JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2005**

Vol. 4, No. 9, November-December 2005

Game programming – The "Why", "What" and "How" with Graphics Objects

Lakshmi Prayaga, Lecturer, Computer Science Department, University of West Florida, Pensacola

Abstract

Game Programming is the latest buzz word in the computer science educational curriculum. Many universities across the world are offering complete CS degree programs in game programming while others such as the Art institute of Portland have summer camps for teens in game programming (Ai of Portland, 2005). This brings us to three interesting questions, The " Why", "What" and "How" of game programming. This paper serves the purpose of analyzing these questions within the context of a Visual basic.Net application. A discussion on the choice of Visual Basic .Net as the environment to create and manipulate the objects in the game is also presented. The code used is also included in the paper.

1 WHY GAME PROGRAMMING?

The idea of using games for educational purposes in K-12 and also for higher education is well documented (Dempsey etal, 1997, cited by Seay and Scott). However, educators are split on the idea of using games in a class room. Some feel it is very effective to teach a concept while others feel it takes up valuable class time. But research (J.C.Herz , in JoyStic Nation, p 1-2 cited by Seay and Scott, Affisco, 1994, p.171 cited by Feldgen and Clua) shows that students these days have a totally different way of learning. They react more to interactive learning. In fact Affisco coins the word 'Edutainment' a combination of entertainment and education and points out that if students are not entertained while they learn, the instructor has lost them. Herz argues that, "If your memories have pop soundtracks or big-screen kisses, if you've ever told an anecdote with instant replays or a coda of stadium applause, it's because you've been brought up with media that furnish those conventions. Video games provide a new set of conventions, which are being rapidly assimilated, as you read this, by a legion of six-year-olds. Their mental grammar is going to reflect that, just as the baby boomers' worldview echoes the impact of television. But whereas TV turned kids of the fifties and sixties into a nation of screen watchers, videogames have created a cadre of screen manipulators (p.1-2)." Continuing on this thought, Lave(1991) and Mcllelan(1995)argue learning happens

within a context and culture of a society, i.e. it is situated. The typical student today(2000 – present) is very well versed in playing games on the computer. They acquire high levels of skills in playing these games and in trying to beat the computer. In addition, many argue that (Coleman et al. 2005) “videogames are the initial draw to technology for a number of students in the first place”, (Pleva 2004) they provide a “stepping stone for kids into the world of technology. Students who play computer games tend to be more comfortable with the technology and more adept at using it.”.

Ideas such as these expressed by educationalists show that games are here to stay and are being used as educational tools, and are in many cases successful at delivering educational content in a way that students internalize those concepts being presented. Seay and Scott cite a reference to an article in the American Scientist which shows that the IQ of present day students is increasing and one of the theories for the increase in IQ “is that our increased exposure to visually rich mass media---television, computers, video games -- has enabled us to perform better on spatial-visual tests. The theory states that the ability to interpret different forms of information from different media and make some kind of sense of it is what these IQ tests actually test for.”

The game environment is almost like a comfort zone for the student by which he is not intimidated. It is in fact his preferred choice of context to work in. It is therefore an area that can be used advantageously by educators to motivate students to be focused in and complete their assignments. In fact Feldgen and Clúa show that most students in the year 2003, chose the context of the game environment for their projects compared to the context of engineering, calculus or business and were also successful in completing their projects. This shows that the gaming environment has a heavy influence on the students mind and therefore also acts as a good motivator. Not surprisingly, many universities (GameDev.Net05) such as University of Pennsylvania, University of Denver, Middlesex University UK, University of North Texas have now started to think seriously about including game technology into the CS curriculum.

2 WHAT IS GAME PROGRAMMING?

“Game programming is a subset of game development, is the programming of computer, console or arcade games. Though often engaged in by professional game programmers, many novices may program games as a hobby. Most software engineering students program games as exercises for learning a programming language or operating system. In fact, the Unix operating system and the C programming language exists because the original programmers wanted to play games” (Wikepedia 2005).

Game programming is both an interdisciplinary and an intra-disciplinary field. It is an intra-disciplinary field of computer science because it includes (Masuch, Freudenberg 2002). “Computer graphics, AI, simulation, user interface design are core components of every game. And more: network techniques, multimedia, databases and many other disciplines take part in game development”. It is an interdisciplinary fields due to its

heavy dependence on photography, physics, imagery, audio and video technology, in short all elements of multi media technology.

The above topics provide the necessary background knowledge to deal with the complex relationships among many abstract and concrete components of game programming. Some of these components are the story line, picturization, characters, shapes, sound, interactivity and other multimedia effects. These components can broadly be classified into two categories, visual and non-visual. This paper focuses on creating and using graphics objects as the primary source of visual representation of the game being planned. Graphics objects form the core of game programming because images used in any game have properties which are similar to the properties of a graphics object. An example is that of an image bound within a rectangular or circular graphics object. We show how geometric shapes can be made into objects that have properties, events, and methods suitable for game programming.

