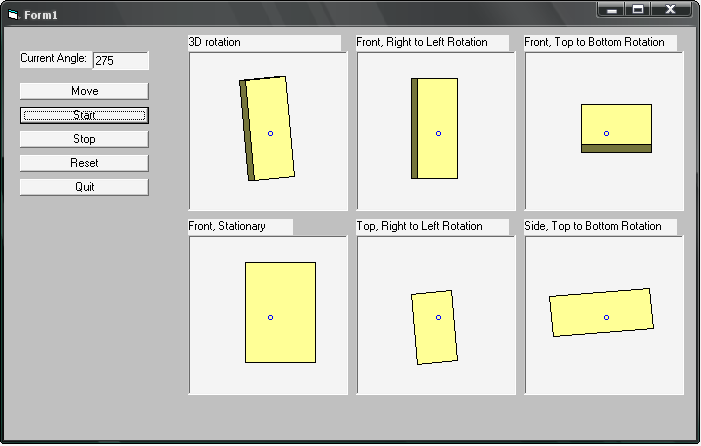
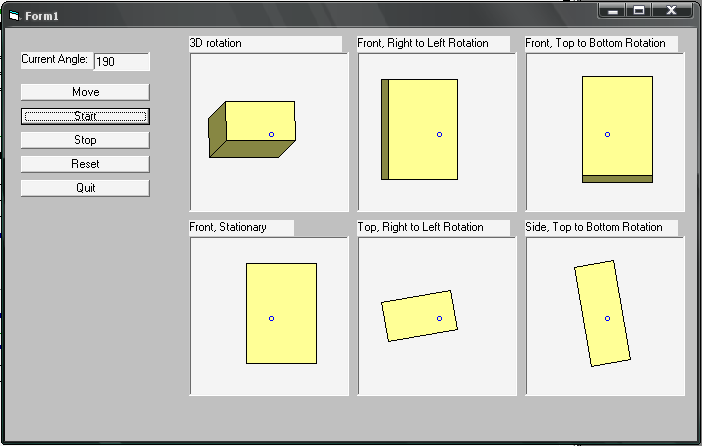
AUTOMATIC SYSTEM

CREATING 3D-GRAPHICS ON VISUAL BASIC

**Dushanbe, 2009**

**INTERFACE**



**SOURCE CODE**

Dim Angle As Double 'The rotation angle

Dim AngleHolder As Double 'holder for previous rotation angle

Dim NumObjectSides As Integer 'Number of sides making up the object

Private Type Point 'The makeup of a point

X As Double 'the X location of the point

Y As Double 'the Y location of the point

Z As Double 'the Z location of the point

End Type

Dim Center As Point 'center of the picboxes

Private Type Verticies 'The verticies of a side

NumPoints As Integer 'The number of points on a line

Points (20) As Point 'the actual endpoints of each line

Normal As Point 'The normal of the Plane

End Type

Dim Sides (50) As Verticies 'the sides of the object

Dim XSides (50) As Verticies 'the X rotation points

Dim YSides (50) As Verticies 'the Y rotation points

Dim ZSides (50) As Verticies 'the Z rotation points

Dim Sides3D (50) As Verticies 'the 3D rotation of points

Dim CosAng (359) As Double 'A lookup table to hold the Cosine Angles

Dim SinAng (359) As Double 'A lookup table to hold the Sine Angles

Private Type POINTAPI 'This is the drawn Points of the

X As Long 'object to fill it and draw it fast

Y As Long 'using a win api function

End Type

Dim tmp () As POINTAPI

'This function is for drawing filled polygons Much faster than anything I wrote

Private Declare Function Polygon Lib "gdi32" \_

(ByVal hdc As Long, lpPoint As POINTAPI, ByVal nCount As Long) As Long

Private Sub BUTNOKAY\_Click ()

