          repeat until getTick > tick;
          tick := tick + 1;
     until button(0);
end.
```

5

Car



The car moves and horns!

It demonstrates the use of procedure, polygons, and complex object animation.

```
program car;
uses Types,QuickDraw,Events,MiscTool;

var
     x: integer;
     speed: integer;
     poly: handle;
     tick: longInt;
     r: rect;

{ Draw hills }

procedure drawHills;
begin
     setSolidPenPat(0);
     poly := openPoly;
     moveTo(0, 100);
     lineTo(106, 50);
     lineTo(159, 100);
     lineTo(212, 50);
     lineTo(265, 100);
     lineTo(318, 50);
     lineTo(371, 100);
     lineTo(0,100);
     closePoly;
     setSolidPenPat(10);
     paintPoly(poly);
end;

{ Draw car }

procedure drawCar;
begin
     setSolidPenPat(7);
     setRect(r, x, 145, x + 170, 170);
     paintRect(r);
     setSolidPenPat(4);
     setRect(r, x + 10, 110, x + 150, 145);
     paintRect(r);
     setSolidPenPat(0);
     setRect(r, x + 20, 150, x + 73, 200);
     paintOval(r);
     setRect(r, x + 94, 150, x + 147, 200);
     paintOval(r);
     setSolidPenPat(11);
     setRect(r, x + 15, 115, x + 75, 140);
     paintRect(r);
     setRect(r, x + 85, 115, x + 145, 140);
     paintRect(r);
end;

begin
     x := 0;
     tick := getTick;
     speed := random mod 6 + 1;

     graphics(320);
     hideCursor;
     setSolidBackPat(0);
     setSolidPenPat(13);
     setRect(r, 0, 0, 320, 100);
     paintRect(r);
     drawHills;

     repeat

          { Draw road }

          setSolidPenPat(1);
          setRect(r,0,100,320,200);
          paintRect(r);

          { Move car }

          x := x + speed;
          drawCar;

          repeat until getTick-tick > 5;
          tick := getTick;

          if x > 320 then begin
               x := 0;
               speed := random mod 6 + 1;
          end;

          if (x > 70) and (x < 90) then
               sysBeep;
     until button(0);
end.
```

6

Snow

It is snowing! Merry Christmas!

This program demonstrates the use of type and array, to generate multiple objects. This is Ding Wen's first program on multiple objects animation.

```
program snow;
uses types,quickDraw,events,miscTool;

type
     snow = record
          x: integer;
          y: integer;
          speed: integer;
     end;

var
     i, slide: integer;
     tick: longInt;
     snows: array[1..100] of snow;

begin
     tick := getTick;

     graphics(320);
     setRandSeed(getTick);
     hideCursor;
     clearScreen(0);

     { Setup snow }

     for i := 1 to 100 do begin
```

```
          snows[i].x := random mod 320;
          snows[i].y := i * 10;
          snows[i].speed := random mod 10 + 1;
     end;

     repeat
          for i := 1 to 100 do begin
               slide := random mod 11 - 5;

               { Erase snow }

               setSolidPenPat(0);
               moveTo(snows[i].x, snows[i].y);
               line(0, 0);

               { Move snow }

               snows[i].y := snows[i].y + snows[i].speed;
               snows[i].x := snows[i].x + slide;

               if snows[i].y > 190 then begin
                    snows[i].y := 0;
                    snows[i].x := random mod 320;
                    snows[i].speed := random mod 10 + 1;
               end;

               { Draw snow }

               setSolidPenPat(15);
               moveTo(snows[i].x, snows[i].y);
               line(0,0);
          end;
          repeat until getTick - tick > 5;
          tick := getTick;
     until button(0);
end.
```

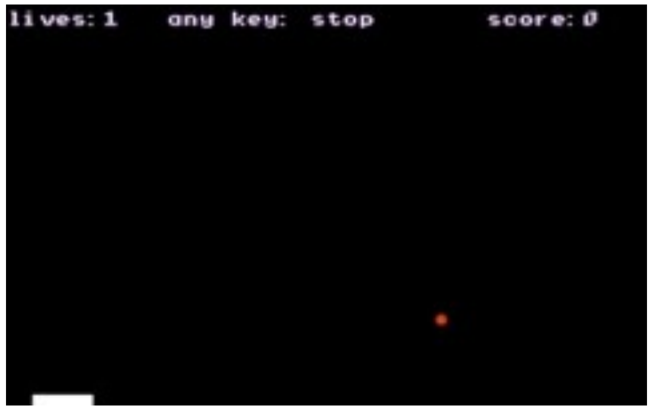
This is Ding Wen's first Object Oriented program, written with Orca/Pascal.

