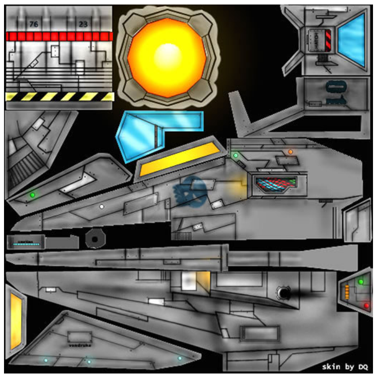

Student Manual:

3D Game Programming with Phrogram™

Sample Middle School Module for

Teaching with Phrogram

Prepared by John Osborne

Windermere Ranch Middle School, San Ramon, CA

Edited by David Witus

The Phrogram Company

FOREWORD

The minimum requirements for this Module are that the student has access to a PC with Phrogram™ installed.

This Student Manual is based upon the electronic book, *"Learn to Program with Phrogram!"* published by Addison Wesley Professional and available online at *http://safari.oreilly.com/9780321496836*.

John Osborne
Teacher, Advanced Technology
Windemere Ranch Middle School
925-479-7400
josborne@srvusd.net

INTRODUCTION

3D game programming sounds like a lot of fun, right? You're thinking that you'll be able to make the next version of Halo.

It's not a dream. It might be far off but you're going to be a lot closer after this module.

Whenever you see the symbol ☞, it means that you have to do something on the computer or in your binder. Also, all the exercises have to be done.

PRE-TEST

The Pre-Test shows you what you already know and lets you see what you going to know after you complete this module. It does not count on your grade.

Answer these questions on a separate piece of paper in your binder. You do not have to repeat the question; just write the number and the answer.

True/False

1. Programming is about frogs
2. Programming will turn you into a frog
3. You are already a frog
4. Games can be created in one class period.
5. Games are more fun in 2D than in 3D.
6. Games can be written in many different languages.
7. Computers use binary, octal, and hexadecimal.
8. Computers like English better than anything else.
9. Algebra is used in introductory computer programming.
10. 3D game programming is too complex for a Middle School student.

RUNNING THE DEMO GAMES

☞Start Phrogram by double-clicking on the frog icon on your desktop. Alternatively, you can go to Start/Programs/Phrogram.

☞Open C:\My Documents\My Phrogram Files\Games and Toys (2D).

You'll see a number of games.

☞Choose the Asteroids game.

☞To run the game, click Code/ Run Program, or hit F5, or click the green triangle at the top.

If you try some of the others, you'll see that they are pretty simple, not very challenging, and only in two-dimensions. Then,

☞Navigate to C:\My Documents\My Phrogram Files\Games and Toys (3D).

Experiment with the game found there. It's definitely better looking. Spend only as much time as you need to get the idea of how to open and run a Phrogram game.

☞In your binder, write the question "What makes this game look like 3D?" and your answer.

CREATING A NEW PROGRAM

Begin a new program with

☞Click File/ New.

Now we get into one of the nitty gritty aspects of programming - details. You have to give line by line details describing what objects you want to see on the screen and how you want them to perform. First we have to name our Sprite object. Then, we find a picture for the Sprite object. Then, we decide where to show the Sprite on the screen. This is a lot of typing but I'll save you much of the typing effort so you can concentrate on learning.

The words "MyNewProgram" will be highlighted in blue.

☞Change "MyNewProgram" to "MyUFO."

☞Open the file "MyUFO.txt" using Notepad.

☞Copy and paste the source code into Phrogram.

Now look at the instructions and I'll explain what they do.

`Define myUFO As Sprite`	We tell Phrogram that we are going to have a Sprite object and would like to call it myUFO.

`myUFO.Load("ufo.gif")`	We tell Phrogram to use the "ufo.gif" picture for the Sprite object
`myUFO.MoveTo(60, 60)`	We give Phrogram the coordinates for where to show the picture on the screen.
`myUFO.Show()`	And finally, we tell Phrogram to make the picture visible.

By the way …

Upper and lower case letters are very important in programming. "M" and a "m" are as different to a computer as "M" and "g" are. You don't have to learn the Camel Casing rules about when to use capital letter but you have to be careful when typing to use the correct letter.

☞Run your program.