'set the angle and draw the rotation

AngleHolder = AngleHolder + 5 'increment the angle

If AngleHolder = 360 Then 'reset the angle back to 0

AngleHolder = 0

End If

TEXTAngle. Text = AngleHolder 'display the current angle

Angle = AngleHolder 'Set the angle for calculations

Redraw 'refresh the display

End Sub

Private Sub BUTNQuit\_Click ()

End 'end the program

End Sub

Private Sub BUTNReset\_Click ()

AngleHolder = 355 'reset so displayed angle will be 0

BUTNOKAY\_Click 'set angles and displays, then redraw

End Sub

Private Sub BUTNStart\_Click ()

Timer1. Enabled = True 'start the autodraw timer

End Sub

Private Sub BUTNStop\_Click ()

Timer1. Enabled = False 'stop the auto draw timer

End Sub

Private Sub Form\_Load ()

'Form1. ScaleMode = vbTwips

Angle = 0 'initialize the angles

AngleHolder = 355

Center. X = Pic3D. Width / 2 'set the centers (all coordinates for the picboxes must be equal

Center. Y = Pic3D. Height / 2 'acroos picboxes for this to work) (i. e. the X dimension in left

Center. Z = Pic3D. Width / 2 'to right picbox must equal X dimension in top to bottom picbox)

'set points for rectangle (could be done in a better way loop etc)

'Also shape does not have to be rectangle can be any shape

'front

Sides (0). Points (0). X = - 20: Sides (0). Points (0). Y = - 50: Sides (0). Points (0). Z = 20

Sides (0). Points (1). X = 50: Sides (0). Points (1). Y = - 50: Sides (0). Points (1). Z = 20

Sides (0). Points (2). X = 50: Sides (0). Points (2). Y = 50: Sides (0). Points (2). Z = 20

Sides (0). Points (3). X = - 20: Sides (0). Points (3). Y = 50: Sides (0). Points (3). Z = 20

'back

Sides (1). Points (0). X = 50: Sides (1). Points (0). Y = - 50: Sides (1). Points (0). Z = - 20

Sides (1). Points (1). X = - 20: Sides (1). Points (1). Y = - 50: Sides (1). Points (1). Z = - 20

Sides (1). Points (2). X = - 20: Sides (1). Points (2). Y = 50: Sides (1). Points (2). Z = - 20

Sides (1). Points (3). X = 50: Sides (1). Points (3). Y = 50: Sides (1). Points (3). Z = - 20

'Top

Sides (2). Points (0). X = - 20: Sides (2). Points (0). Y = - 50: Sides (2). Points (0). Z = - 20

Sides (2). Points (1). X = 50: Sides (2). Points (1). Y = - 50: Sides (2). Points (1). Z = - 20

Sides (2). Points (2). X = 50: Sides (2). Points (2). Y = - 50: Sides (2). Points (2). Z = 20

Sides (2). Points (3). X = - 20: Sides (2). Points (3). Y = - 50: Sides (2). Points (3). Z = 20

'bottom

Sides (3). Points (0). X = - 20: Sides (3). Points (0). Y = 50: Sides (3). Points (0). Z = 20

Sides (3). Points (1). X = 50: Sides (3). Points (1). Y = 50: Sides (3). Points (1). Z = 20

Sides (3). Points (2). X = 50: Sides (3). Points (2). Y = 50: Sides (3). Points (2). Z = - 20

Sides (3). Points (3). X = - 20: Sides (3). Points (3). Y = 50: Sides (3). Points (3). Z = - 20

'Lside

Sides (4). Points (0). X = - 20: Sides (4). Points (0). Y = - 50: Sides (4). Points (0). Z = - 20

Sides (4). Points (1). X = - 20: Sides (4). Points (1). Y = - 50: Sides (4). Points (1). Z = 20

Sides (4). Points (2). X = - 20: Sides (4). Points (2). Y = 50: Sides (4). Points (2). Z = 20

Sides (4). Points (3). X = - 20: Sides (4). Points (3). Y = 50: Sides (4). Points (3). Z = - 20

'Rside

Sides (5). Points (0). X = 50: Sides (5). Points (0). Y = - 50: Sides (5). Points (0). Z = 20

