An Effective Approach for Teaching Computer Programming to Freshman Engineering Students

Mohammad H.N. Naraghi and Bahman Litkouhi
Department of Mechanical Engineering

Manhattan College
Riverdale, NY 10471

I. Introduction

Computer programming is an essential and integral part of any engineering program. Engineering students in their junior and senior years face the task of solving problems using numerical approaches. Good programming skills will enable them to tackle those problems easily. Furthermore, a good knowledge of computer programming makes an engineering graduate more attractive to research oriented engineering employers as well as graduate engineering programs. In order to enable students to use their programming skills during the four years of engineering education, the best time for teaching programming is freshman year.

In the past, Fortran was the engineering and scientific programming language. During the1960's, 70's and, to some extend, the1980's, Fortran was the only language with scientific functions. With the emergence of object oriented programming languages (C++, Java and Visual Basic) more attractive alternatives to Fortran became available. All of the new object-oriented programming languages have a comprehensive scientific function library. The question that every engineering program has to answer is "which one of these languages is appropriate for a freshman engineering programming course?" In order to use the object-oriented capabilities of Java and C++ and develop an interesting project, students have to go through a long period of instructions, which often cannot be done in one semester. In fact, we may end up losing students. Retention of engineering freshman is a critical issue in most engineering programs.

Visual Basic (VB) is an attractive alternative to C++ or Java. VB, with its Control Objects, makes computer programming a very interesting subject. In fact, it helps the students like programming. With a few weeks of instruction, students can develop sophisticated programs with a GUI (Graphic User Interface). Additionally, migrating to other languages becomes easy. We, at Manhattan College, teach Visual Basic programming in one of our introductory freshman engineering courses (Computer Programming for Engineers) for eight weeks. The remaining part of this course is devoted to teaching Spreadsheets and Mathcad. One interesting feature of this course that makes it attractive for our freshmen is its semester long group projects. These projects

involve development of computer games, which require some engineering and scientific knowledge. One of these projects is development of a pool game. In this project students have to incorporate the friction and dynamics of bouncing balls when they hit each other or the sides of the a pool table. Additionally, students have to develop programming skill to determine the location of an array of balls (eight balls plus a cue ball) at various time steps and show it in real time (animated mode) on the GUI screen.

II. Description of "Computer Programming for Engineers" Course

The Computer Programming for Engineers introduces freshman students to software and computer skills that can be used in all engineering disciplines. The course is broken into two major modules: computer programming (Visual Basic) and mathematics software (MathCAD). In addition to these two modules, a brief overview of AutoCAD (engineering drawing) and spreadsheets are presented for two weeks at the beginning of the course. Students who are taking this course have already seen an extensive coverage of AutoCAD and spreadsheets in a previous course. After the two weeks overview of AutoCAD and Spreadsheets, the programming module is covered for eight weeks and mathematics module is covered for four weeks. One week is devoted to project presentation, bringing the total coverage of the course to fourteen weeks. The detailed syllabus of the course is provided in its web site: http://www.manhattan.edu/~mnaraghi/engs116/.

The programming module of the course, which is the focus of this paper, covers introductory topics in Visual Basic. These topics are program development cycle, programming tools, the programming environment, VB objects, VB events, numbers, strings, input and output, built-in functions, subprograms, modular design, structured programming, relational and logical operators, IF blocks, select case blocks, use of looping, creating and using arrays, 2-D arrays and VB graphics. An interesting and unique feature of this course is its semester long group project.

At the beginning of the semester students are asked to form their project groups. Students are given the freedom to form groups based on their preference (choosing classmates that they can work with comfortably). By the third week of the course, each group submit a proposal. Each team can select from a list of projects, or in some cases propose their own project. Students taking this course have little or no background in engineering. Therefore, it is impossible to make these projects challenging and interesting hardcore engineering programming projects. The graphic user interface and control objects makes Visual Basic a powerful and easy to use language for developing interesting animated projects or games. We therefore decided to have student groups to develop animated games that have some engineering and scientific implications. Two typical projects developed by students during the last two years are described below:

A. Pool Game Project

In this project students developed a pool game consisting of a white cue ball and eight colored balls. The final graphic user interface of this project is shown in Figure 1. The

user can specify the initial velocity and angle of the cue ball, and then by pressing the "Hit That Ball" button the cue ball will move along the specified direction, hitting other balls. Ball movements are shown in real time on the computer screen. If any of these balls enter one the six holes, ten points are given to the scores and that ball is shown in the box corresponding to the balls that are in the pockets.

