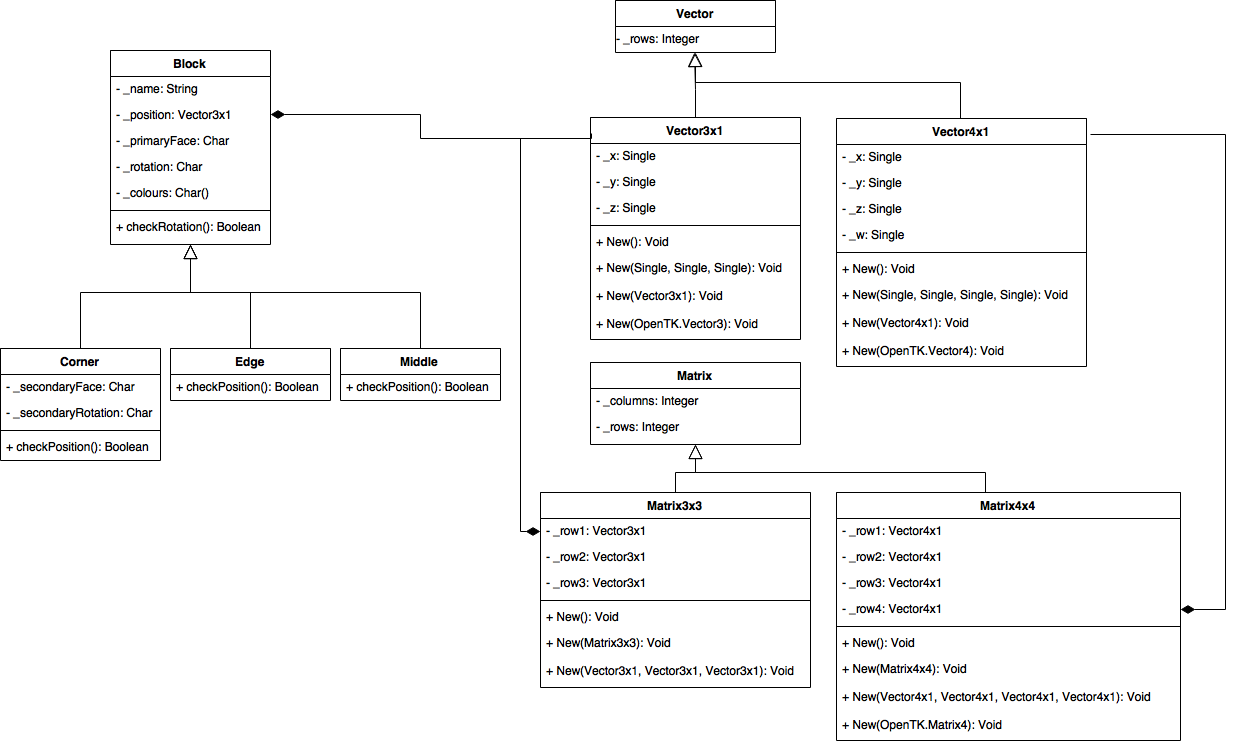
# Class Diagram

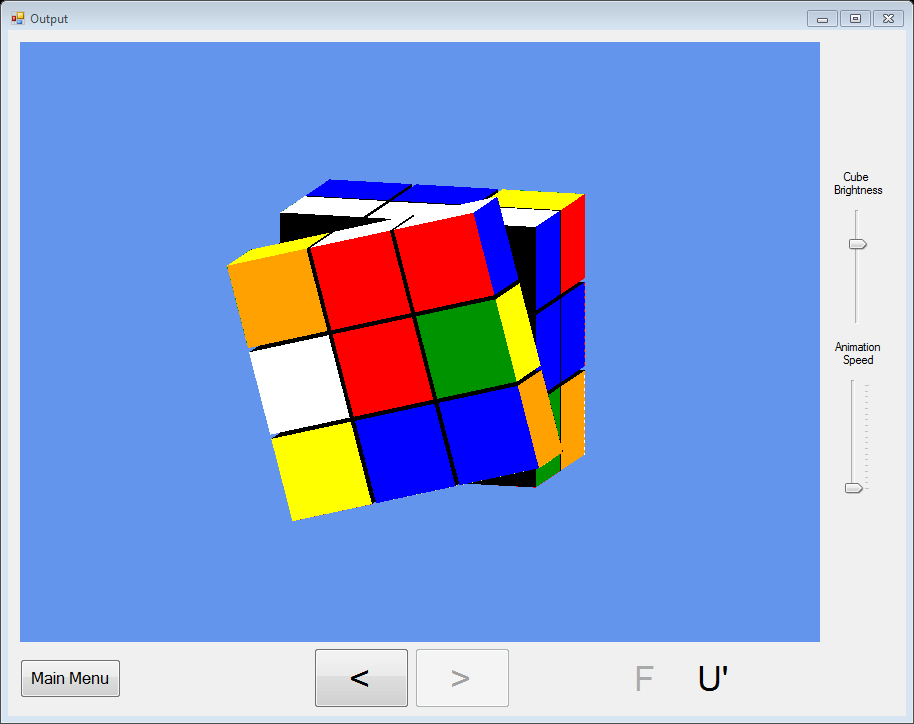


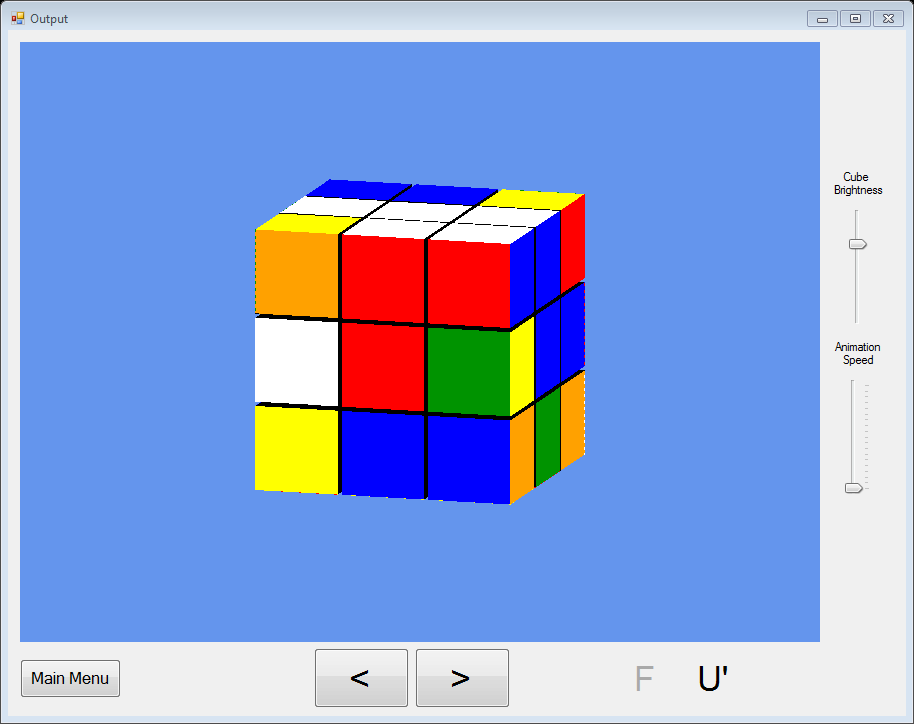
# Screenshots

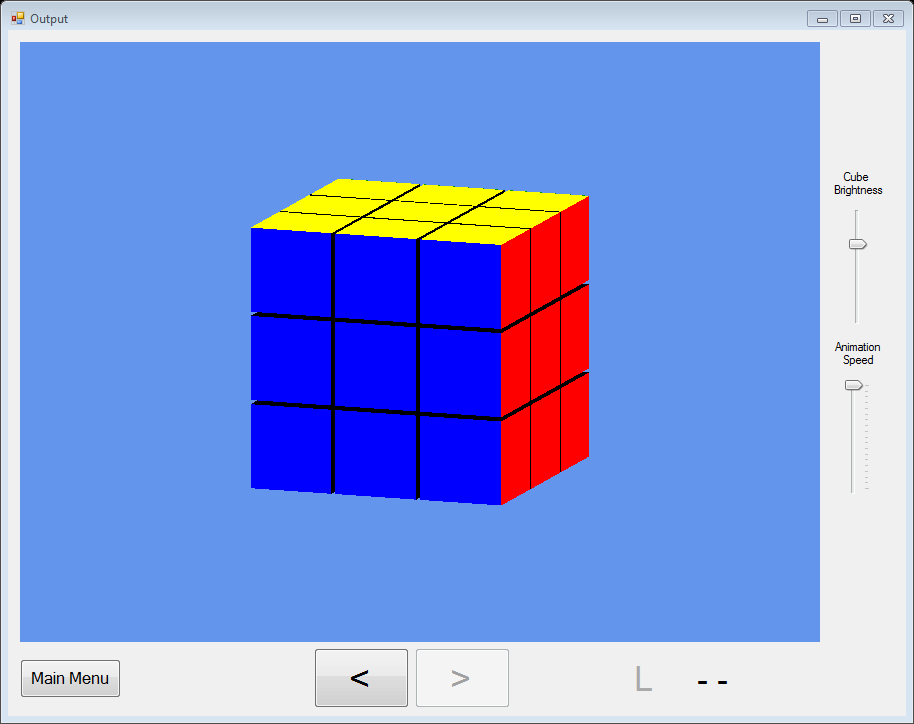
## Input Screens

## 

## Output Screen







# Analyse Class

## Analyses the inputted cube to see if it is a possible cube

''' <summary> Checks there are 9 of each colour in the array </summary>

''' <returns> true if there are 9 of each </returns>

Shared Function checkNumber(ByVal Stickers(,) As Char) As Boolean

If PublicFuncs.OtherIn1DIntArray(PublicFuncs.NumberInCharArray(Stickers), 9) Then

Console.WriteLine("Error: Incorrect Number")

MsgBox("You have not entered the correct number of each colour. Are you sure you entered all the squares correctly?")

Return False

Exit Function

Else

Return True

End If

End Function

'Checks adjacent middle edge cube != opposite colour or same colour - adjacent across 2 faces, not the same face

Shared Function checkEdges(ByVal Stickers(,) As Char) As Boolean

Dim oppcolour As Char