In addition to introducing students to the idea of "objects", game programming can be used to introduce introductory data structures such as arrays and records. Together, these topics form the core of an introductory object oriented programming (OOP) course with gaming as the context. As discussed earlier, students can relate to this context, and hence there is a better chance of meeting and improving student learning outcomes in an introductory OOP course. In recent years game programming is being introduced into the CS curriculum in many different ways such as (deLaet et al. 2005)"a games course for women, a software development course that uses games as projects, an introductory games programming course in Java, and an advanced graphics course that focuses on games".

Regardless of which way a game programming course is introduced, there are some essential elements of the subject which must be taught. These include graphics objects, animation, physics of collisions, collision detection, collision response and multimedia effects. The elements of game programming presented in this paper implement the recommendations of Cunningham et al. (2004) as shown in Fig1.

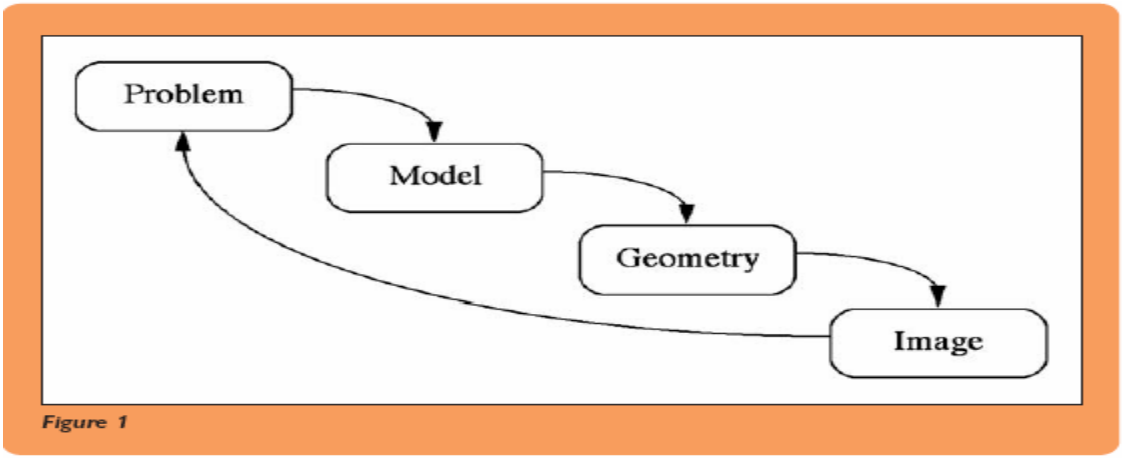


Figure1

Figure one provides a framework that explains the design process and is discussed below:

- Problem statement: In order to create a game, it is important to formulate the story line, followed by identification of the individual components participating in the game and their relationships. An example is to visualize a scenario when in which a group of objects appear on the screen and start moving randomly.
- Model: Once the statement of the problem is verbalized, the next step is to conceptualize or abstractly visualize a model that represents this problem.
- Geometry: This conceptualization must now be translated into figures, outlines or geometric shapes. For example the objects can be thought of as spherical, rectangular, triangular or polygonal objects which have geometric properties such as points, angles, vertices and lines. Objects may also have irregular shapes, with pictures of trees, birds, animals etc.
- Image: The last phase is to visually depict this model on the screen of a computer by using an image that corresponds to this model.

These four phases constitute the building blocks of game programming. As shown in fig.1, looping back to the initial phase (or to any phase from any phase, similar to the waterfall life cycle model in software engineering - Schach02) may occur many times during the design of the program. Only one loop back is shown in the figure to avoid clutter. The next section is a discussion on the implementation of these core concepts.

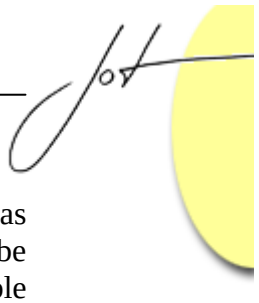
3 THE HOW OF GAME PROGRAMMING:

We discuss the implementation of the principles of game programming based on fig1 with reference to a specific game environment being designed. Graphical reusable/replaceable objects are used to illustrate the use of the four elements of game programming discussed above.

The problem statement/ story line of the game: The game being designed is a combat operation between different geometric shapes. The user/player controls one of the objects and the rest of the objects try to hit the user/player object and destroy it. The elements required at this stage include creation of objects for the game, the movement of these objects (animation) on the game board, the collision of objects, the result of this collision and audio clips to add sound effects to the environment.