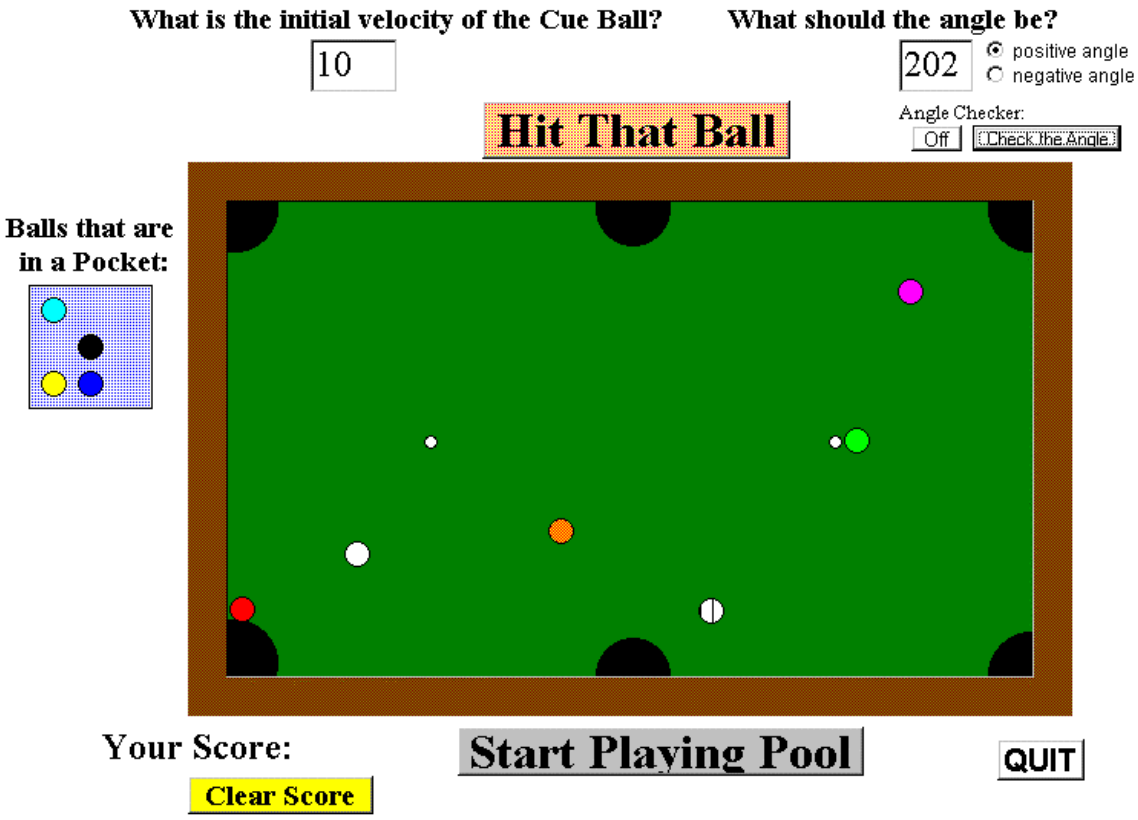

Figure 1: GUI of pool game project

This project involved extensive programming efforts. Each ball is represented by a control array. The motion of each ball is shown in real time using an animation scheme that will be described later. Additionally, it introduced students to the concept of friction, impact of two balls and reflection off from the side-wall of the pool table. The friction factor that results in slowing the ball is taken to be 0.1. Upon impact, the reflection direction follows the impact laws as discussed in the standard dynamics textbook [1]. The pool table is a picture box with a green background. Each ball is a circular shape object of a different color. All nine balls form a control array with a dimension of 9 (representing all nine balls). Six pockets are represented by four quarter circles at four corners and two half circles in the middle of long sides, as depicted in Figure 1.