Sides (5). Points (1). X = 50: Sides (5). Points (1). Y = - 50: Sides (5). Points (1). Z = - 20

Sides (5). Points (2). X = 50: Sides (5). Points (2). Y = 50: Sides (5). Points (2). Z = - 20

Sides (5). Points (3). X = 50: Sides (5). Points (3). Y = 50: Sides (5). Points (3). Z = 20

'set the number of edges for each side

For i = 0 To 5

Sides (i). NumPoints = 3

Next i

'set the number of sides the object has

NumObjectSides = 5

'Calculate the Normals

FindNormals

'Create the Lookup table for the Cos and Sin functions.

'This method is much faster than calculating each step

CreateTables

'set angles and displays, then redraw

BUTNOKAY\_Click

End Sub

Sub Redraw ()

'clear the picboxes

PicXY. Cls

PicFRL. Cls

PicTop. Cls

PicFTB. Cls

PicSTB. Cls

Pic3D. Cls

'draw the front of the box in the stationary view

DrawShape Sides (0), PicXY, "FRONT"

'repeat loop 6 times once for each side the rotation of each point must

'be calculated to find the new position of the normal of each side to

'determine if it's visible

For j = 0 To 5

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'draw the points for a top to bottom rotation (rotation around

'the X axis)

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

For i = 0 To Sides (0). NumPoints

XSides (j). NumPoints = Sides (0). NumPoints

XSides (j). Points (i). X = Sides (j). Points (i). X 'new x value

XSides (j). Points (i). Y = Sides (j). Points (i). Y \* CosAng (Angle) - Sides (j). Points (i). Z \* SinAng (Angle) 'new y value

XSides (j). Points (i). Z = Sides (j). Points (i). Z \* CosAng (Angle) + Sides (j). Points (i). Y \* SinAng (Angle) 'new z value

XSides (j). Normal. X = Sides (j). Normal. X

XSides (j). Normal. Y = Sides (j). Normal. Y \* CosAng (Angle) - Sides (j). Normal. Z \* SinAng (Angle) 'new y value

XSides (j). Normal. Z = Sides (j). Normal. Z \* CosAng (Angle) + Sides (j). Normal. Y \* SinAng (Angle) 'new z value

Next i

'check to see if plane is visible if so draw it

If VisiblePlane (XSides (j), 0, 0, 1000) Then

'Draw lines in top, top to bottom rotation

DrawShape XSides (j), PicFTB, "FRONT"

End If

If VisiblePlane (XSides (j), 1000, 0, 0) Then

'Draw points in side, top to bottom rotation view

DrawShape XSides (j), PicSTB, "SIDE"

End If

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'draw the points for a left to right rotation (rotation around

'the Y axis)

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

For i = 0 To Sides (0). NumPoints

YSides (j). NumPoints = Sides (0). NumPoints

YSides (j). Points (i). X = Sides (j). Points (i). X \* CosAng (Angle) + Sides (j). Points (i). Z \* SinAng (Angle) 'new x value

YSides (j). Points (i). Y = Sides (j). Points (i). Y 'new y value

YSides (j). Points (i). Z = Sides (j). Points (i). Z \* CosAng (Angle) - Sides (j). Points (i). X \* SinAng (Angle) 'new z value

YSides (j). Normal. X = Sides (j). Normal. X \* CosAng (Angle) + Sides (j). Normal. Z \* SinAng (Angle) 'new x value

YSides (j). Normal. Y = Sides (j). Normal. Y 'new y value

YSides (j). Normal. Z = Sides (j). Normal. Z \* CosAng (Angle) - Sides (j). Normal. X \* SinAng (Angle) 'new z value

Next i

'check to see if plane is visible if so draw it

If VisiblePlane (YSides (j), 0, 0, 1000) Then

'Draw lines in front right to left rotation view

DrawShape YSides (j), PicFRL, "FRONT"