The model: A model is an abstract visualization of the problem being discussed. The model for this game is the interaction between the two sets of objects, system/game controlled objects and the user controlled object. These shapes do not have any social skills, so when they interact with the user object the user object is either destroyed completely or shattered into smaller pieces. So, the outcome of the combat is collision, destruction or the survival of some objects

Geometry: The geometry is the crucial piece of game programming. It is at this stage that decisions regarding the size of the game board, the types of objects, the shape of the objects and the location of the objects on the board, have to be made. During the



game, various geometrical transformations of the objects have to be performed, such as rotations, translations, scaling, etc. Students designing the game must therefore be particularly familiar with coordinate geometry, linear algebra and trigonometry to be able to make these decisions. The importance of geometry in game programing is also evident as it is included as one of the major courses students wishing to major in game programming must take. (Coleman et al. 2005). We assume that students entering to take a course on game programming have this pre-requisite knowledge.

Image: The images on the screen follow the decisions made in the geometry phase. Images translate the model into a visualization. Depending on the expertise of the programmer, imagery can become a very complex element of game programming.

Why Visual basic.Net: The choice of Visual basic.Net as the programming language to introduce students to game programming was the result of an experiment conducted on a batch of at risk high school students. These students were ninth graders who were participants of the "reach for tomorrow" program which is designed to motivate at risk students to acquire military scholarships and pursue higher education. (Reachfortomorrow.org) The students had no prior computer programming experience and were also not very comfortable using the computer. The limitations under which the experiment was conducted were:

- Limited amount of time (two and a half hours)
- Students having limited computer experience
- Zero programming experience

Given these limitations, I needed to choose a language that was user friendly and did not have a high learning curve. Visual basic.net seemed to fit this bill. I used VB.net to get the students develop a simple game that included the basics of game programming by

- using picture boxes to hold the objects used in the game
- simulating animations and movements with timer controls
- simulating ideas of velocity, speed, direction
- including the idea of collision detection and response

The result of this experiment was that the students enjoyed the experience. In fact students at this stage were motivated enough to experiment with changing the speed, direction, size and other aspects of the objects used in the program. This shows that choosing the proper context (games) and providing the proper and easy to use programming environment (user friendly VB.Net tool box) generates the required motivation for students to pursue their goals. Visual basic.Net is a very user friendly language because of its rich GUI environment. Presented below is the design of an application that includes the basic elements of a game program discussed so far.

4 DESIGN

The table below shows the design of the game in terms of the elements used in the game and their corresponding VB.Net elements.

Game Element	VB.Net Element
An area to play the game	PictureBox
The Players	Graphics Objects
Movement of players	Controlled by timer Controls
Collision - detection	Intersectswith property
Collision – Response	Simulation of Explosion and collision

The graphics objects used in this program are created by instantiating the shape object named ObjectShape. Since the objectShape is an object, we can reuse it and instantiate it as many times as necessary. The ObjectShape object has properties and events as follows:

- Properties:
  Color, shape, size
- Events:
  Click

Using this method we set up six small ellipses which are the opponents in the game and one slightly larger ellipse which is controlled by the user. The user controls his piece with the mouse, so he can move around the play ground.

5 THE WORKING OF THE GAME

To maintain clarity and for demonstration purposes the elements of game programming incorporated in this example are kept simple. These include animation/movement of the objects, velocity, collision, graphics objects and audio rendition.

Animation

Animation is provided by using a timer control. The timer control’s tick event is used to move the graphic object on the screen, horizontally, vertically or diagonally. The timer control’s tick event also checks for collision occurrence and if collision occurs, the function collide is called to swap the velocities of the objects. The code for moving the objects is included in the moveAround sub procedure.

Collision

Collision detection and response are other key elements of game programming. There are several kinds of collision responses that occur when objects collide. We discuss three types of collision responses - one in which an object bounces off a wall, one in which two objects have a head on collision and bounce off each other and a third in which an explosion or shattering occurs on collision

- A ball hits a wall and bounces off. (code in the sub procedure moveAround)
- A ball collides with another ball and the two balls bounce off of each other. (code in the sub procedure collide) In the sub procedure collide the velocities of the two objects are interchanged.
- The collision response resulting in an explosion is achieved by disposing off the object (user controlled object) and creating six smaller objects on the screen. This creates the effect of an object being shattered and explode into small pieces.

Sound:

The next element used in the game to provide sound effects is audio. SoundWav.vb is a simple class file used in the application to provide the functionality of playing a wave sound file when necessary. This is a modified version of a class file available on the internet (soundFile) We have used two different sound files for collision between the game objects and collision between the game object and the user object. When collision occurs between the game object and the user object we have a “crashing” or “exploding” sound, and when collision occurs between game objects we have a ‘ding’ sound. This completes the design of the game elements.