The time interval for the real time animation is 0.02 seconds and the position of each ball is revised at each time step. Additionally, at each time step, through some IF statements, the program checks whether any of the balls hit the side-walls, other balls, or enter the pockets. If any ball enters a pocket, a positive score of 10 is granted, and if no ball enters

any pocket, 5 points are subtracted from the scores. If the cue ball enters a pocket, the game is lost.

This project involved some engineering concepts, primarily dynamics of motion, impact of objects, and reflection after impact. Additionally, it involved complicated programming. This project was originally developed by a two-member group of students in the spring of 1998. It has been one of inspiring projects for both our freshman as well as the high school students in our outreach programs.

B. Tank Killer Project

This project involves projectile motion of a canon fired from the gun of a tank. Two tanks and three cows are involved (see Figure 2 for GUI of the project). The user specifies the initial velocity of the cannon and its angle, and by pressing the fire button, the cannon fires at the specified direction. The motion of the cannon is shown in real time using an animation scheme. If the cannon hits the other tank, a positive score of ten is given. No score is given if the cannon misses target. If the cannon hits any of three cows, five points are deducted. Each time the game starts, the tanks and cows are positioned randomly. Additionally, the wind direction and magnitude are randomly assigned. This randomness makes the game more challenging.

The scientific and engineering concepts used in this project involve time dependent projectile motion, i.e., *x* and *y* coordinates of an object as functions of time, and initial velocity and angle, given by:

$$x = v\cos(\theta)t \qquad\qquad y = -\frac{g}{2}t^2 + v\sin(\theta)t$$

The programming challenge in this project involves real time animation of the projectile motion of the cannon and uses logical statements (IF statements) to determine the location of cannon (whether it hits the other tank or the cows).

Program listing and details of these two projects and other projects are given in the course web site (http://www.manhattan.edu/~mnaraghi/engs116/engs116.html). All of these projects involve animation. In the following section we will discuss simple approaches for creating real time animation. This topic is missing from standard Visual Basic programming textbooks (such as, [2] and [3]).

III. Animation Using Visual Basic

The speed of plotting a graph or moving an object in Visual Basic is controlled by the speed of the computer that the program is executed on. Many modern computers are so fast that plotting is almost done instantaneously. Therefore, it is impossible for a