```
program snow;
uses common,quickDrawII,eventMgr,mscToolSet;

type
     snow = object
          x:integer;
          y:integer;
          procedure setup;
          procedure move;
          procedure draw;
          procedure erase;
     end;

var
     i:integer;
     tick:longInt;
     snows:array[1..300] of snow;

{ Setup }

procedure snow.setup;
begin
     x := randomInteger mod 319;
     y := randomInteger mod 100;
end;

{ Move snow }

procedure snow.move;
var
     slide, speed: integer;

begin
     speed := randomInteger mod 20 + 1;
     slide := randomInteger mod 11 - 5;

     x := x + slide;
     y := y + speed;

     if y > 199 then begin
          x := randomInteger mod 319;
          y := 0;
     end;
end;

{ Draw snow }

procedure snow.draw;
begin
     setSolidPenPat(15);
     moveTo(x, y);
     line(0, 0);
end;

procedure snow.erase;
begin
     setSolidPenPat(0);
     moveTo(x, y);
     line(0, 0);
end;
```

```
begin
     tick := getTick;
     startDesk(320);
     setRandSeed(getTick);
     hideCursor;
     clearScreen(0);

     { Setup snow }

     for i := 1 to 300 do begin
          new(snows[i]);
          snows[i].setup;
     end;

     repeat
          for i := 1 to 300 do begin
               snows[i].erase;
               snows[i].move;
               snows[i].draw;
          end;

          repeat until getTick - tick > 5;
          tick := getTick;
     until button(0);

     { Clear snow }

     for i := 1 to 300 do
          dispose(snows[i]);

     endDesk;
end.
```

7

Paddle



This is Ding Wen’s first graphical game!

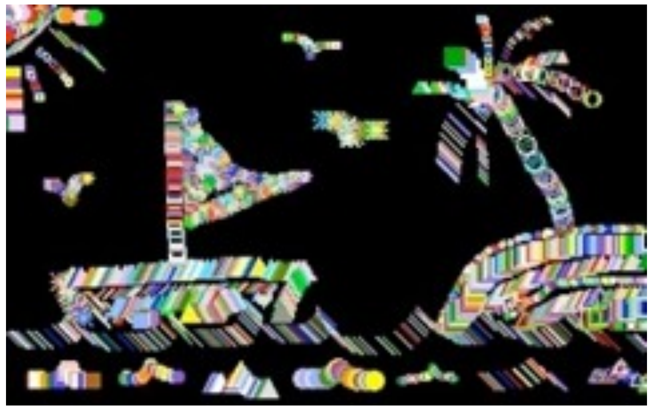
This game is modeled after the famous Pong, and it is designed and written completely by himself without external help.

```
program Paddle;
uses types,quickDraw,events,miscTool;

var
     pos: integer;
     pos2: integer;
     types: integer;
     key: integer;
     score: integer;
     life: integer;
     event: eventRecord;
     r: rect;
     mouse: point;
     tick: longInt;
     x: integer;
     y: integer;
     x2: integer;
     y2: integer;
     sx: integer;
     sy: integer;
     gameOver: boolean;
     ball: rect;

{***Erase All***}

procedure eraseAll;
```

```
begin
     setSolidPenPat(0);
     setRect(r, pos, 195, pos2, 200);
     paintRect(r);
     setRect(r, x, y, x2, y2);
     paintOval(r);
end;

{***Draw paddle***}

procedure drawPaddle;
begin
     getMouse(mouse);

     pos := mouse.h;
     pos2 := pos + 30;

     if pos < 0 then begin
          pos := 0;
          pos2 := 30;
     end;

     if pos2 > 320 then begin
          pos := 290;
          pos2 := 320;
     end;

     setSolidPenPat(15);
     setRect(r, pos, 195, pos2, 200);
     paintRect(r);
end;

{***Draw ball***}

procedure drawBall;
begin
     setSolidPnPat(7);
     setRect(ball, x, y, x2, y2);
     paintOval(ball);
end;