User Interface:

The program has two buttons, Wall Collision/Dispersion and Ball Collision (Figure2). The Wall Collision/Dispersion checks for

- Collision of the small balls with the wall and if collision occurs the balls bounce off the wall.
- It also checks for collision between the small ball and the large ball and if collision occurs the large ball shatters into small pieces. (Figure3)

The Ball Collision button checks for

- Collision between two small balls on the screen and if collision occurs, the swapping of velocities takes place. (Figure4)
- It also checks for:
- collision between the small ball and the larger ball which is the user controlled piece and if collision occurs the user control is disposed.
- Wall collision is also included in this button. See figures 1, 2 and 3 for the game in action

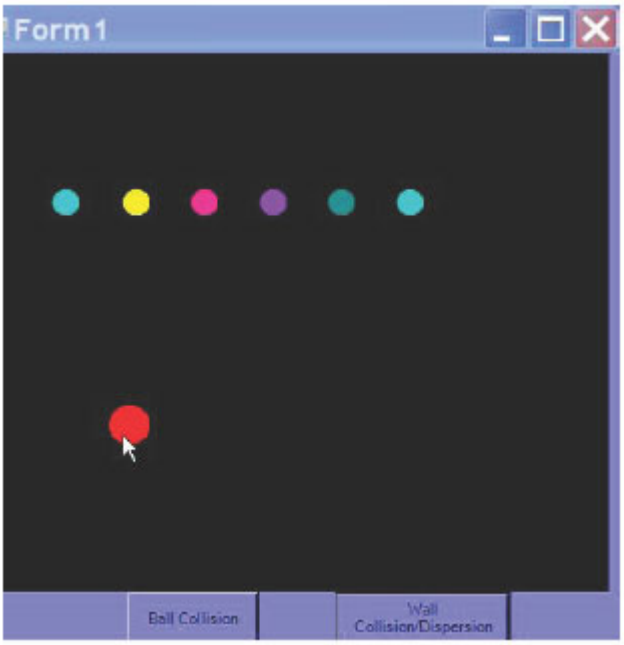


Figure 1

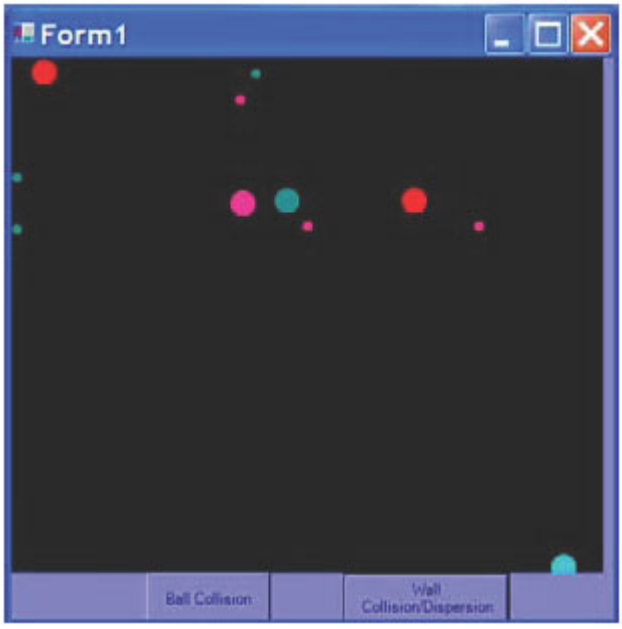


Figure 2

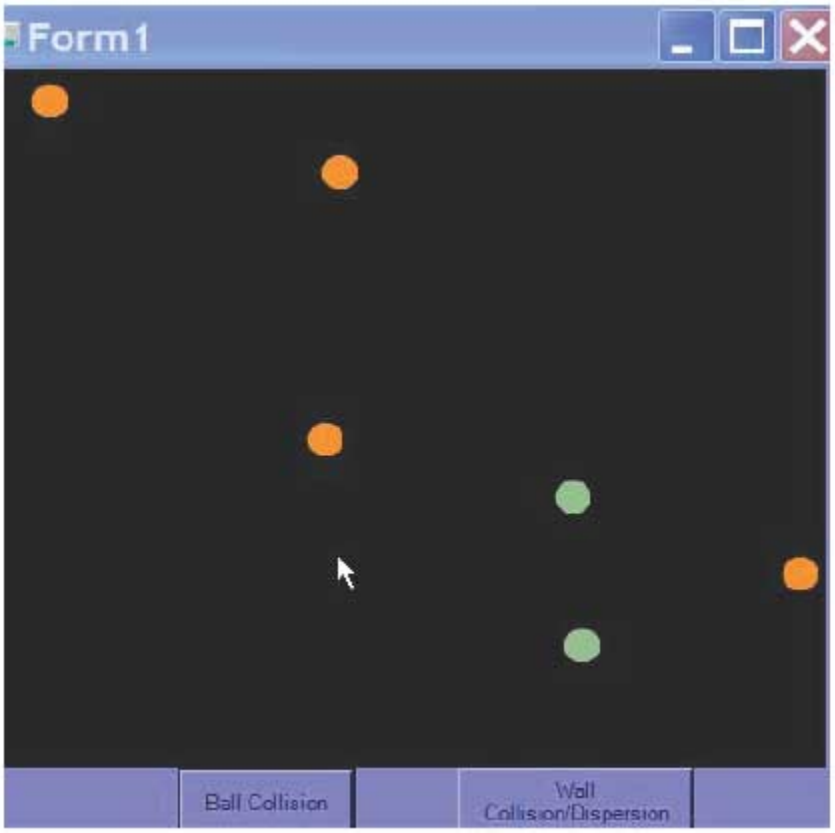


Figure 3

6 CODE FOR THE PROGRAM

Part A – The Class file - Object Shape:

```
#Region "Import"
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Drawing.Color
Imports System.Drawing.Graphics
#End Region

#Region "Class ObjectShape"
Public Class ObjectShape
    Inherits System.Windows.Forms.UserControl

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()
        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
    End Sub
```

```
    'UserControl1 overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As
Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
       InitializeComponent()
        '
        'UserControl1
        '
        Me.Name = "ObjectShape"
    End Sub
#End Region

    Private Sub UserControl1_Load(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles MyBase.Load

    End Sub

#Region "Variables"
    Public p1 As New System.Drawing.Pen(System.Drawing.Color.Blue)
    Public p2 As New System.Drawing.Pen(System.Drawing.Color.Orange)
    Public p3 As New
     System.Drawing.Pen(System.Drawing.Color.PowderBlue)
    Public br1 As SolidBrush
    Public gradBrush As System.Drawing.Drawing2D.LinearGradientBrush
    Public frm1 As System.Windows.Forms.Form
    Public g As Graphics
    Public gp As GraphicsPath
    Public reg As Region
    Public rect As Rectangle
    Public localmpos As Point
    Public chk As Boolean
    Private _fillColor As Color = System.Drawing.Color.Red
    Public _gradColor As Boolean
    Private _shape As Shapes
    Public Tri_Points(2) As Point
    Private velx, vely As Integer

#End Region
```

```
#Region "Enums"
    Public Enum Shapes
        Rectangle
        Ellipse
        Triangle
        Arc
        Line
        Text
        MixShapes
    End Enum
#End Region

#Region "Events"

    Protected Overrides Sub OnPaint(ByVal e As
     System.Windows.Forms.PaintEventArgs)
        Dim point(2) As Point
        Dim g As Graphics
        e.Graphics.SmoothingMode = e.Graphics.SmoothingMode.AntiAlias
        p1.Width = 2.0
        br1 = New SolidBrush(_fillColor)
         If _gradColor = True Then
            point(0) = New Point(Me.ClientRectangle.Left,
            Me.ClientRectangle.Top)
            point(1) = New Point(Me.ClientRectangle.Left +
            Me.ClientRectangle.Width, Me.ClientRectangle.Top +
            Me.ClientRectangle.Height)
            gradBrush = New