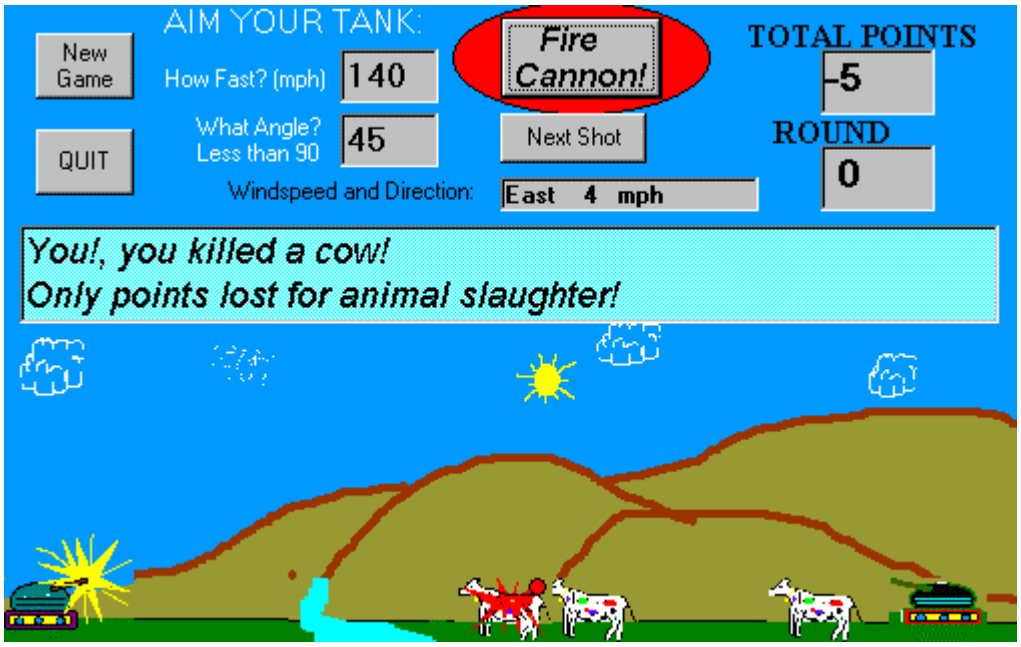

Figure 2: GUI of Tank Killer project

programmer to see the real-time motion of an object. A time delay routine can be used to postpone plotting the new position of the object.

A. Time Delay Routine

Delaying any action (plotting, moving an object, making an object visible or invisible), in Visual Basic, can be achieved by a simple dummy loop. The function of this dummy loop is to kill time until computer time (computational time) becomes the same as real time (or time that an action happens in the real world). The computation time can be evaluated using the Visual Basic Timer. The value returned by the function Timer is the number of seconds from midnight to the time currently stored in the computer's internal clock. The Timer function often is used in comparing computation times of various algorithms. Here, it will be used in a time delay routine. A typical delay routine can be written in the following format:

```
Tstart = Timer
t = 0
Do While t < 20
   t = t + 0.02
Rem Start dummy loop to kill time
 Do While (Timer - Tstart < t)
 Loop
Rem code for real-time animation
     .
     .
     .
     .
Loop
```

Note that `Tstart` in the above routine is set equal to the beginning of the computation (value of `Timer` at the beginning of the computation). Also, t (time in equations) is set to zero. The maximum allowable value of `t` is 20S and it is incremented by 0.02 seconds. Note that most personal computers have a time resolution of 20ms (0.02 s). The loop after time increment statement (`t=t+0.02`) is a dummy loop (does nothing other than kill the time) which keeps looping until the difference between the current computational time and starting time becomes equal or more than the time in the code for real-time animation. The following example illustrates the application of this time routine in Visual Basic programs.

The application of the time delay routine is demonstrated via a Visual Basic program that shows real-time locations of a circular object being thrown from the center of the coordinates at an initial velocity and angle. Initial velocity and angles are input via text boxes. The axes of coordinates, as well as, the initial location of the object are shown by clicking on the start button. The object moves by clicking on the "throw the object" button.

The coordinates of an object in a Cartesian coordinate system x and y are given by equations of projectile motion. The properties of objects involved in this example are given in the following table and layouts of objects are shown in Figure 3.

Object	**Property**	**Setting**
Form	Name	frmTraj
	Caption	Trajectory of a circular object
Label	Name	lblVelocity
	Caption	Initial Velocity
Label	Name	lblAngle
	Caption	Initial Angle
TextBox	Name	txtVelocity
	Caption	blank
TextBox	Name	txtVelocity
	Caption	blank
TextBox	Name	txtVelocity
	Caption	blank
CommandButton	Name	cmdStart
	Caption	Start
CommandButton	Name	cmdThrow
	Caption	Throw the object
CommandButton	Name	cmdQuit
	Caption	Quit