{***Wait***}

procedure wait(time:longInt);
begin
     repeat until getTick-tick > time;
     tick := getTick;
end;

{***Check collision***}

procedure checkCollision;
begin
     if (x > pos) and (x < pos2) and (y2 > 195)
       and (y < 200) then begin
          sy := -sy;
          score := score + 10;
          sysBeep;
     end
     else if y > 220 then begin
          life := life - 1;
          x := 160;
          y := 100;
          x2 := 165;
          y2 := 105;
          sysBeep;
     end;

     if (x < 0) or (x2 > 320) then begin
          sx := -sx;
          sysBeep;
     end;

     if (y < 10) then begin
          sy := -sy;
          sysBeep;
     end;
end;

{***Write numbers***}

procedure writeScore;
begin
     moveTo(0, 10);
     writeln('lives:', life);
     moveTo(80, 10);
     writeln('any key: stop');
     moveTo(240, 10);
     writeln('score:',score);
end;

{***Setup***}

procedure setup;
begin
     life := 2;
     tick := getTick;
     x := 160;
     y := 100;
     x2 := 165;
     y2 := 105;
     gameOver := false;
     types := keyDownMask;
     sx : =5;
     sy := 5;
end;

begin
     setup;

     graphics(320);
     hideCursor;
     clearscreen(0);
     setBackColor(0);
     setForeColor(12);
     moveTo(0,100);

     repeat
          wait(1);
          eraseAll;

          {***Move ball***}

          x := x + sx;
          y := y + sy;
          x2 := x2 + sx;
          y2 := y2 + sy;
          {***Draw objects***}

          drawPaddle;
          drawBall;
          writeScore;
          if getNextEvent(keyDownMask, event) then
               if event.what = keyDownEvt then halt;

          checkCollision;

          {***Game over!!!***}

          if life=-1 then gameOver := true;
     until gameOver;

     setForeColor(7);
     moveTo(120, 100);
     writeln('GAME OVER!');
     repeat until button(0);
     sysBeep;
end.
```

8

Doodle Kids



Ding Wen’s first event driven program - a full screen painting program for kids that draw random shapes and colors!

Press Delete key to clear the picture and Space key to do color cycling! Press ESC key to quit the program.

This program is written for his younger sisters. It is designed and written completely by himself.

Doodle Kids is now available for iPhone, iPad and Android!

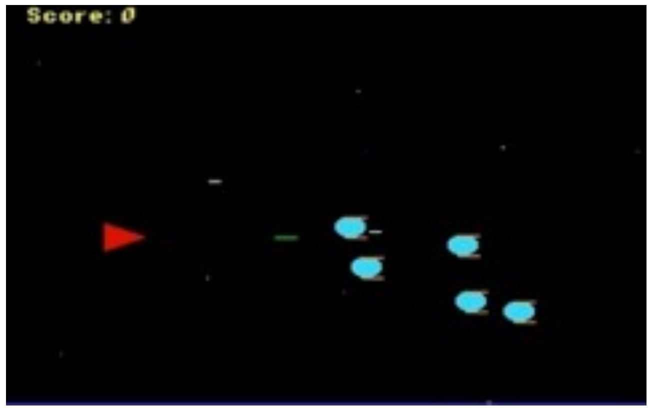
```
program DoodleKids;
uses Types, Quickdraw, Events, miscTool;

var
     etypes, color : integer;
     key, size, shape, x, y, i : integer;
     event : eventRecord;
     mouse : point;
     tri : handle;
     rec : rect;
     animate, draw : boolean;
     tick : longInt;

{ Color cycle }

procedure cycle;
begin
     if animate then begin
          color := getColorEntry(0, 1);
          for i := 1 to 14 do
               setColorEntry(0, i,
          getColorEntry(0, i + 1));
          setColorEntry(0, 15, color);
     end;
end;
```

```
{ Start drawing }

procedure startDraw;
begin
     if draw then begin
          size := random mod 9 + 3;
          getMouse(mouse);
          x := mouse.h;
          y := mouse.v;

          setSolidPenPat(random mod 15 + 1);

          if shape < 5 then
               setRect(rec, x - size div 2,
                 y - size div 2, x + size div 2,
                 y + size div 2);

          if (shape > 6) and (shape < 9) then begin
               tri := openPoly;
               moveTo(x - size div 2, y + size div 2);
               line(size, 0);
               lineTo(x, y - size div 2);
               closePoly;
          end;