What did you see? We did all that work just to have a picture of a flying saucer sitting in the corner of the screen! Now think about Pocket Tanks and try to imagine how much programming work went into making it fun to play. That's why there are so many jobs available making games and they pay very well.

EXERCISE 1:

Change the line of code that places the UFO on the screen. The screen is 500 by 400 pixels.

. Place the UFO in the bottom right corner.

. Place it in the bottom left corner.

. Display a different UFO. You can look at other pictures in the c:\My Document\My Phrogram Files\Media\Images folder. Find a different UFO and use it in your program.

. Save your program as Exercise1.

Adding Sound and Moving the UFO

OK, let’s make this thing a little more like a game. First, let’s add the sound of a UFO using the command. Then let’s add some commands to move the UFO around the screen.

☞Begin a new program.

☞Change “MyNewProgram” to “MyUFO2.”

☞Open the file “MyUFO2.txt” using Notepad.

☞Copy and paste the source code into Phrogram.

☞Run your program.

What did you see? The UFO should have moved down the screen without you having to change the program each time. Now look at the new instructions and I’ll explain what they do.

`PlaySound(“eerie.wav”)`	We tell Phrogram to play a sound file. Can you tell me the name of the file?
`Define ufoY As Integer`	We tell Phrogram that we want ufoY to be a number. Remember, we have to tell the computer everything.

`For ufoY = 51 to 150`
`   Delay(49)`
`   myUFO.MoveTo(50,ufoY)`
`Next`
	This is called a loop. We’ll talk more about loops later. We want the computer to keep doing something while looping. The something is “Delay(49) and “myUFO.MoveTo(50,ufoY)”.
	
	The Delay(49) command tells the computer not to do anything for 49 milliseconds (not a very long time to us but an eternity to a computer).
	
	The myUFO.MoveTo(50,ufoY) does what you did earlier in Exercise 1. The first time, it says myUFO.MoveTo(50,51). The next time, it says myUFO.MoveTo(50,52), and so on until it gets to myUFO.MoveTo(50,150).

Programming Concepts

Computers are actually pretty dumb. They only have the ability to add and subtract. Through very clever engineering, they have been made powerful enough to understand your commands in a structured language.

In the old days (only a few decades ago!), most people who programmed computers did so with machine language, a very difficult task but necessary to write an operating system. COBOL was used by the application programmers. COBOL is pretty much gone. Today, there are many programming languages but you probably hear of Visual Basic, C, and Java the most. Phrogram is very similar to Visual Basic. It is somewhat like Java too, so once you master Phrogram, you will be able to more quickly learn those languages. C is closer to machine language and has been replaced by later versions C++ and C#.

Look for the word “Main” on the second line of the program you just created. This is the start of a method. We’ll learn more about methods much later. For now, think of Main as a great big road sign saying START HERE!

Exercise 2

Pretend that you are a computer. Ask your partner to program you to open your Binder using the following commands. If your partner gives you a command you do not understand, do nothing.

- Move whole right arm forward one inch
- Move whole left arm forward one inch
- Move whole right arm backward one inch
- Move whole left arm backward one inch
- Raise whole right arm one inch
- Raise whole left arm one inch
- Raise left hand one inch
- Raise right hand one inch
- Lower whole right arm one inch
- Lower whole left arm one inch
- Lower left hand one inch

- Lower right hand one inch
- Right hand grasp
- Left hand grasp

Debugging Your Programs

For most programmers, programs rarely work on the first try. Typically, you finish writing your code and you get what are known as "compilation" errors – the code will not "compile" into something that can be understood by computers at the binary "byte" level – the level of 1's and 0's that computers ultimately understand.

How to find and fix compilation errors is not that easy to teach or to understand right away. You will gradually develop this skill with practice. Your ability to "debug" your programs – find the problems and fix them - is essential to your ability to becoming good at programming.

Most experienced programmers write the code in their program one step at a time, making only one small change, testing it, then moving on to the next small change, and so on, until they complete the project. If your program doesn't work after you change it, it will be easier to resolve the problem if you haven't made a few more – or a lot more changes – since the change you made that caused the problem. Remember this when you move on to larger game projects. In this class, you'll be learning one thing and then testing it. That's why you started with showing a UFO that did not do much. When you finish this module, you'll have a fun game you can run on any computer.