            System.Drawing.Drawing2D.LinearGradientBrush(point(0),
            point(1), Color.Red, Color.Yellow)
            e.Graphics.FillRectangle(gradBrush, Me.ClientRectangle)
            e.Graphics.DrawRectangle(p1, Me.ClientRectangle)
        ElseIf Me._shape = Shapes.Rectangle And _gradColor = False Then
            e.Graphics.FillRectangle(br1, Me.ClientRectangle)
            e.Graphics.DrawRectangle(p1, Me.ClientRectangle)
        ElseIf Me._shape = Shapes.Ellipse Then
            e.Graphics.FillEllipse(br1, Me.ClientRectangle)
        ElseIf Me._shape = Shapes.Triangle Then
            Tri_Points = GenerateTrianglePoints(New Rectangle(0, 0,
       Me.Width,   Me.Height))
            e.Graphics.FillPolygon(br1, Tri_Points)
            e.Graphics.DrawPolygon(p1, Tri_Points)
        ElseIf Me._shape = Shapes.Arc Then
            e.Graphics.DrawArc(p1, Me.ClientRectangle, 12, 84)
        ElseIf Me._shape = Shapes.Line Then
            Dim pt1 As Point = New Point(30, 30)
            Dim pt2 As Point = New Point(110, 100)
            e.Graphics.DrawLine(p1, pt1, pt2)
        ElseIf Me._shape = Shapes.Text Then
            e.Graphics.DrawString("Welcome to the Graphics World",
            Me.Font, New SolidBrush(Color.Red), 20, 20)
        ElseIf Me._shape = Shapes.MixShapes Then
            e.Graphics.FillEllipse(br1, Me.ClientRectangle)
            e.Graphics.DrawArc(p2, 100, 100, 100, 100, 12, 84)
```

```
            e.Graphics.DrawPolygon(p3, Tri_Points)
        End If
        MyBase.OnPaint(e)
    End Sub

    Protected Overrides Sub OnMouseDown(ByVal e As
    System.Windows.Forms.MouseEventArgs)

    End Sub

#End Region

#Region "Properties"
    Public Property chgColor() As Color
        Get
            Return _fillColor
        End Get
        Set(ByVal Value As Color)
            _fillColor = Value
            Invalidate()        'Forces a paint event
        End Set
    End Property


    Public Property chgShape() As Shapes
        Get
            Return _shape
        End Get
        Set(ByVal Value As Shapes)
            _shape = Value
            Invalidate()
        End Set
    End Property
    Public Property vx() As Integer
        Get
            vx = velx
        End Get
        Set(ByVal Value As Integer)
            velx = Value
        End Set
    End Property
    Public Property vy() As Integer
        Get
            vy = vely
        End Get
        Set(ByVal Value As Integer)
            vely = Value
        End Set
    End Property
    Private Function GenerateTrianglePoints(ByVal Rect As Rectangle) As
    Point()
        'Generate Triangle

        Dim _Width As Integer = Rect.Width
```

```
        Dim _Height As Integer = Rect.Height
        Dim CenterPoint As Integer
        Dim LeftPoint As Integer
        Dim RightPoint As Integer
        Try
            CenterPoint = _Width / 2 + Rect.X
            LeftPoint = _Height + Rect.Y
            RightPoint = _Width + Rect.X
            Tri_Points(0) = New Point(CenterPoint, Rect.Y)
            Tri_Points(1) = New Point(Rect.X, LeftPoint)
            Tri_Points(2) = New Point(_Width + Rect.X, _Height + Rect.Y)
            Return Tri_Points
        Catch ex As Exception
            Throw ex
            Return Nothing
        Finally
        End Try
    End Function
#End Region
End Class
#End Region
```