          { Draw shape }

          case shape of
               1: paintOval(rec);
               2: paintRect(rec);
               3: frameOval(rec);
               4: frameRect(rec);
               5: begin
                    moveTo(x + size div 2, y + size div 2);
                    lineTo(x - size div 2, y - size div 2);
               end;
               6: begin
                    moveTo(x - size div 2, y + size div 2);
                    lineTo(x + size div 2, y - size div 2);
               end;
               7: paintPoly(tri);
               8: framePoly(tri);
               9: begin
                    moveTo(x + size div 2, y + size div 2);
                    lineTo(x - size div 2, y - size div 2);
                    moveTo(x - size div 2, y + size div 2);
                    lineTo(x + size div 2, y - size div 2);
                    moveTo(x - size div 2, y);
                    line(size, 0);
                    moveTo(x,y - size div 2);
                    line(0,size);
               end;
          end;
     end;
end;

begin
     etypes := keyDownMask + mDownMask + mUpMask;
     tick := getTick;

     { Setup }

     graphics(320);
     clearScreen(0);
```

```
while true do begin
     startDraw;
     if getTick - tick > 9 then begin
          tick := getTick;
          cycle;
     end;

     if getNextEvent(etypes,event) then begin
          case event.what of

               { Keys }

               keyDownEvt: begin
                    key := event.message;

                    case key of
                         27: halt;
                         32: begin
                              if not animate then
                                   animate := true
                              else animate := false;
                         end;
                         127: clearScreen(0);
                    end;
               end;

               { Mouse }

               mouseDownEvt: begin
                    shape := random mod 9 + 1;
                    draw := true;
               end;

               mouseUpEvt: draw := false;
          end;
     end;
end;
end.
```

9

Invader War



Ding Wen's first space game is written in Orca/Pascal.

Lim Ding Wen shows his game Invader War.

Invader War is also available for iPhone and iPad.

```
program InvaderWar;
uses Common,QuickDrawII,EventMgr,MscToolSet;

type
     star = object
          x,y,speed : integer;
          procedure setup;
          procedure draw;
     end;

     invader = object
          x,y,bx,by,speed : integer;
          fshoot : boolean;
          procedure setup;
          procedure draw;
          procedure bang;
          procedure fire;
     end;

var
     i,posX,posY,etypes,bulletX,bulletY,score: integer;
     stars: array[1..10] of star;
     invaders: array[1..5] of invader;
     tick,key: longInt;
     event: eventRecord;
     shippoly,fire: polyHandle;
     gameOver,shoot: boolean;
     str: string;

procedure explode;
var
     a,ex,ey: integer;
     r: rect;
     str: string;

begin
     sysBeep;

     for a := 1 to 15 do begin
          setSolidPenPat(a);
          ex := randomInteger mod 20 + (posX - 30);
          ey := randomInteger mod 20 + (posY - 20);
          setRect(r,ex,ey,ex + 20, ey + 20);
          paintOval(r);
          tick := getTick;

          repeat until getTick - tick > 1;
     end;

     moveTo(75, 100);
     str := concat('Game Over. Score:', cnvis(score));
     drawString(str);
     repeat until button(0);
     gameOver := true;
end;

procedure drawShip;
var
     mouse : point;

begin
     setSolidPenPat(0);
     paintPoly(shippoly);
     paintPoly(fire);

     getMouse(mouse);
     posX := mouse.h;
     posY := mouse.v;

     if posX < 20 then posX := 20;
     if posY < 20 then posY := 20;
     if posY > 195 then posY := 195;

     shippoly := openPoly;
     moveTo(posX, posY);
     lineTo(posX-20, posY-7);
     line(0,14);
     lineTo(posX, posY);
     closePoly;

     setSolidPenPat(7);
     paintPoly(shippoly);

     fire := openPoly;
     moveTo(posX-20, posY-7);
     line(0, 14);
     lineTo(posX-25, posY);
     lineTo(posX-20, posY-7);
     closePoly;
     setSolidPenPat(randomInteger mod 2 * 6);
     paintPoly(fire);
end;

procedure bangShip;
var
     i: integer;

begin
     for i := 1 to 15 do
     if (invaders[i].x < posX) and
       (invaders[i].x > posX - 35) and
       (invaders[i].y < posY + 7) and
       (invaders[i].y > posY - 12) then explode;
end;