End If

If VisiblePlane (YSides (j), 0, 1000, 0) Then

'Draw lines in top right to left rotation view

DrawShape YSides (j), PicTop, "TOP"

End If

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'draw the points for a sideways rotation (rotation around

'the Z axis)

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'Remove comments to do calculation

'Rotate Z direction

'For i = 0 To Sides (0). NumPoints

' ZSides (j). NumPoints = Sides (0). NumPoints

' ZSides (j). Points (i). X = Sides (j). Points (i). X \* CosAng (Angle) + Sides (j). Points (i). Y \* SinAng (Angle) 'new x value

' ZSides (j). Points (i). Y = Sides (j). Points (i). Y \* CosAng (Angle) - Sides (j). Points (i). X \* SinAng (Angle) 'new y value

' ZSides (j). Points (i). Z = Sides (j). Points (i). Z 'new z value

' ZSides (j). Normal. X = Sides (j). Normal. X \* CosAng (Angle) + Sides (j). Normal. Y \* SinAng (Angle) 'new x value

' ZSides (j). Normal. Y = Sides (j). Normal. Y \* CosAng (Angle) - Sides (j). Normal. X \* SinAng (Angle) 'new y value

' ZSides (j). Normal. Z = Sides (j). Normal. Z 'new z value

'Next i

'Rotate values rotated in X direction in Z direction to make "spinning effect"

For i = 0 To Sides (0). NumPoints

Sides3D (j). NumPoints = Sides (0). NumPoints

Sides3D (j). Points (i). X = XSides (j). Points (i). X \* CosAng (Angle) + XSides (j). Points (i). Y \* SinAng (Angle) 'new x value

Sides3D (j). Points (i). Y = XSides (j). Points (i). Y \* CosAng (Angle) - XSides (j). Points (i). X \* SinAng (Angle) 'new y value

Sides3D (j). Points (i). Z = XSides (j). Points (i). Z 'new z value

Sides3D (j). Normal. X = XSides (j). Normal. X \* CosAng (Angle) + XSides (j). Normal. Y \* SinAng (Angle) 'new x value

Sides3D (j). Normal. Y = XSides (j). Normal. Y \* CosAng (Angle) - XSides (j). Normal. X \* SinAng (Angle) 'new y value

Sides3D (j). Normal. Z = XSides (j). Normal. Z 'new z value

Next i

'check to see if plane is visible if so draw it

If VisiblePlane (Sides3D (j), 0, 1000, 0) Then

'Draw the 2 direction rotation

DrawShape Sides3D (j), Pic3D, "TOP"

End If

Next j

'draw centerpoint of each picbox in Blue

PicXY. Circle (Center. X, Center. Y), 30, RGB (0, 0, 255)

PicFRL. Circle (Center. X, Center. Y), 30, RGB (0, 0, 255)

PicTop. Circle (Center. X, Center. Y), 30, RGB (0, 0, 255)

PicFTB. Circle (Center. X, Center. Y), 30, RGB (0, 0, 255)

PicSTB. Circle (Center. X, Center. Y), 30, RGB (0, 0, 255)

Pic3D. Circle (Center. X, Center. Y), 30, RGB (0, 0, 255)

End Sub

Private Sub Timer1\_Timer ()

'rotate the rectangle

BUTNOKAY\_Click

End Sub

Private Function DrawShape (shape As Verticies, PicBox As PictureBox, View As String)

' add 75 to all points to center object

'determine view

If View = "FRONT" Then

'create lppoints for the win func call

ReDim tmp (shape. NumPoints) As POINTAPI

'fill in the drawing points tmp. x as the value going in the x dir etc

For i = 0 To shape. NumPoints

tmp (i). X = shape. Points (i). X + 75

tmp (i). Y = shape. Points (i). Y + 75

Next i

'Draw solid polygons

'calculate light value (ambient + Max \* (normal of plane \* light position)

Colr = 100 + 200 \* (shape. Normal. Z)

'Fill object as solid

PicBox. FillStyle = 0

'Choose the color (this way makes a shade of yellow)