Exercise 3

1. On line 8 of your MyUFO2, change the line *myUFO.Show()* to *myUFO.Show*. You are removing the (). Run your program. Find the error message. Use it to find and fix the program.
2. Open the file Exercise3.txt. Copy and paste the code into a new program. Run the program. Use the error messages to find and fix the errors. When you are asked to save it, name the program Exercise3.
3. Run the Exercise3 program. Where did the UFO go? There was no error message. Write this question "How would you know something was wrong?" in your binder and write your answer.

Variables, Loops, and Keywords

In the MyUFO2 program, you used a variable (ufoY) and a loop (For … Next). Variables hold values. Any time you want to keep score, you will need variables. Loops are a great way to save typing. Imagine having to type over 100 lines of code to move the UFO from 50,50 to 50,150. Let's use loops to make the UFO go up as well as down.

☞Open the file while.txt.

☞Copy and paste the code into a new program.

☞Name the program While.

☞Run your program.

Whoops! Why did you get a syntax error message? It turns out that programming languages reserve a few words for themselves. They have special meaning and we cannot use them for our names. These are called keywords. Keywords show up in blue on the screen.

☞Write in your binder how many keywords you can see in the program

Use MyWhile as a program name and your program should run. The UFO goes down to 50 , 150; then it goes back up to 50 , 50. The keywords While and End While give you another way of doing loops. You're telling Phrogram to do something while a condition exists. In this program, the condition is when ufoY is greater than 50, keep doing something. The something is the same as before with one important addition. We reduced the value of ufoY by 1 each time the loop runs. After 100 loops, ufoY will be back to 50. The condition no longer exists and the While loop stops.

☞Open the file while2.txt.

☞Copy and paste the code into a new program.

☞Name the program MyWhile2.

☞Run your program.

This program added another variable ufoX to use for the x-axis coordinate. It added 1 to ufoX until ufoX reached 150. This caused the UFO to move to the right.

Exercise 4

1. Change your MyWhile2 program to move the UFO all the way across the screen, but not off of it. Save your program as Exercise4.
2. Change your MyWhile2 program to move the UFO all the way down the screen, but not off of it.
3. Add a loop to move the UFO back to the top after it reaches the bottom right of the screen. Save your program.

Methods

Methods provide a way to take a set of program steps and combine them into a larger step, so you can reuse subsets of your program without copying and pasting. So far, we have only used one method. By creating methods, you avoid creating a lot of duplicate code. This makes your program smaller and easier to maintain. This saves you a lot of tedious typing.

☞Open the file methods.txt.

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Run your program.

Look at the Main method. It’s a lot shorter now. We added another method to the program to do all the work of moving the UFO. We’ll use this technique later to handle the game player’s keyboard instructions to move the UFO.

Size of Screen

Phrogram can tell us the size of the screen. This is important when the game player wants to make the screen bigger without having to call us in to change the program. There are two built-in methods we can use: `ScreenWidth()` and `ScreenHeight()`.

☞Open the file screensize.txt.

☞Copy and paste the code into the Main method.

☞Name the program ScreenSize.

☞Run your program.

It did not do anything, right? We have to learn one more thing before we can use it.

IF

What is the biggest word in the word? The answer is if. In our lives, if lets us make choices. To a computer, if gives the ability to do so many things that are impossible without it.

☞Open the file if.txt.

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Run your program.

Whoa! Now we're getting somewhere. The UFO bounces around the screen just like it was supposed to all along. And we owe it all to if. If lets us make a decision,

"if the UFO is at right edge of the screen, then make it go left," or "if the UFO is at bottom edge of the screen, then make it go up."

When you talk, you might say if the time is greater than the end of the period then leave for the next class otherwise stay here. In Phrogram, If statements are coded almost the same way. We use ">" for greater than. We use a variable for the time and the end of the class. We use "else" for "otherwise."

```
If time > EndOfClassPeriod Then
     Leave for next class
Else
     Stay here
End If
```

Also, note that every If needs an End If just like every Method needs an End Method and every Program needs an End Program.

We added a couple of other things to the program that we can use later. We added two variables Xaxis and MoveX into Xaxis. Phrogram does not let us write “Add MoveX to Xaxis.” Instead, we write “Xaxis = Xaxis + MoveX.”

We also multiplied a variable. In this case, we multiplied MoveX times a minus 1. This changes the variable to a negative number. Now each time we add MoveX, we’re adding a negative number.

