

Oblique Projection

3D Computer Graphics and Animation Programming Assignment 2



Ida Bagus Bhaskara (001201500016)

Vera Debora Vitamas (001201500076)

Jonathan Surya Sandjaya (001201500055)

CIT 2 2015

**Table of Contents**

Cover Page 1

**Table of Contents 2**

[Chapter 1 Introduction 2](#_Toc506148694)

[Chapter 2 Basic Theory 2](#_Toc506148695)

[2.1 Animation 2](#_Toc506148696)

[Chapter 3 Implementation 3](#_Toc506148697)

[3.1 Main Interface of the Application 3](#_Toc506148698)

[3.2 Features of the Applicaton 4](#_Toc506148699)

[Chapter 4 Design 4](#_Toc506148700)

[4.1 Variables Used in the Program 4](#_Toc506148701)

[4.2 Representation of Edge and Point in the Application 5](#_Toc506148702)

[4.3 Initializing the Cube 6](#_Toc506148703)

[4.4 Setting the Edge 7](#_Toc506148704)

[4.5 Setting the Point 7](#_Toc506148705)

[4.6 Drawing the Cube 7](#_Toc506148706)

[4.7 Hiding the Cube 7](#_Toc506148707)

[4.8 Rotation Tick Method 8](#_Toc506148708)

[4.9 Setting the Matrix 9](#_Toc506148709)

[4.10 Multiplying the Matrix 9](#_Toc506148710)

[Chapter 5 Evaluation 10](#_Toc506148711)

[5.1 Display the Oblique Projection with Different Value of Phi 10](#_Toc506148712)

[5.2 Display the Oblique Projections with Different Values of Alpha 11](#_Toc506148713)

[Chapter 6 Work Log 12](#_Toc506148714)

[Chapter 7 Conclusion and Remarks 13](#_Toc506148715)

# **Introduction**

1. About the Application

This is a 3D oblique projection simulator. It applies the oblique projection which uses computer graphic animation algorithms to rotate and to create an oblique view of a cube. This simulator allows the user to initialize the alpha and theta of the view, rotating the cube in 3 different coordinates (, and ), and stop the cube rotation.

This program uses Microsoft Visual Studio as the programming platform and Visual Basic as the programming language. This report covers the basic theory, implementation, design, evalution, work log, and conclusion.

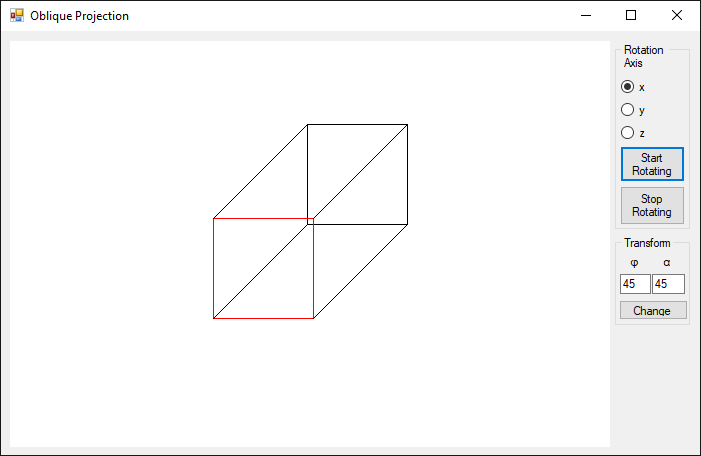
# **Basic Theory**

## Animation

# **Implementation**

## Main Interface of the Application

This application has one interface, which it uses as its main interface. The interface is pictured below:



7

6

5

4

3

2

1

The interface consists of the following components:

1. Canvas Picture Box

The canvas is used to display the cube using the application.

1. Rotation Axis Radio Buttons

The radio buttons are used to change the , and coordinate of cube rotation.

1. Start Rotating Button

The button is used to rotate the cube in the canvas.

1. Stop Rotating Button

The button is used to stop the rotation of the cube.

1. Phi Textbox

The input box is used to determine the value of the phi.

1. Alpha Textbox

The input box is used to determine the value of the alpha.

1. Change Button

The change button is used to change the value of alpha and theta of the cube based on the value entered on the textboxes.

## Features of the Applicaton

The application has several features, namely:

1. Rotating the 3D Cube Around the , and Axes

To rotate the cube, the user must choose 1 out of 3 axes shown in the radio button and then click the start rotating button to rotate the 3D cube.