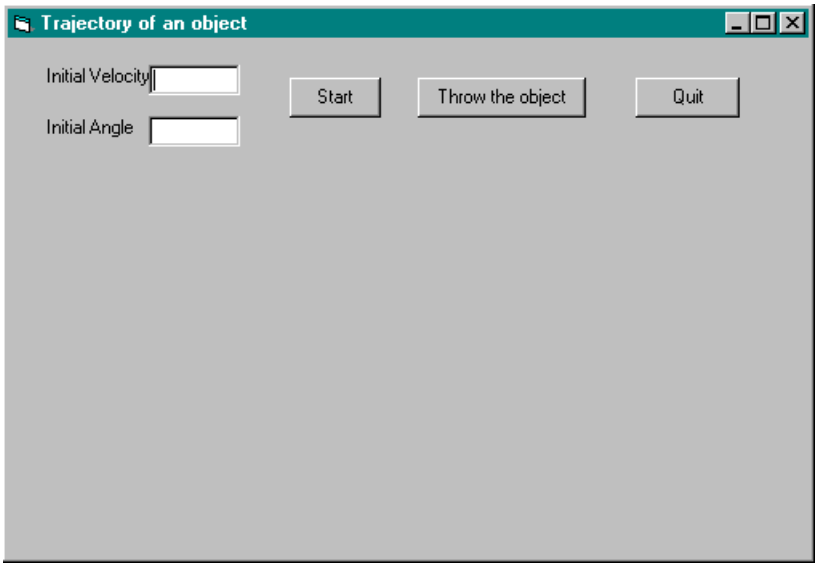

Figure 3: Layout of Objects for Trajectory of an object example

The Visual Basic code for this problem is given below:

```
Private Sub cmdQuit_Click()
        End
End Sub

Private Sub cmdStart_Click()
        frmTraj.Cls
        frmTraj.Scale (-2, 15)-(15, -2)
        frmTraj.Line (-2, 0)-(15, 0)
        frmTraj.Line (0, -2)-(0, 15)
        frmTraj.Circle (0, 0), 0.25
End Sub

Private Sub cmdThrow_Click()
        Dim x, y, pi, Tstart, t, v0, theta As Single
        x = 0
        y = 0
        pi = 4 * Atn(1)
        Tstart = Timer
        t = 0
        v0 = Val(txtVelocity)
        theta = Val(txtAngle) * pi / 180
        Do While (y >= 0)
           Let t = t + 0.02
           Do While (Timer - Tstart < t)
           Loop
           x = v0 * Cos(theta) * t
           y = -4.905 * t ^ 2 + v0 * Sin(theta) * t
           frmTraj.Cls
         frmTraj.Line (-2, 0)-(15, 0)
```

```
         frmTraj.Line (0, -2)-(0, 15)
         frmTraj.Circle (x, y), 0.25
      Loop
End Sub
```

Note that in the above program the time increment is set to 0.02 s since the timer resolution in most is 0.02 s. In this program, first, the initial velocity and angle with respect to the x-axis must be specified. Then, by clicking on the start button, axes of coordinates, along with a circle at the origin appear. Finally, by clicking on the throw the object button, the real-time trajectory of the circle will be shown while, it stays above the x-axis. By trying various angles, one notice that the maximum horizontal distance that the circle travels can be achieved when the angle is 45^o (known from physics and dynamics).

An alternative way is to introduce the circle as an object. This can be done by adding a shape to the list of objects with the following properties:

Object	**Property**	**Setting**
Shape	Name	shpCircle
	Shape	Circle
	Visible	False

Then statement `frmTraj.Circle (0, 0), 0.25` in subroutine `cmdStart_Click` is replaced by:

```
shpCircle.Visible=True
shpCircle.Move 0,0
```

Also, in `cmdThrow_Click()` subprogram, statement `frmTraj.Cls` is removed and `frmTraj.Circle (x, y), 0.25` is replaced by `shpCircle.Move x,y`. Since `frmTraj.Cls` is removed, no blinking will occur in this procedure. Note that with this procedure the position of the circle, at the beginning and end of the throwing process, slightly below the x-axis. This is due to the fact that coordinates x and y in Shapes are coordinates of the upper-left-hand-side of the object.

One can make this object throwing program more sophisticated by introducing real images of, for example, basketball and basketball players. This can be accomplished by using Image from the object toolbar. The image of a ball can be downloaded from the Web (from basketball game pictures) and the ball image can be isolated using a graphics program. A good graphics program for this purpose is Paint Shop Pro (can be downloaded from www.shareware.com). In the next section the use of Timer Control (Object) in animation will be demonstrated.