It takes a little getting used to but once you get used to it, it will be hard to think any other way. It will also give you a head start in Algebra.

Exercise 5

1. Make the program run twice as long as it does now.
2. When the Xaxis gets to 150 make the UFO skip 150 pixels to the right.
3. When MoveX is negative, make the UFO skip 150 pixels to the left.

Keyboard Controls

All games have keyboard controls, so let’s get started.

☞Open the file keyboard1.txt.

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Run your program.

So, we see that we can capture the keyboard with the IsKeyDown() method. Inside the () and between quotes, place the key you want to capture.

Exercise 6

1. Change the keyboard1 program to use the “a” key for left, the “s” key for right, the “w” key for up, and the “z” key for down. Name it exercise6.
2. Change the exercise6 program to move the UFO back to 50,50 when escape has been entered.

3. Change the exercise6 program to move the UFO twice as fast/slow when the Page Up/ Page Down keys are pressed.

PROGRAM PRACTICES

Until now, we have been “hard coding” numbers into our program. Good programming practices call for using a variable in place of a number.

☞Open the file keyboard2.txt.

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Run your program.

Does it do anything different than keyboard1? Now, if you had to do exercise 6 all you would to do is add the following code:

```
Define Faster As Integer = 3
If IsKeyDown("PageUp") Then
MoveX = Faster + MoveX
End If
If IsKeyDown("PageDown") Then
MoveX = MoveX - Faster
End If
```

EXERCISE 7

1. Add the code above to the keyboard3 program. Name it exercise7.
2. Change the program to move the UFO three times as fast/ slow when the Page Up/Page Down keys are pressed.
3. Change the program to move the UFO ten times as fast/ slow when the Page Up/Page Down keys are pressed.

Now, we’re going to take advantage of methods.

☞Open the file keyboard3.txt.[1]

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Run your program.

The program works exactly like it did before. Look at the Main method. By reading the code and comments, you know what the program was designed to do. The difference is in the organization of the code.

Comments are made by starting the line with //. Phrogram will turn all characters green to tell you that everything on this line is a comment.

Maintenance of source code is one of the biggest costs in program development. It is hard to analyze a program and predict what it was designed to do. It is extremely hard to make changes to a program developed by someone else unless they did a good job with comments and organization. If they have, you do not have to wade through line after line of code to try to guess what the programmer was thinking. When you want to make a change, you can make it in one place instead looking at every line.

☞Write this question and your answer in your binder. What could go wrong with relying on comments?

☞Open the file keyboard5.txt.

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Read the comments and Run the program.

[1] c

☞Write this question and your answer in your binder. What was wrong with this program?

EXERCISE 8

1. Open the exercise5 program you created. Re-organize the code into at least one additional method. Save it as exercise8.
2. Modify the exercise8 program with comments.
3. Use variables in place of hard-coded numbers in the exercise8 program.
4. Ask the Teacher to review your program. He will look for plentiful comments, no hard-coded numbers, and a streamlined Main method.

UFO PONG

This is it. We'll complete a game of pong.

☞Open the file ufopong.txt.

☞Copy all the code.

☞In Phrogram, do Edit/ Select All.

☞Paste the code into Phrogram replacing everything.

☞Run the program.

Look at the source code and notice the organization. There are several new instructions in this program that you'll want to learn how to use.

Instruction	Comment
`Alert`	A nice way to display a message on the screen.
`myUFO.UnLoad()`	We've used Load, now let's use UnLoad. The UFO is going to start from the middle of the screen and when it runs off the edge, we UnLoad it and Load it in the middle.
`LeftScoreDisplay.Load (LeftScoreValue +`	This is a handy way to build a file name. For the first point, LeftScoreValue will equal 1 so the file name will

`".gif")`	be 1.gif. Look in the images folder and you'll see files for 0 through 9.
`ClearBackground(Black )`	Paints screen the color specified.
`If … And`	The If statement can have multiple conditions.
`If myUFO.Intersects(LeftPaddle) Then`	Uses the property, Intersects, to tell us if the UFO is in the same space as LeftPaddle.

Spend a little more time looking at the code and learning how it works. This is an excellent way to learn how to write code. New programmers are frequently assigned to study existing code before being given their own assignment.