1. Changing the Values of Alpha () and Phi ()

To change the value of alpha and theta the user must input the values in the textboxes and click change button.

1. Stopping the Rotation of the Cube

To stop the cube rotation the user must click the stop rotating button.

# Design

## Variables Used in the Program

The following variables are used in this program:

* Dim graphics As Graphics  
  This variable is used to pass the canvas variable to the Graphics class so that functions from the Graphics class, for example the DrawLine function, can be used to draw the cube’s edges in PictureBox1, which acts as the canvas.
* Dim canvas As Bitmap  
  This variable is used to initialize the canvas as a New Bitmap with PictureBox1’s width and height as its properties.
* Dim phi As Single

This variable used to store the value of the cube.

* Dim alpha As Single

This variable used to store the value of the cube.

* Dim vertex(7) As Point

This array variable is used to store the cube’s vertices.

* Dim edges(12) As Edge

This array variable is used to store the cube’s edges.

* Dim view(3, 3), screen(3, 3) As Single

These 2D array variable is used to store the and row matrices.

* Dim VR(7), VS(7) As Point

These 2D array variable is used to store the result of multiplying the cube’s vertex with the and matrices respectively.

* Dim deg As Single = 0

This variable is used to store the degree that is used to calculate the rotation matrix and will be incremented each tick in the RotationTick\_Tick.

## Representation of Edge and Point in the Application

The edge and point are both represented as a Structure in this application. Inside Structure Edge there are two variables defined, namely point1 and point2. These variables will define the two vertices that creates an edge. There is one variable that is defined as Structure Edge in this application, and that is the edges(12) variable.

Structure Point consists of four variables, namely x, y, z, w. These variables will define the 1x4 matrix that determine the position of the vertex. There are three variables that are defined as Structure Point in this application, they are vertex(7), VR(7), and VS(7).

## Initializing the Cube

The cube is initialized by the Sub Init. The Sub initalized the cube by calling several Subs, namely SetPoint, SetColMat, SetEdge, MultiplyMat, and DrawCube. First, the Sub fills the vertex array using SetPoint. Then, it sets the and matrix column-by-row with SetColMat. Edges are then set using SetEdge. It then multiplies the vertices with and by multiplying each vertex with and using MultiplyMat and looping it for 8 times. It then calls DrawCube to show the 3D cube on the canvas.

Sub Init(phi As Single, alpha As Single)

SetPoint(vertex(0), -1, -1, 1)

SetPoint(vertex(1), 1, -1, 1)

SetPoint(vertex(2), 1, 1, 1)

SetPoint(vertex(3), -1, 1, 1)

SetPoint(vertex(4), -1, -1, -1)

SetPoint(vertex(5), 1, -1, -1)

SetPoint(vertex(6), 1, 1, -1)

SetPoint(vertex(7), -1, 1, -1)

SetColMat(screen, 0, 50, 0, 0, 300)

SetColMat(screen, 1, 0, -50, 0, 180)

SetColMat(screen, 2, 0, 0, 0, 0)

SetColMat(screen, 3, 0, 0, 0, 1)

SetColMat(view, 0, 1, 0, (CotDegree(phi) \* CosDegree(alpha)) \* 2, 0)

SetColMat(view, 1, 0, 1, (CotDegree(phi) \* SinDegree(alpha)) \* 2, 0)

SetColMat(view, 2, 0, 0, 0, 0)

SetColMat(view, 3, 0, 0, 0, 1)

SetEdge(edges(0), 0, 1)

SetEdge(edges(1), 1, 2)

SetEdge(edges(2), 2, 3)

SetEdge(edges(3), 3, 0)

SetEdge(edges(4), 4, 5)

SetEdge(edges(5), 5, 6)

SetEdge(edges(6), 6, 7)

SetEdge(edges(7), 7, 4)

SetEdge(edges(8), 0, 4)

SetEdge(edges(9), 1, 5)

SetEdge(edges(10), 2, 6)

SetEdge(edges(11), 3, 7)

For i = 0 To 7

VR(i) = MultiplyMat(vertex(i), view)

VS(i) = MultiplyMat(VR(i), screen)

Next

DrawCube()

End Sub

## Setting the Edge

The edge is defined by the edge variable, which is defined as Structure Edge. It accepts three parameters; the edge array, the first point, and the second point.