B. Timer Control

The Timer control, which appears as an alarm clock in the toolbox, can be use in animation and simulation. The timer control appears as a small alarm clock at design time but is invisible during run time, so it does not matter where it is located. An important

property of the timer control is the interval, which is measured in milliseconds. In order to begin a timer event, it must first be turned on by setting its Enabled property to True. It can be turned off by setting Enabled Property to False or Interval to 0. A digital clock is a very simple but very useful application involving a timer control. Once the application of the timer control as a digital clock is understood, a more complicated application involving animation will be discussed. The Digital Clock application includes a timer and a label with a border.

Example, Digital Stopwatch: This program creates a digital stopwatch that shows the elapse time in seconds.

Object	**Property**	**Setting**
frmDSW	Name	Digital Stopwatch
cmdStart	Caption	Start Watch
cmdStop	Caption	Stop Watch
tmrWatch	Interval	100
	Enabled	False
LblSeconds	Caption	Seconds
LblTime	Caption	(blank)

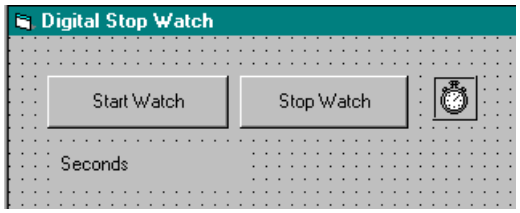

Figure 4: Layout of objects for digital stop watch example

```
Private Sub cmdStart_Click()
     lblTime.Caption = 0   'Reset the Watch
     tmrWatch.Enabled = True
End Sub

Private Sub cmdStop_Click()
     tmrWatch.Enabled = False
End Sub

Private Sub tmrWatch_Timer()
     lblTime.Caption = Val(lblTime.Caption) + 1
End Sub
```

Now by running this program the elapsed time can be measured. An interesting application of the timer control is animation. The following example demonstrates the application of Timer control in animation.

Example, Runner: In this example the timer control is used to make various frames of a runner visible and invisible, and by doing so, show a running animation. Twelve frames of a runner are available (these frames are taken from a sample case in Paint Shop Pro, Version 5). These pictures (frames) are named according to the following figure:

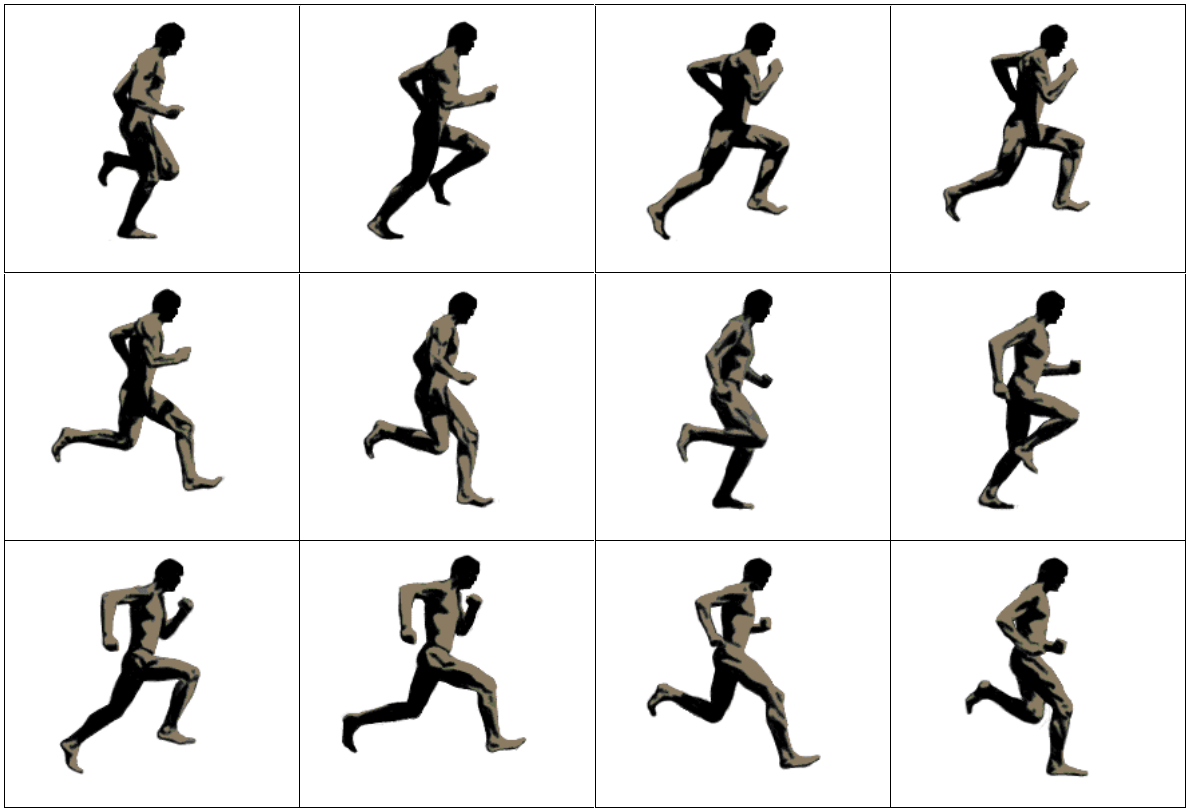

Figure 5: Images of a runner at different time frames (ImgRunner1.gif through ImgRunner12.gif, from the upper-left-hand-side to the lower-right-hand-side).

The object properties for this example are as follows:

Object	**Property**	**Setting**
frmRunner	Caption	Runner
cmdRun	Caption	Run
cmdStop	Caption	Stop
tmrTimer	Enable	False
	Interval	100
LblTimer	Caption	(blank)
imgRunner	Picture	Runner1.gif
	Index	0
	Visible	False
imgRunner	Picture	Runner2.gif
	Index	1
	Visible	False

.	.	.
.	.	.
.	.	.
.	.	.
imgRunner	Picture	Runner12.gif
	Index	11
	Visible	False

Note that the location of images (imgRunner1 through imgRunner12) is not important. They are initially invisible. The object layout can be similar to that shown in Figure 5.

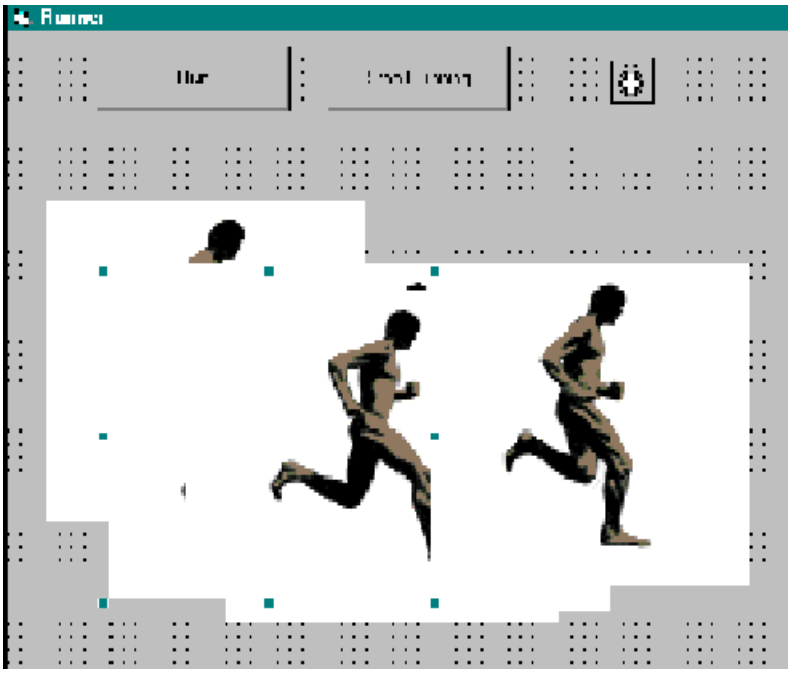

Figure 6: Layout of objects for the runner problem

The code listing for this problem is as follows:

```
Private Sub cmdRun_Click()
   For i = 0 To 11
      imgRunner(i).Top = 800
      imgRunner(i).Left = 800
   Next i

   lblTime.Caption = 0
   tmrTimer.Enabled = True
End Sub

Private Sub cmdStop_Click()
   tmrTimer.Enabled = False
 End
```

```
End Sub

Private Sub tmrTimer_Timer()
   lblTime.Caption = lblTime.Caption + 0.1
   For t = 0 To 12 Step 1.2
      For i = 0 To 11
   If (lblTime.Caption = t + (i + 1) * 0.1) Then
      imgRunner(i).Visible = False
      If i = 11 Then
         imgRunner(0).Visible = True
      Else
         imgRunner(i + 1).Visible = True
      End If
   End If
      Next i
   Next t
End Sub
```