Part B Class for Sound File –

```
Imports System.Resources
Imports System.IO

Public Class WinWavSound
    'Declare constants
    Public Const SND_ASYNC As Int32 = 1
    Public Const SND_MEMORY As Int32 = 4

    'this is the API we want to play embedded Wav resource...
       Public Declare Function PlayWavSound Lib "winmm.dll" Alias
      "PlaySound" (ByVal data() As Byte, ByVal hMod As IntPtr, ByVal
      dwFlags As Int32) As   Boolean

    Public Shared Function PlayWavResource(ByVal EmbeddedWav As String)
            'BE SURE (embedded).wav HAS BUILD ACTION IN PROPERTIES SET
            TO EMBEDDED RESOURCE
                  'get the namespace
            Dim strNameSpace As String =
                  System.Reflection.Assembly.GetExecutingAssembly().GetN
                  ame().Name.ToString()
                  ' get the resource into a stream
            Dim Str As Stream =
                  System.Reflection.Assembly.GetExecutingAssembly().GetM
                  anifestResourceStream(strNameSpace + "." +
                  EmbeddedWav)
                  'bring stream into a byte array
            Dim byteStr(Str.Length) As Byte
                  Str.Read(byteStr, 0, Int(Str.Length))
```

```
                'play the resource
                PlayWavSound(byteStr, IntPtr.Zero, SND_ASYNC Or
SND_MEMORY)
    End Function
End Class
```