procedure invader.setup;
begin
     x := randomInteger mod 100 + 220;
     y := randomInteger mod 185 + 15;
end;

procedure invader.draw;
var
     r: Rect;

begin
     speed := randomInteger mod 3 + 1;

     setSolidPenPat(0);
     setRect(r,x,y,x+15,y+10);
     paintOval(r);
     moveTo(x + 5,y);
     line(10,0);
     moveTo(x + 5,y + 10);
     line(10,0);

     x := x - speed;
     if posY < y + 2 then y := y - 1;
     if posY > y then y := y + 1;

     if x < 0 then begin
          x := 320;
          y := randomInteger mod 200;
     end;

     setSolidPenPat(11);
     setRect(r, x ,y, x + 15, y + 10);
     paintOval(r);
     setSolidPenPat(6);
     moveTo(x + 5,y);
     line(10,0);
     moveTo(x + 5,y + 10);
     line(10,0);
end;

procedure invader.bang;
var
     r: Rect;

begin
     if shoot then begin
          setSolidPenPat(0);
          moveTo(bulletX, bulletY);
          line(-10, 0);
          bulletX := bulletX + 3;
          setSolidPenPat(10);
          moveTo(bulletX, bulletY);
          line(-10, 0);

          if (bulletY > y - 2) and (bulletY < y + 12)
            and (bulletX > x - 1)
            and (bulletX < x + 16) then begin
               setSolidPenPat(15);
               setRect(r, x - 10, y - 10, x + 20, y + 20);
               paintOval(r);
               score := score + (320 - (x - posX)) div 10;
               tick := getTick;
               repeat until getTick - tick > 5;

               setSolidPenPat(0);
               paintOval(r);

               x := 320;
               y := randomInteger mod 200;
               moveTo(bulletX, bulletY);
               line(-5,0);
               shoot := false;
          end;

          if bulletX > 330 then shoot := false;
     end;
end;

procedure invader.fire;
begin
     if fshoot then begin
          setSolidPenPat(0);
          moveTo(bx, by);
          line(5,0);
          bx := bx - 5;
          setSolidPenPat(14);
          moveTo(bx, by);
          line(5,0);
          if (bx < posX) and (bx > posX - 25) and
            (by < posY + 5) and (by > posY - 5) then
               explode;

          if bx < -5 then fshoot := false;
     end;

     if not fshoot then begin
          if (y + 3 > posY - 5) and (y + 3 < posY + 5) then
            begin
               bx := x;
               by := y + 3;
               fshoot := true;
          end;
     end;
end;

procedure star.setup;
begin
     x := randomInteger mod 320;
     y := randomInteger mod 185 + 15;
     speed := randomInteger mod 6 + 1;
end;

procedure star.draw;
```

```
begin
     setSolidPenPat(0);
     moveTo(x,y);
     line(0,0);

     x := x - speed;
     if x < 0 then begin
          x := 320;
          speed := randomInteger mod 6 + 1;
     end;

     setSolidPenPat(i);
     moveTo(x,y);
     line(0,0);
end;

begin
     etypes := keyDownMask + mDownMask;
     tick := getTick;

     startDesk(320);
     hideCursor;
     clearScreen(0);
     setForeColor(9);
     setBackColor(0);
     setRandSeed(tick);

     for i := 1 to 10 do begin
          new(stars[i]);
          stars[i].setup;
     end;

     for i := 1 to 5 do begin
          new(invaders[i]);
          invaders[i].setup;
     end;

     repeat
          for i := 1 to 10 do stars[i].draw;
          for i := 1 to 5 do invaders[i].draw;
          for i := 1 to 5 do invaders[i].bang;
          for i := 1 to 5 do invaders[i].fire;

          drawShip;
          bangShip;

          if getNextEvent(etypes,event) then begin
               case event.eventWhat of
                    keyDownEvt: begin
                         key := event.eventMessage;
                         if key = 27 then
                              gameOver := true;
                    end;

                    mouseDownEvt: begin
                         if not shoot then begin
                              shoot := true;
                              bulletX := posX;
                              bulletY := posY;
                         end;
                    end;
               end;
          end;
          moveTo(10,10);
          str := concat('Score: ',cnvis(score));
          drawString(str);
          repeat until getTick > tick;
          tick := tick + 1;
     until gameOver;