Sub SetEdge(ByRef edge As Edge, n1 As Integer, n2 As Integer)

edge.point1 = n1

edge.point2 = n2

End Sub

## Setting the Vertex

The vertex is defined by the vertex variable, which is defined as Structure Point. It accepts four parameters; the vertex array and the x,y, and z axis position of the vertex.

Sub SetPoint(ByRef point As Point, x As Integer, y As Integer, z As Integer)

point.x = x

point.y = y

point.z = z

point.w = 1

End Sub

## Drawing the Cube

The cube is drawn using DrawCube. Showing the front side of the cube as a red-outlined square is done by drawing the first four edges stored in the edges array black and then the next four red. The cube is then completed by drawing the rest of the edges black.

Sub DrawCube()

Dim i, j, k As Integer

For i = 0 To 3

graphics.DrawLine(Pens.Black, VS(edges(i).point1).x, VS(edges(i).point1).y, VS(edges(i).point2).x, VS(edges(i).point2).y)

Next

For j = 4 To 7

graphics.DrawLine(Pens.Red, VS(edges(j).point1).x, VS(edges(j).point1).y, VS(edges(j).point2).x, VS(edges(j).point2).y)

Next

For k = 8 To 11

graphics.DrawLine(Pens.Black, VS(edges(k).point1).x, VS(edges(k).point1).y, VS(edges(k).point2).x, VS(edges(k).point2).y)

Next

PictureBox1.Image = canvas

End Sub

## Hiding the Cube

Hiding the cube is done using the DrawLine method, similar to DrawCube. But instead of drawing the edges black, in this method the cube is drawn white; similar to the canvas.

Sub HideCube()

Dim i As Integer

For i = 0 To 11

graphics.DrawLine(Pens.White, VS(edges(i).point1).x, VS(edges(i).point1).y, VS(edges(i).point2).x, VS(edges(i).point2).y)

Next

PictureBox1.Image = canvas

End Sub

## Setting the Matrix

In this application, matrices are defined column-by-column. Defining the matrix is done via Sub SetColMat. It accepts six parameters; the transformation matrix, the column index, and the values to be set into the matrix, which is a, b, c, and d value.

Sub SetColMat(ByRef Matrix(,) As Single, col As Integer, a As Double, b As Double, c As Double, d As Double)

Matrix(0, col) = a

Matrix(1, col) = b

Matrix(2, col) = c

Matrix(3, col) = d

End Sub

## Multiplying the Matrix

Multiplying the matrix is done via Function MultiplyMat and returns Point. The function accepts two parameters, namely the vertex and the multiplier matrix. It multiplies the vertex with the matrix column-by-row.

Function MultiplyMat(point As Point, M(,) As Single) As Point

Dim result As Point

result.x = (point.x \* M(0, 0) + point.y \* M(1, 0) + point.z \* M(2, 0) + point.w \* M(3, 0))

result.y = (point.x \* M(0, 1) + point.y \* M(1, 1) + point.z \* M(2, 1) + point.w \* M(3, 1))

result.z = (point.x \* M(0, 2) + point.y \* M(1, 2) + point.z \* M(2, 2) + point.w \* M(3, 2))

result.w = 1

Return result

End Function

## Rotation Tick Method

In this application, the cube can be rotated based on the axis the user selected. The cube is then rotated per tick. This means that the cube changes it degree of rotation every tick. At the beginning of the tick, it hid the cube beforehand using HideCube. It then increments the deg variable by 5. Before setting the Rotation matrix, it checks which axis radio button is currenly selected by the user, whether it’s the x, y, or the z axis radio button. It then multiplies the vertices with , and by multiplying each vertex with and using MultiplyMat and looping it for 8 times. It then calls DrawCube to show the 3D cube on the canvas. It also refreshes the canvas after it has drawn the cube.