Dim adjacentFace, adjacentEdgeNumber As Integer

Dim wrongface, adjwrongface As String

For face = 0 To 5

For edgeNumber = 1 To 7 Step 2

adjacentFace = PublicFuncs.AdjacentEdge(face, edgeNumber)(0)

adjacentEdgeNumber = PublicFuncs.AdjacentEdge(face, edgeNumber)(1)

oppcolour = PublicFuncs.Opposite(Stickers(face, edgeNumber)).ToString()

'if the adjacent sticker is the same or opposite colour return false

If Stickers(adjacentFace, adjacentEdgeNumber) = oppcolour Or

Stickers(adjacentFace, adjacentEdgeNumber) = Stickers(face, edgeNumber) Then

wrongface = PublicFuncs.ColourChar2Word(Stickers(face, 4)).ToLower()

adjwrongface = PublicFuncs.ColourChar2Word(Stickers(adjacentFace, 4)).ToLower()

MsgBox("The middle cube on the " & wrongface & "/" & adjwrongface & " edge can't be that. Are you sure you entered it correctly?")

Return False

End If

Next

Next

Return True

End Function

'Check adjacent 2 corner stickers are not = opposite colour, checking all corner stickers checks each corner cube - adjacent across 2 faces, not the same face

Shared Function checkCorners(ByVal Stickers(,) As Char) As Boolean

Dim oppcolour As Char

Dim w, x, y, z As Integer

Dim wrongface, adjwrongface, adjwrongface2 As String

'Console.WriteLine("Check Corners:")

For a = 0 To 5

For b = 0 To 8 Step 2

If b <> 4 Then

w = PublicFuncs.AdjacentConers(a, b)(0)

x = PublicFuncs.AdjacentConers(a, b)(1)

y = PublicFuncs.AdjacentConers(a, b)(2)

z = PublicFuncs.AdjacentConers(a, b)(3)

oppcolour = PublicFuncs.Opposite(Stickers(a, b)).ToString()

'if either adjacent corner sticker is the same or opposite colour return false

If Stickers(w, x) = oppcolour Or Stickers(y, z) = oppcolour Or

Stickers(w, x) = Stickers(a, b) Or Stickers(y, z) = Stickers(a, b) Then

wrongface = PublicFuncs.ColourChar2Word(Stickers(a, 4)).ToLower()

adjwrongface = PublicFuncs.ColourChar2Word(Stickers(w, 4)).ToLower()

adjwrongface2 = PublicFuncs.ColourChar2Word(Stickers(y, 4)).ToLower()

MsgBox("The cube on the " & wrongface & "/" & adjwrongface & "/" & adjwrongface2 & " corner can't be that. Are you sure you entered it correctly?")