PicBox. FillColor = RGB (Colr, Colr, Colr / 2)

'draw the polygon

Polygon PicBox. hdc, tmp (0), shape. NumPoints + 1

'draw rest of objects transparently

PicBox. FillStyle = 1

ElseIf View = "TOP" Then

'creat lppoints for the win func call

ReDim tmp (shape. NumPoints) As POINTAPI

'fill in the drawing points tmp. x as the value going in the x dir etc

For i = 0 To shape. NumPoints

tmp (i). X = shape. Points (i). X + 75

tmp (i). Y = shape. Points (i). Z + 75

Next i

'Draw solid polygons

'calculate light value (ambient + Max \* (normal of plane \* light position)

Colr = 100 + 200 \* (shape. Normal. Y)

'Fill object as solid

PicBox. FillStyle = 0

'Choose the color (this way makes a shade of yellow)

PicBox. FillColor = RGB (Colr, Colr, Colr / 2)

'draw the polygon

Polygon PicBox. hdc, tmp (0), shape. NumPoints + 1

'draw rest of objects transparently

PicBox. FillStyle = 1

ElseIf View = "SIDE" Then

'creat lppoints for the win func call

ReDim tmp (shape. NumPoints) As POINTAPI

'fill in the drawing points tmp. x as the value going in the x dir etc

For i = 0 To shape. NumPoints

tmp (i). X = shape. Points (i). Z + 75

tmp (i). Y = shape. Points (i). Y + 75

Next i

'Draw solid polygons

'calculate color value (ambient + Max \* (normal of plane \* light position)

Colr = 100 + 200 \* (shape. Normal. X)

'Fill object as solid

PicBox. FillStyle = 0

'Choose the color (this way makes a shade of yellow)

PicBox. FillColor = RGB (Colr, Colr, Colr / 2)

'draw the polygon

Polygon PicBox. hdc, tmp (0), shape. NumPoints + 1

'draw rest of objects transparently

PicBox. FillStyle = 1

End If

End Function

Private Function VisiblePlane (shape As Verticies, CameraX As Integer, CameraY As Integer, CameraZ As Integer)

'this function takes the normal of the plane and returns True if visible FALSE if

'not visible

'Camera is the spot the object is being viewed from

'Find the dot product

D = (shape. Normal. X \* CameraX) + (shape. Normal. Y \* CameraY) + (shape. Normal. Z \* CameraZ)

'return true if object is visible

VisiblePlane = D >= 0

End Function

Private Function FindNormals ()

'This function finds the normal of each plane

For i = 0 To NumObjectSides

'Find the normal vector

With Sides (i)

' \* \* \* \* \* \* \* \*

Nx = (. Points (1). Y - Points (0). Y) \* (. Points (. NumPoints). Z - Points (0). Z) - (. Points (1). Z - Points (0). Z) \* (. Points (. NumPoints). Y - Points (0). Y)

Ny = (. Points (1). Z - Points (0). Z) \* (. Points (. NumPoints). X - Points (0). X) - (. Points (1). X - Points (0). X) \* (. Points (. NumPoints). Z - Points (0). Z)

Nz = (. Points (1). X - Points (0). X) \* (. Points (. NumPoints). Y - Points (0). Y) - (. Points (1). Y - Points (0). Y) \* (. Points (. NumPoints). X - Points (0). X)

'Normalize the normal vector (make length of 1)

Length = Sqr (Nx ^ 2 + Ny ^ 2 + Nz ^ 2)

. Normal. X = Nx / Length

. Normal. Y = Ny / Length

. Normal. Z = Nz / Length

End With

Next i

End Function

Private Function CreateTables ()

'Create cosine and sine lookup table

For i = 0 To 359

CosAng (i) = Cos (i \* (3.14159265358979/180)) 'convert degrees to radians

SinAng (i) = Sin (i \* (3.14159265358979/180)) 'convert degrees to radians

Next i

End Function

**Main Interface**