Private Sub RotationTick\_Tick(sender As Object, e As EventArgs) Handles RotationTick.Tick

Dim Rot(3, 3) As Single

HideCube()

deg = deg + 5

If XButton.Checked = True Then

SetColMat(Rot, 0, 1, 0, 0, 0)

SetColMat(Rot, 1, 0, CosDegree(deg), -SinDegree(deg), 0)

SetColMat(Rot, 2, 0, SinDegree(deg), CosDegree(deg), 0)

SetColMat(Rot, 3, 0, 0, 0, 1)

ElseIf YButton.Checked = True Then

SetColMat(Rot, 0, CosDegree(deg), 0, SinDegree(deg), 0)

SetColMat(Rot, 1, 0, 1, 0, 0)

SetColMat(Rot, 2, -SinDegree(deg), 0, CosDegree(deg), 0)

SetColMat(Rot, 3, 0, 0, 0, 1)

ElseIf ZButton.Checked = True Then

SetColMat(Rot, 0, CosDegree(deg), -SinDegree(deg), 0, 0)

SetColMat(Rot, 1, SinDegree(deg), CosDegree(deg), 0, 0)

SetColMat(Rot, 2, 0, 0, 1, 0)

SetColMat(Rot, 3, 0, 0, 0, 1)

End If

For i = 0 To 7

VR(i) = MultiplyMat(vertex(i), Rot)

VR(i) = MultiplyMat(VR(i), view)

VS(i) = MultiplyMat(VR(i), screen)

Next

DrawCube()

PictureBox1.Refresh()

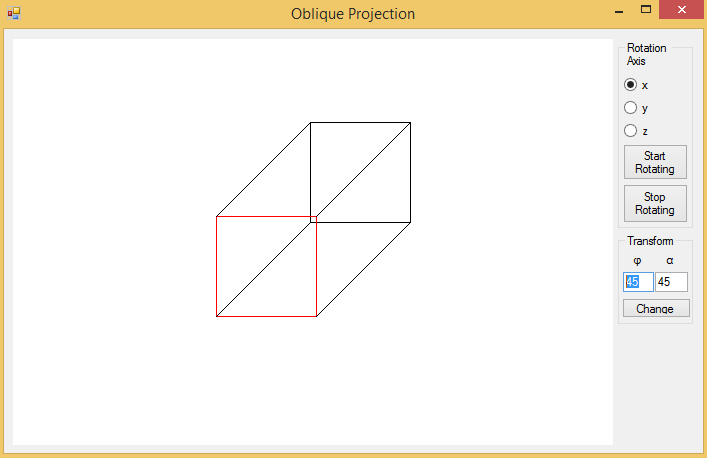
End Sub

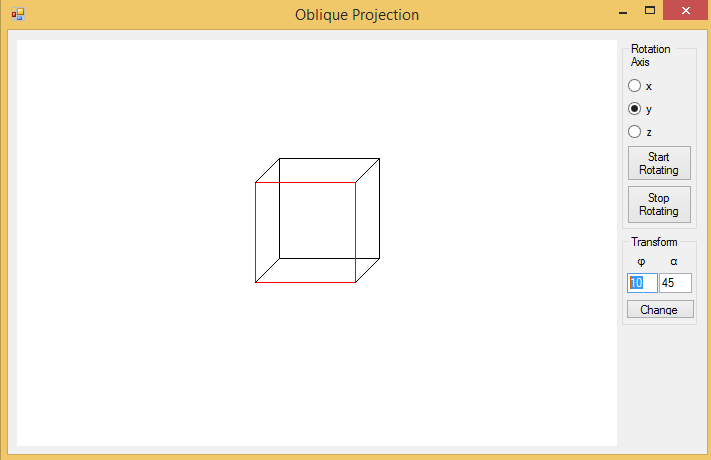
# Evaluation

To ensure that the application works as it is designed, test cases are conducted. These test cases are as follows:

## Display the Oblique Projection with Different Value of Phi

In this case, the user tries to change the theta value in the projection.