Note that in this code, initially after the run button is clicked the same coordinates (locations) are assigned to all twelve images. In other words, they are put on top of each other. Then when the Timer is Enabled, images are made visible in there order with a time delay of 0.1 s. The whole sequence of twelve images is put in a loop to make the animation process continuous. The user can click on the stop button to terminate the program. Figure 7 shows the final results during a run.

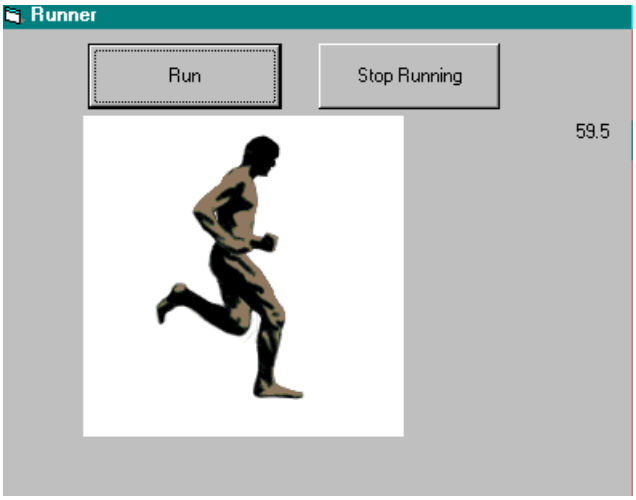

Figure 7: A typical run for runner example problem

IV. Conclusions

Since we started this practice four years ago, our students have developed a number of interesting projects such as, basketball, war game, etc.. Additionally, a number of our students who decided to learn other languages, such as Fortran and C++, were able to learn them quickly and easily, mostly via self-study. The programming skills that some

of these students gained from this course resulted in getting summer internships at prestigious laboratories, such as, NASA.

Bibliography

1. Beer, F.P. & Johnston, E.R, Vector Mechanics for Engineers, Statics and Dynamics, 6th Ed., McGraw-Hill, 2000.
2. Schneider, D.I., An Introduction to Programming Using Visual Basic 6.0, 4th Ed., Prentice Hall, 1999.
3. Coburn, E.J., Visual Basic 5 Made Easy, 2nd Ed., PWS, 1999.

MOHAMMAD H.N. NARAGHI

Dr. Mohammad H.N. Naraghi is a Professor of Mechanical Engineering at Manhattan College. He received his M.S. and Ph.D. in Mechanical Engineering from University of Akron. Dr. Naraghi worked closely with NASA Lewis Research Center, through research grants and a number of fellowships, to develop a comprehensive Rocket Thermal Evaluation code (RTE). He received a certificate of recognition from NASA for the creative development of technically significant software. Dr. Naraghi worked closely with NASA as well as a number of Aerospace companies to expand capabilities of the code. Dr. Naraghi's research is in Thermal/Fluids area and he has published more than fifty articles in ASME, AIAA and international journals and conferences. He is recipient of a number of research grants from NASA and Air Force.

BAHMAN LITKOUHI

Dr. Bahman Litkouhi is a Professor and the Chair of Mechanical Engineering Department at Manhattan College. He also served as the Director of Freshman Engineering Program in the School of Engineering for four years (1996-2000). Dr. Litkouhi received his M.S. and Ph.D. degrees in Mechanical Engineering from Michigan State University. His interests include analysis and design of thermal/fluid systems, bioengineering modeling, and computational heat transfer. Dr. Litkouhi is a registered Professional Engineer in New York State and is actively involved in joint research with industry.