Post Test

Write the number of the question and your answer in your binder. Do not answer on this page.

True/ False

1. You have to give line by line details.
2. 3D uses shading and lighting to give the impression of 3D.
3. Programming is too hard for the average Middle School student.
4. The instruction *Define myUFO As Sprite* will display a UFO on the screen.
5. The instruction *myUFO.Load("ufo.gif")* will display a UFO on the screen.
6. While … Next While is an example of a loop.
7. For … Next is an example of a loop.
8. Variables are used to hold the score in a game.
9. Comments are a lot of extra work that won't help me.
10. There is a Method in every program.
11. Machine language is a lot easier to program.

Fill in the blank

12. To display the UFO near the middle of the screen, use the instruction myUFO.MoveTo(_____, ______).
13. My programs rarely work on the ______ try.
14. Comments are created by starting a line with ______.
15. How many complex programming languages can you name _________________, _____________, _______________. ________________, __________________.
16. Every program has a _______ Method.
17. Program, Method, If, and While all require a _________________________.
18. If moveX < 0 __________ moveX = moveX * -1.
19. To get the screen size, use ScreenWidth() and ___________________.

20. The computer reserves certain words to itself. These are called ___________.

21. Game Programming has____ jobs.

22. To add 1 to moveX, ___________ = ____________ + __________.

Answer with a complete sentence.

23. What is wrong with relying on comments in an existing program?

____
____
____
____

24. Why are there so many jobs in game programming?

____
____
____
____

25. Why is maintenance such a big expense?

____
____
____

26. What is the advantage of using hard-coded number instead of variables?

____
____
____

27. Extra Credit: How does the computer understand complex programming languages like Visual Basic, Java, C++, or C#?

____
____
____
____

28. Extra credit: Why are we fascinated with games?

____
____
____

Phrogram

Samples & Exercises

While2

```
Define myUFO As Sprite

myUFO.Load("ufo.gif")

myUFO.MoveTo(60,60)

myUFO.Show()

PlaySound("eerie.wav")

Define ufoY As Integer

Define ufoX As Integer

For ufoY = 51 To 250

Delay (49)

myUFO.MoveTo(60, ufoY)

Next

ufoX = 50

PlaySound("eerie.wav")

While ufoX < 150

Delay (49)

myUFO.MoveTo(ufoX, ufoY)

ufoX = ufoX + 1

End While
```

Keyboard 1

```
Program Keyboard1
Method Main()
Define myUFO As Sprite
Define xAxis As Integer = 290
Define Yaxis As Integer = 225
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()
While Not IsKeyDown("Escape")
Delay(10)
If IsKeyDown("Right") Then
Xaxis = Xaxis + 3
End If
If IsKeyDown("Left") Then
Xaxis = Xaxis - 3
End If
If IsKeyDown("Up") Then
Yaxis = Yaxis - 2
End If
If IsKeyDown("Down") Then
Yaxis = Yaxis + 2
End If
myUFO.MoveTo(Xaxis, Yaxis)
End While
End Method
End Program
```

Keyboard 2

```
Program Keyboard2
Method Main()
Define myUFO As Sprite
Define xAxis As Integer = 290
Define Yaxis As Integer = 225
Define MoveX As Integer = 3
Define MoveY As Integer = 2
Define Xstart As Integer = 50
Define Ystart As Integer = 50
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()
While Not IsKeyDown("Escape")
Delay(10)
If IsKeyDown("Right") Then
Xaxis = Xaxis + MoveX
End If
If IsKeyDown("Left") Then
Xaxis = Xaxis - MoveX
End If
If IsKeyDown("Up") Then
Yaxis = Yaxis - MoveY
End If
If IsKeyDown("Down") Then
Yaxis = Yaxis + MoveY
End If
myUFO.MoveTo(Xaxis, Yaxis)
End While
myUFO.MoveTo(Xstart, Ystart)
End Method
End Program
```