Part C – Main program:

```
Imports System.Drawing
Imports System.Drawing.Drawing2D

Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim Ball() As ObjectShape.ObjectShape
    Dim smallBall() As ObjectShape.ObjectShape
    Dim navBall As ObjectShape.ObjectShape
    Dim g As Graphics
    Dim gp1, gp2 As GraphicsPath
    Dim reg, reg1, reg2 As Region
    Dim rect, rect1, rect2 As Rectangle
    Dim rectA() As Rectangle
    Dim pen1 As New Pen(linGrBrush)
    Dim colorChoice(4) As System.Drawing.Color
    Dim colorChoiceBig(5) As System.Drawing.Color

    Private Sub BtnWall_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
        Timer1.Interval = 10
        Timer1.Start()

    End Sub

    Private Sub moveAround(ByVal Ball As ObjectShape.ObjectShape)
        'Moves the ball around window, bouncing off walls
        With Ball
            If .Left <= 0 Or .Left >= PictureBox1.Width - .Width Then
                .vx = -.vx
            ElseIf .Top <= 0 Or .Top >= PictureBox1.Height - .Height
Then
                .vy = -.vy
            End If
            .Left += .vx
            .Top += .vy
        End With
    End Sub

Private Sub collide(ByVal Ball1 As ObjectShape.ObjectShape, ByVal Ball2
As ObjectShape.ObjectShape)

        Dim temp As Integer
        temp = Ball1.vx
        Ball1.vx = Ball2.vx
        Ball2.vx = temp
```

```
        Ball2.chgColor = Color.DarkSeaGreen
        temp = Ball1.vy
        Ball1.vy = Ball2.vy
        Ball2.vy = temp
        Ball1.chgColor = Color.DarkOrange
 End Sub

Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim myColorBig As Integer
        g = Me.CreateGraphics
        gp1 = New GraphicsPath
        gp2 = New GraphicsPath
        pen1.Width = 4
        'set color for array elements for the six balls created
        colorChoiceBig(0) = System.Drawing.Color.Yellow
        colorChoiceBig(1) = System.Drawing.Color.Red
        colorChoiceBig(2) = System.Drawing.Color.DarkCyan
        colorChoiceBig(3) = System.Drawing.Color.DeepPink
        colorChoiceBig(4) = System.Drawing.Color.DarkTurquoise
        colorChoiceBig(5) = System.Drawing.Color.DarkOrchid

        Dim j As Integer
        ReDim Ball(5)
        For j = 0 To 5
            myColorBig = CInt(Rnd() * 5)
            Ball(j) = New ObjectShape.ObjectShape 'set shape, size for
balls
            With Ball(j)
                .Left = j * 50 + 50
                .Top = 100
                .Width = 20
                .Height = 20
                .Text.Format(j)
                .chgColor = colorChoiceBig(myColorBig)
                .chgShape = .Shapes.Ellipse
                Randomize()
                .vx = 2 * CInt(Int((2 * Rnd()) - 1)) + j - 3
                Randomize()
                .vy = 2 * CInt(Int((2 * Rnd()) - 1)) + j - 3
            End With
            PictureBox1.Controls.Add(Ball(j))

        Next

        navBall = New ObjectShape.ObjectShape  'create user controlled
object
        With navBall
            .Left = PictureBox1.Width / 2
            .Top = 10
            .Width = 30
            .Height = 30
            .chgColor = Color.Red
            .chgShape = .Shapes.Ellipse
```

```
        End With
        PictureBox1.Controls.Add(navBall)
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        Dim rect(5), rect2(5) As Rectangle
        Dim navRect, navRect2 As Rectangle
        Dim i, j, k, temp, newLeft, newTop, newleft1, newtop1, mycolor
       As  Integer
        colorChoice(0) = System.Drawing.Color.Yellow
        colorChoice(1) = System.Drawing.Color.Red
        colorChoice(2) = System.Drawing.Color.DarkCyan
        colorChoice(3) = System.Drawing.Color.DeepPink
        colorChoice(4) = System.Drawing.Color.DarkTurquoise

        With navBall 'create user controlled object
            navRect = New Rectangle(.Left, .Top, .Width, .Height)
        End With

        For j = 0 To 5 'create balls
            With Ball(j)
                rect(j) = New Rectangle(.Left, .Top, .Width, .Height)
                moveAround(Ball(j))

                If navRect.IntersectsWith(rect(j)) Then
                    newLeft = (navRect.Left + rect(j).Left) / 2
                    newTop = (navRect.Top + rect(j).Top) / 2
                    myWav.WinWav.PlayWavResource("DRUMROLL.WAV")
                    navBall.Dispose()

                    ReDim smallBall(5) 'create small balls
                    For k = 0 To 5

                        Randomize()
                        mycolor = CInt(Rnd() * 4)
                        smallBall(k) = New ObjectShape.ObjectShape
                        With smallBall(k)
                            .Left = newLeft
                            .Top = newTop
                            .Width = 8
                            .Height = 8
                            .chgColor = colorChoice(mycolor)
                            .chgShape = .Shapes.Ellipse
                            .vx = 2 * CInt(Int((2 * Rnd()) - 1)) + k - 3
                            .vy = 2 * CInt(Int((2 * Rnd()) - 1)) + k - 3
                        End With

                        PictureBox1.Controls.Add(smallBall(k))
                    Next

                    Timer1.Stop()
                    Timer2.Interval = 10
                    Timer2.Start()
```

```
            End If
        End With
    Next
End Sub

Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer2.Tick
    Dim j, k As Integer

    For j = 0 To 5
        moveAround(Ball(j))
        moveAround(smallBall(j))
    Next
End Sub

Private Sub BtnBallCollision_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
    Timer3.Interval = 30
    Timer3.Start()
End Sub

Private Sub Timer3_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer3.Tick

    Dim rect() As Rectangle
    ReDim rect(5)
    Dim navRect, navRect2 As Rectangle
    Dim i, j, k, temp, newLeft, newTop, newleft1, newtop1, mycolor
   As          Integer
    With navBall
        navRect = New Rectangle(.Left, .Top, .Width, .Height)
    End With

    For j = 0 To 5
        With Ball(j)
            rect(j) = New Rectangle(.Left, .Top, .Width, .Height)
        End With
    Next

    For i = 0 To 5
        For j = 0 To 5 'check if balls collide with one another
            If ((i < j) And (rect(i).IntersectsWith(rect(j)))) Then
                myWav.WinWav.PlayWavResource("chimes.wav")
                Beep()
                With Ball(i)
                    .Top -= .vy
                    .Left -= .vx
                End With

                With Ball(j)
                    .Top -= .vy
                    .Left -= .vx
                End With
                collide(Ball(i), Ball(j))
```

```
                ElseIf navRect.IntersectsWith(rect(j)) Then
                    newLeft = (navRect.Left + rect(j).Left) / 2
                    newTop = (navRect.Top + rect(j).Top) / 2
                    navBall.Dispose()
                End If

            Next
        Next

        For j = 0 To 5
            moveAround(Ball(j))
        Next
    End Sub

    Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e
    As System.Windows.Forms.MouseEventArgs) Handles
    PictureBox1.MouseMove
        navBall.Left = e.X
        navBall.Top = e.Y
    End Sub

End Class
```