Return False

End If

End If

Next

Next

Return True

End Function

'Checks if corners are in right rotation (i.e. stickers on cube) in right order - (i.e. if 3rd sticker clockwise = what it should be based on the 1st and 2nd)

Shared Function checkCornerRotation(ByVal Stickers(,) As Char) As Boolean

Dim w, x, y, z As Integer

Dim supposedcolour As Char

Dim wrongface, adjwrongface, adjwrongface2 As String

For a = 0 To 5

For b = 0 To 8 Step 2

If b <> 4 Then

w = PublicFuncs.AdjacentConers(a, b)(0)

x = PublicFuncs.AdjacentConers(a, b)(1)

y = PublicFuncs.AdjacentConers(a, b)(2)

z = PublicFuncs.AdjacentConers(a, b)(3)

If Stickers(a, b) = "W" Or Stickers(a, b) = "Y" Then

If Stickers(a, b) = "W" Then

Select Case Stickers(w, x)

Case "R" : supposedcolour = "G"

Case "O" : supposedcolour = "B"

Case "G" : supposedcolour = "O"

Case "B" : supposedcolour = "R"

End Select

ElseIf Stickers(a, b) = "Y" Then

Select Case Stickers(w, x)

Case "R" : supposedcolour = "B"

Case "O" : supposedcolour = "G"

Case "G" : supposedcolour = "R"

Case "B" : supposedcolour = "O"

End Select

End If

If Stickers(y, z) <> supposedcolour Then

wrongface = PublicFuncs.ColourChar2Word(Stickers(a, 4)).ToLower()

adjwrongface = PublicFuncs.ColourChar2Word(Stickers(w, 4)).ToLower()

adjwrongface2 = PublicFuncs.ColourChar2Word(Stickers(y, 4)).ToLower()

MsgBox("The cube on the " & wrongface & "/" & adjwrongface & "/" & adjwrongface2 & " corner can't be that. Are you sure you entered it correctly?")

Return False

Exit Function

End If

End If

End If

Next

Next

Return True

End Function

# Processing Class – Solve Cube Subroutine

Private Sub SolveCube(ByRef corners() As Blocks.Corner, ByRef edges() As Blocks.Edge)

Dim Instructions As String = ""

Dim scrambledCorners() As Corner = copyCornerArray(corners)

Dim scrambledEdges() As Edge = copyEdgeArray(edges)

Dim top As Char = "W"

Dim front As Char = "R"

lblStatus.Text = "Solving Top Face"

lblStatus.Refresh()

If CheckForAnyCompleteFace(corners, edges, top, front, Instructions) Then

If CheckEdgesOfTopCross(corners, edges, top, front) Then

If Not CheckCornersOfTopFace(corners, edges, top, front) Then

DoTopCorners(corners, edges, top, front, Instructions)

End If

Else

'PermuteTopEdges(corners, edges, top, front, Instructions)

DoBottomEdges(corners, edges, top, front, Instructions)

DoTopFace(corners, edges, top, front, Instructions)

End If

Else

If CheckForAnyCross(corners, edges, top, front, Instructions) Then

If CheckEdgesOfTopCross(corners, edges, top, front) Then

If Not CheckCornersOfTopFace(corners, edges, top, front) Then

DoTopCorners(corners, edges, top, front, Instructions)

End If

Else

'PermuteTopEdges(corners, edges, top, front, Instructions)

DoBottomEdges(corners, edges, top, front, Instructions)

DoTopFace(corners, edges, top, front, Instructions)

End If

Else

DoTopFace(corners, edges, top, front, Instructions)

End If

End If

'top face done::

If Not CheckTopFaceRotation(corners, edges, top, front) Then

OrientateTopFace(corners, edges, top, front, Instructions)

End If

lblStatus.Text = "Solving Middle Row"

lblStatus.Refresh()

If Not CheckMiddleRow(corners, edges, top, front) Then

DoMiddleRow(corners, edges, top, front, Instructions)

End If

lblStatus.Text = "Solving Bottom Cross"

lblStatus.Refresh()

If Not CheckBottomCross(corners, edges, top, front, Instructions) Then

DoBottomCross(corners, edges, top, front, Instructions)

End If

If Not CheckEdgesOfTopCross(corners, edges, top, front) Then

DoBottomEdges(corners, edges, top, front, Instructions)

End If

lblStatus.Text = "Solving Bottom Face"

lblStatus.Refresh()

If Not CheckCornersOfTopFace(corners, edges, top, front) Then

PositionBottomCorners(corners, edges, top, front, Instructions)

End If

If Not CheckTopFace(corners, edges, top, front) Then

OrientateBottomCorners(corners, edges, top, front, Instructions)

End If

Console.WriteLine("Should now be complete")

If CheckComplete(corners, edges, top, front) Then

lblStatus.Text = "Solved"

lblStatus.Refresh()

Dim InstructionsList() As String

InstructionsList = OptimiseInstructions(Instructions)

OutputInstructions(corners, edges, Instructions)

Dim outputForm As New \_3DOutput(scrambledCorners, scrambledEdges, InstructionsList)

outputForm.Show()

Me.Close()

Else

Console.WriteLine("CUBE NOT SOVLED :(")

lblStatus.Text = "Not solved"

lblStatus.Refresh()

End If

End Sub