Keyboard 3

```
Program Keyboard3
Method Main()
Define myUFO As Sprite
Define xAxis As Integer = 290
Define Yaxis As Integer = 225
Define MoveX As Integer = 3
Define MoveY As Integer = 2
Define Faster As Integer = 3
Define Xstart As Integer = 50
Define Ystart As Integer = 50
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()
While Not IsKeyDown("Escape")
If IsKeyDown("PageUp") Then
MoveX = Faster + MoveX
End If
If IsKeyDown("PageDown") Then
MoveX = MoveX - Faster
End If
Delay(10)
If IsKeyDown("Right") Then
Xaxis = Xaxis + MoveX
End If
If IsKeyDown("Left") Then
Xaxis = Xaxis - MoveX
End If
If IsKeyDown("Up") Then
Yaxis = Yaxis - MoveY
End If
If IsKeyDown("Down") Then
Yaxis = Yaxis + MoveY
End If
myUFO.MoveTo(Xaxis, Yaxis)
End While
myUFO.MoveTo(Xstart, Ystart)
End Method
End Program
```

Keyboard4

```
Program UsingMethods
Define myUFO As Sprite
Define xAxis As Integer = 290
Define Yaxis As Integer = 225
Define MoveX As Integer = 3
Define MoveY As Integer = 2
Define Faster As Integer = 3
Define Xstart As Integer = 50
Define Ystart As Integer = 50
Method Main()
     SetUpTheScreen()
// the player can stop the game by hitting the Esc key
     While Not IsKeyDown("Escape")
// let the player move the UFO with the arrow keys or speed
it up/ slow it down with the Page Up/ Page Down keys
     HandleKeys()
// move the UFO to the next position
     MoveTheUFO()
     End While
// Player hit escape. Move the UFO back to its docking
location
     myUFO.MoveTo(Xstart, Ystart)
End Method
Method SetUpTheScreen()
     myUFO.Load("ufo.gif")
     myUFO.MoveTo(Xaxis, Yaxis)
     myUFO.Show()
End Method
Method HandleKeys()
     If IsKeyDown("PageUp") Then
          MoveX = Faster + MoveX
     End If
     If IsKeyDown("PageDown") Then
          MoveX = MoveX - Faster
     End If
     Delay(10)
     If IsKeyDown("Right") Then
          Xaxis = Xaxis + MoveX
     End If
     If IsKeyDown("Left") Then
          Xaxis = Xaxis - MoveX
     End If
     If IsKeyDown("Up") Then
          Yaxis = Yaxis - MoveY
     End If
     If IsKeyDown("Down") Then
          Yaxis = Yaxis + MoveY
     End If
End Method
     Method MoveTheUFO()
```

```
          myUFO.MoveTo(Xaxis, Yaxis)
          End Method
     End Program
```

If

```
Program IfQuestion
Method Main()

Define Xaxis As Integer = 0
Define Yaxis As Integer = 0
Define MoveX As Integer = 3
Define MoveY As Integer = 2
Define count As Integer

Define RightEdgeofScreen As Integer
RightEdgeofScreen = ScreenWidth() - 65
Define BottomEdgeOfScreen As Integer
BottomEdgeOfScreen = ScreenHeight() - 35

Define myUFO As Sprite
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()

For count = 1 to 1000
Delay (10)
Xaxis = Xaxis + MoveX
Yaxis = Yaxis + MoveY
myUFO.MoveTo(Xaxis, Yaxis)

If Xaxis > RightEdgeofScreen Then
MoveX = MoveX * -1
Else
If Xaxis < 0 Then
MoveX = MoveX * -1
End If
End If

If Yaxis > BottomEdgeofScreen Then
MoveY = MoveY * -1
Else
If Yaxis < 0 Then
MoveY = MoveY * -1
End If
End If

Next
End Method
End Program
```

Exercise 3

```
Define myUFO As Sprite
myUFO.Load("ufo.gif")
myUFO.MoveTo(60,60)
myUFO.Show()
PlaySound(eerie.wav)
Define ufoY As Integer
For ufoY = 551 to 1250
Delay 49
myUFO.MoveTo(60, ufoY)
Next
```

Exercise 5

```
Program IfQuestion
Method Main()

Define Xaxis As Integer = 0
Define Yaxis As Integer = 0
Define MoveX As Integer = 3
Define MoveY As Integer = 2
Define count As Integer

Define RightEdgeofScreen As Integer
RightEdgeofScreen = ScreenWidth()
Define BottomEdgeOfScreen As Integer
BottomEdgeOfScreen = ScreenHeight()

Define myUFO As Sprite
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()

For count = 1 to 1000
Delay (10)
Xaxis = Xaxis + MoveX
Yaxis = Yaxis + MoveY
If Xaxis = 150 Than
Xaxis = Xaxis + 150
End If
myUFO.MoveTo(Xaxis, Yaxis)

If Xaxis > RightEdgeofScreen Then
MoveX = MoveX * -1
Else
If Xaxis < 0 Then
MoveX = MoveX * -1
End If
End If

If Yaxis > BottomEdgeofScreen Then
MoveY = MoveY * -1
Else
If Yaxis < 0 Then
MoveY = MoveY * -1
End If
End If

Next
End Method
End Program
```