7 CONCLUSION/FUTURE WORK

Game programming can be a useful and powerful tool. It can be used in educational settings to help students understand abstract concepts through games, with which most students are familiar. Examples of introducing abstract concepts through game programming include OOP, data structures, principles of physics, chemistry and mathematics. We also believe that game programming can exert a positive influence on at risk students by providing them with much needed excitement and motivation to pursue higher education as was evident by our experience Additionally, game programming is also a good field with plenty of opportunities as a career choice. Several web sites such as International Game developers association(igda.org) provide detailed information on the type of jobs available in the gaming industry along with the salary ranges. In attempting to promote this idea of introducing game programming to students, our future work will concentrate on developing a set of tools for the most commonly used game programming techniques with a grounding in physics as applicable to the gaming environment.

REFERENCES

[Ai05] Art Institute of Portland, retrieved on July 10th, 2005 from: http://www.aipd.aii.edu/summerteen.html

[Seay,Scott97] Jerry Seay, Robert Scott, Education and Simulation/Gaming and Computers, 1997, retrieved on July 10th, 2005 from http://www.cofc.edu/~seay/cb/simgames.html

[Feldgen,Clua04] Maria Feldgen, Osvaldo Clua, "Games As A Motivation for Freshmen to Learn Programming", 34th ASEE/IEEE Frontiers in Education Conference, October 2004

[Lave,Wenger90] Lave, J., & Wenger, E. (1990). Situated Learning: Legitimate Periperal Participation. Cambridge, UK: Cambridge University Press.

[McLellan95] McLellan, H. (1995). Situated Learning Perspectives. Englewood Cliffs, NJ: Educational Technology Publications.

[Coleman etal05] Ron Coleman, Mary Krembs, Alan Labouseur, Jim Weir(2005), "Game Design & Programming Concentration Within the Computer Science Curriculum", ACM SIGCSE 2005, pg 545.

[Pleva04] Greg, Pleva. (2004), "Game programming and the Myths of Childs Play", Journal of Computing Sciences in Colleges, Volume 20, Issue 2, pg 126.

[Fernald05] Joseph Fernald (2005) "Game Development Schools – Part 1", on July 10th, 2005 from: http://www.gamedev.net/reference/business/features/schools1/

[Wikipedia05] online encyclopedia, retrieved on July 10th from http://en.wikipedia.org/wiki/Game_programming

[Masuch, Freudenberg02] Maic Masuch, Bert Freudnberg, Teaching 3D game programming, retrieved on July 10th from http://isgwww.cs.uni-magdeburg.de/~bert/publications/Masuch-2002-TCG.pdf

[Delaet etal] Marianne deLaet, Kuffner James, Slattery michael, Elizabeth Sweedyk, "Computer games and CS Education: Why and How", retrieved on July 10th from http://db.grinnell.edu/sigcse/sigcse2005/Program/viewAcceptedProposal.asp?sessionType=panel&sessionNumber=16

[Cunningham etal04] Steve Cunningham, Werner Hansmann, Cary laxer, Jiaoying Shi, "The Beginning Graphics course in Computer Science", Computer Graphics, Volume 38, Number 4, November 2004, pg 24

[Schach02] Stephen R. Schach (2002), Object Oriented and Classical Software Engineering, Fifth Edition, Chapter 3, pg66

[Coleman etal05] Ron Coleman, Mary Krembs, Alan Labouseur, Jim Weir(2005), "Game Design & Programming Concentration Within the Computer Science Curriculum", ACM SIGCSE 2005, pg 547.

[reachfortomorrow.org] retrieved on July 10th, 2005, from: http://www.reachfortomorrow.org/01about.htm

[soundfile] "Play an Embedded sound file", retrieved on July 10th, 2005, from: http://www.vb-helper.com/howto_net_play_embedded_sound.html

[igda.org] retrieved on July 10th, 2005, from: http://www.igda.org/breakingin/career_paths.htm

About the author:

Lakshmi Prayaga is a lecturer in the Computer Science Department, at the University of West Florida. She is the co-author of a text book, "Programming the Web with ColdFusion MX and XHTML". She has presented several papers in conferences and workshops such as ACMSE, ELearn, EdMedia and NCEI. Her research interests include building visual tools to serve as teaching aids, web server technologies and computers in education.