     for i := 1 to 15 do
          dispose(stars[i]);

     for i := 1 to 5 do
          dispose(invaders[i]);

     endDesk;
end.
```

5

What’s Next?

This is the end of the book, but I hope that this is just the beginning of the journey!

More programming languages

Although Pascal or BASIC are both great programming languages for learning, they are no longer widely used. So what programming languages should you learn?

Here are some possibilities:

- JavaScript: the language used for the web. You can use JavaScript and HTML5 to create a web app.
- ActionScript: the language used for Flash development. You can use ActionScript to create Flash application.
- Objective-C: the language used for iOS and Mac development. You can use Objective-C to create iPhone, iPad and Mac applications.

No matter what languages you choose, you already have the knowledge to learn yet another one!

Lim Ding Wen and his iPhone app.

Apple II

The Apple II is a set of 8-bit home computers, one of the first highly successful mass-produced microcomputer products, designed primarily by Steve Wozniak, manufactured by Apple Computer (now Apple Inc.) and introduced in 1977.

词汇相关表词汇

将相关词汇拖到这里

索引

查找词汇

Apple IIGS

Apple IIGS arrived on 15th September 1986 and it is the final computer in Apple II series. It is a 16 bit computer and has both excellent graphics and music capabilities at that time.

Related Glossary Terms

将相关词汇拖到这里

Index 查找词汇

BASIC Programming Language

BASIC is a family of general-purpose, high-level programming languages whose design philosophy emphasizes ease of use - the name is an acronym from **Beginner's All-purpose Symbolic Instruction Code**.

词汇相关表词汇

将相关词汇拖到这里

索引 查找词汇

Complete Pascal

Complete Pascal (formerly TML Pascal II) is a native code Pascal compiler for the Apple IIGS which provides an elegant yet powerful programming environment that lets you write, edit, compile and run applications and desk accessories using the Apple IIGS Toolbox with incredible speed and simplicity

GSoft BASIC

GSoft BASIC let anyone write BASIC programs specific to the Apple IIGS and using its toolbox and other native IIGS features.

词汇相关表词汇

将相关词汇拖到这里

索引 查找词汇

Object Pascal

Object Pascal is an extension of the Pascal language that was developed at Apple Computer by a team led by Larry Tesler in consultation with Niklaus Wirth, the inventor of Pascal.

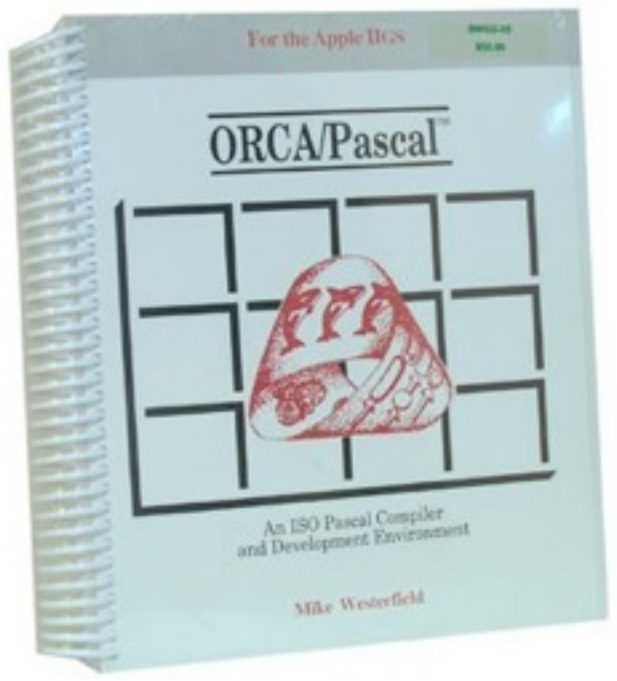
词汇相关表词汇

将相关词汇拖到这里

索引 查找词汇

Orca/Pascal

ORCA/Pascal is a complete Pascal language development system for the Apple IIGS computer.



词汇相关表词汇

将相关词汇拖到这里

索引 查找词汇

Pascal Programming Language

Pascal is an influential imperative and procedural programming language, designed in 1968/9 and published in 1970 by Niklaus Wirth as a small and efficient language intended to encourage good programming practices using structured programming and data structuring.

词汇相关表词汇

将相关词汇拖到这里

索引 查找词汇