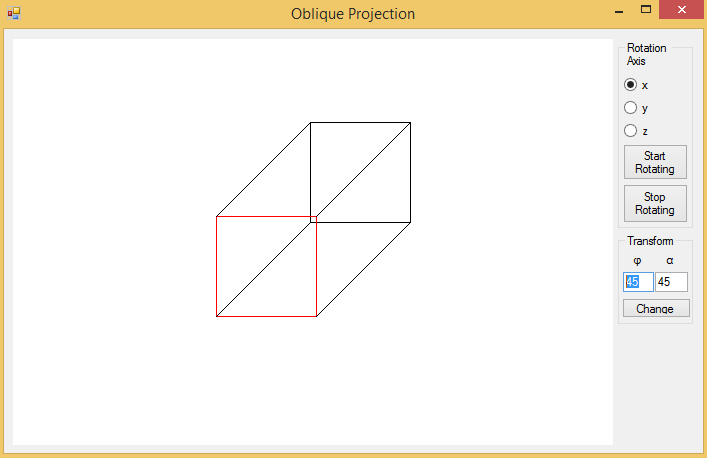


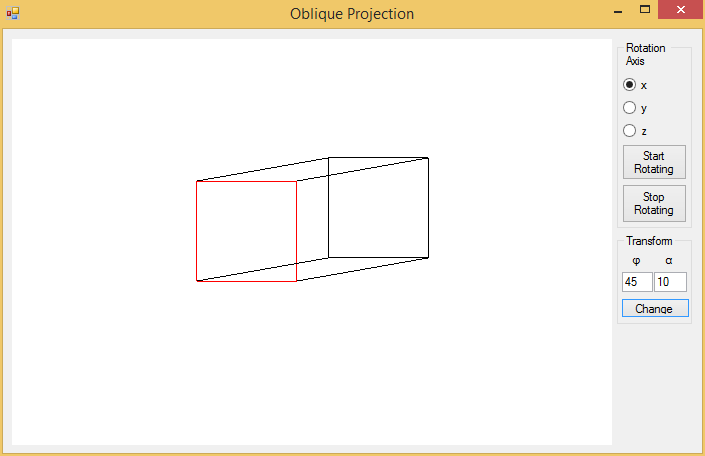


This case is successful because the program can change the view of the cube.

## Display the Oblique Projections with Different Values of Alpha

In this case, the user tries to change the alpha value in the projection.





This case is successful because the program can change the view of the cube.

# Work Log

The work log is extracted directly from Visual Studio’s Git Log History, which is also available publicly at <https://github.com/bakanui/ObliqueProjection/commits/master>.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Author | Date | Time | Commit Message |
| 1 | Bhaskara Ida Bagus | 1/26/2018 | 5:39:49 PM | Initialized project |
| 2 | Vera Debora Vitamas | 1/29/2018 | 1:57:02 PM | Trying to create matrix structure |
| 3 | Vera Debora Vitamas | 1/29/2018 | 3:30:21 PM | Trying Matrix Multiplication |
| 4 | Vera Debora Vitamas | 1/29/2018 | 7:05:34 PM | Trying to initialize cube |
| 5 | Bhaskara Ida Bagus | 2/2/2018 | 11:05:35 PM | Add report template |
| 6 | Bhaskara Ida Bagus | 2/3/2018 | 3:24:05 PM | Buggy cube; shows a square instead of a cube |
| 7 | Bhaskara Ida Bagus | 2/3/2018 | 3:50:18 PM | Square is now a Cube, but not Oblique |
| 8 | Jonathan Surya Sandjaya | 2/4/2018 | 8:12:53 PM | Cube is now oblique |
| 9 | Bhaskara Ida Bagus | 2/4/2018 | 9:14:46 PM | Phi and Theta can now be set by user, buggy rotation |
| 10 | Bhaskara Ida Bagus | 2/4/2018 | 10:19:30 PM | Declaring DegToRad as a function |
| 11 | Bhaskara Ida Bagus | 2/5/2018 | 11:26:05 PM | Rotation in the x axis works |
| 12 | Bhaskara Ida Bagus | 2/6/2018 | 6:30:29 PM | Beautifying UI, Rotation on x, y, and z axis can now be done |
| 13 | Bhaskara Ida Bagus | 2/6/2018 | 6:45:40 PM | Fix Rot Matrix, Renaming theta to alpha |
| 14 | Bhaskara Ida Bagus | 2/7/2018 | 1:43:12 AM | Update work log in report |
| 15 | Jonathan Surya Sandjaya | 2/7/2018 | 7:06:55 PM | creating report |
| 16 | Jonathan Surya Sandjaya | 2/11/2018 | 7:03:14 PM | Add reset button |
| 17 | Jonathan Surya Sandjaya | 2/11/2018 | 7:33:47 PM | update report |

# Conclusion and Remarks

The program works well. The 3D cube rotation, oblique projection, and reset work as expected. This can be proven by the success of the program in conducting each test case. Overall, this program doesn’t have anything bug.

Through this programming assignment, we learn some important things such as it’s definitely hard to do this programming assignment in 2 weeks , we can know the ability and work ethic of each member, how to divide the time efficiently between doing this programming assignment and other assignments, and the most important thing is through this programming assignment, we can improve our coding skill where we are forced to work harder in order to reach a goal in the certain limit time.