Exercise 5extra

```
Program IfQuestion
Method Main()

Define Xaxis As Integer = 0
Define Yaxis As Integer = 0
Define MoveX As Integer = 3
Define MoveY As Integer = 2
Define count As Integer

Define RightEdgeofScreen As Integer
RightEdgeofScreen = ScreenWidth()
Define BottomEdgeOfScreen As Integer
BottomEdgeOfScreen = ScreenHeight()

Define myUFO As Sprite
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()

For count = 1 to 1000
Delay (10)
Xaxis = Xaxis + MoveX
Yaxis = Yaxis + MoveY
If Xaxis = 150 Then
If MoveX > 0 Then
Xaxis = Xaxis + 150
Else
Xaxis = Xaxis - 150
End If
End If
myUFO.MoveTo(Xaxis, Yaxis)

If Xaxis > RightEdgeofScreen Then
MoveX = MoveX * -1
Else
If Xaxis < 0 Then
MoveX = MoveX * -1
End If
End If

If Yaxis > BottomEdgeofScreen Then
MoveY = MoveY * -1
Else
If Yaxis < 0 Then
MoveY = MoveY * -1
End If
End If

Next
End Method
End Program
```

Exercise 6

```
Program Keyboard1
Method Main()
Define myUFO As Sprite
Define xAxis As Integer = 290
Define Yaxis As Integer = 225
myUFO.Load("ufo.gif")
myUFO.MoveTo(Xaxis, Yaxis)
myUFO.Show()
While Not IsKeyDown("Escape")
Delay(10)
If IsKeyDown("s") Then
Xaxis = Xaxis + 3
End If
If IsKeyDown("a") Then
Xaxis = Xaxis - 3
End If
If IsKeyDown("w") Then
Yaxis = Yaxis - 2
End If
If IsKeyDown("z") Then
Yaxis = Yaxis + 2
End If
myUFO.MoveTo(Xaxis, Yaxis)
End While
myUFO.MoveTo(50,50)
End Method
End Program
```

Ufopong

```
Program UFOPong
Define LeftPaddle As Sprite
Define RightPaddle As Sprite
Define myUFO As Sprite
Define moveX As Integer
Define moveY As Integer
Define LeftScoreValue As Integer = 0
Define LeftScoreDisplay As Sprite
Define RightScoreValue As Integer = 0
Define RightScoreDisplay As Sprite
Define EndGame As Integer = 10
Define LeftScoreLocationX As Integer = 200
Define LeftScoreLocationY As Integer = 30
Define RightScoreLocationX As Integer = 396
Define RightScoreLocationY As Integer = 30
Define Xspeed As Integer = 10
Define Yspeed As integer = 10

Method Main()
      Define CurrentTime As Decimal
      Define TimeLastChecked As Decimal
      SetUpGameScreen()
      Alert("Left player use w and z keys, Right player use
up and down arrows", "Let's go")
      StartTheUFO()
      While LeftScoreValue < EndGame And RightScoreValue <
EndGame
            CurrentTime = TickCount()
            If CurrentTime - TimeLastChecked > 25 Then
                  HandleKeys()
                  MoveTheUFO()
                  TimeLastChecked = CurrentTime
            End If
      End While
      ShowTheWinner()
End Method

Method ScorePointLeft()
      myUFO.UnLoad()
      PlaySound("CannonHit.wav")
      LeftScoreValue = LeftScoreValue + 1
      If LeftScoreValue < EndGame Then
            LeftScoreDisplay.Load(LeftScoreValue + ".gif")
            LeftScoreDisplay.MoveTo(LeftScoreLocationX,
LeftScoreLocationY)
            LeftscoreDisplay.Show()
            StartTheUFO()
      End If
End Method

Method ScorePointRight()
      myUFO.UnLoad()
      PlaySound("CannonHit.wav")
      RightScoreValue = RightScoreValue + 1
      If RightScoreValue < EndGame Then
           RightScoreDisplay.Load(RightScoreValue + ".gif")
           RightScoreDisplay.MoveTo(RightScoreLocationX,
RightScoreLocationY)
           RightscoreDisplay.Show()
           StartTheUFO()
      End If
End Method

Method SetUpGameScreen()
      ClearBackground(Black)
      LeftPaddle.Y = 225
      LeftPaddle.Load("leftpaddle.gif")
      LeftPaddle.MoveTo( 30, LeftPaddle.Y)
      LeftPaddle.Show()
      LeftScoreDisplay.Load("0.gif")
      LeftScoreDisplay.MoveTo(LeftScoreLocationX,
LeftScoreLocationY)
      LeftscoreDisplay.Show()

      RightPaddle.Y = 225
      RightPaddle.Load("rightpaddle.gif")
      RightPaddle.MoveTo( 597, RightPaddle.Y)
      RightPaddle.Show()
      RightScoreDisplay.Load("0.gif")
      RightScoreDisplay.MoveTo(RightScoreLocationX,
RightScoreLocationY)
      RightscoreDisplay.Show()
End Method

Method ShowTheWinner()
      ClearSprites()
      PlaySound("tada.wav")
      Alert("Congratulations!", "We have a winner")
End Method

Method HandleKeys()
      If IsKeyDown("Up") And RightPaddle.Y > 1 Then
           RightPaddle.Y = RightPaddle.Y - Yspeed
      End If
      If    IsKeyDown("Down")    And    RightPaddle.Y    <
ScreenHeight() Then
           RightPaddle.Y = RightPaddle.Y + Yspeed
      End If
      RightPaddle.MoveTo(597, RightPaddle.Y)
      If IsKeyDown("w") And LeftPaddle.Y > 1 Then
           LeftPaddle.Y = LeftPaddle.Y - Yspeed
      End If
      If  IsKeyDown("z")  And  LeftPaddle.Y  <  ScreenHeight()
Then
            LeftPaddle.Y = LeftPaddle.Y + Yspeed
      End If
      LeftPaddle.MoveTo(30, LeftPaddle.Y)
End Method

Method MoveTheUFO()
      myUFO.X = myUFO.X + moveX
      myUFO.Y = myUFO.Y + moveY
      myUFO.Show()
      If myUFO.X > (ScreenWidth() + 5) Then
            ScorePointLeft()
            Return
      Else If myUFO.X < 0 Then
            ScorePointRight()
            Return
      Else If myUFO.Y > (ScreenHeight() - 20) Then
            moveY = moveY * -1
            If moveY = 0 Then
                  moveY = -1
            End If
            PlaySound("Bounce.wav")
      Else
            If myUFO.Y < 10 Then
                  moveY = moveY * -1
                  If moveY = 0 Then
                        moveY = -1
                  End If
            myUFO.Y = myUFO.Y + moveY
            PlaySound("Bounce.wav")
            End If
      End If
      If myUFO.Intersects(LeftPaddle) Then
            PlaySound("Bounce.wav")
            moveX = moveX * -1
            moveY = moveY - 1 + Random(0,2)
            myUFO.X = LeftPaddle.X + LeftPaddle.Width + moveX
            myUFO.Y = myUFO.Y + moveY
            myUFO.MoveTo(myUFO.X, myUFO.Y)
      End If
      If myUFO.Intersects(RightPaddle) Then
            PlaySound("Bounce.wav")
            moveX = moveX * -1
            moveY = moveY - 1 + Random(0,2)
            myUFO.X = RightPaddle.X + myUFO.Width + moveX
            myUFO.Y = myUFO.Y + (moveY*2)
            myUFO.MoveTo(myUFO.X, myUFO.Y)
      End If
End Method

Method StartTheUFO()
```

```
     Delay(2000)
     myUFO.Load("SmallUFO.gif")
     myUFO.X = 300
     myUFO.Y = 250
     myUFO.Show()
     moveX = 6
     moveY = Random(1,4)
     If Random(1,2) = 1 Then
          moveX = moveX * -1
     End If
     If Random(1,2) = 1 Then
          moveY = moveY * -1
     End If
End Method

End Program
```