Appendix 1. Andrzej Jarosz, Tomasz Pecko, Przemysław Śleszyński

Algorithm (operating procedure) for view range calculation and its analysis

```
Option Explicit

Type TPoint
    """INPUT DATA
    x   'X coordinate carthesian (meters)
    Y   'Y coordinate carthesian (meters)
    z   'altitude (meters)
    h   'height of vegetation
    a   'attractiveness factor
    """OUTPUT DATA
    vp  'visible points
    f   'visible points on the area limits
    oa  'output aggregated attractiveness value
End Type

Const ObsH = 1.7 'height of the observer m

Dim Points() As TPoint  'two dimensional array of points
    'must be sorted by X ascending, Y ascending
    'to be filled with the input data by an external process

''''''
'main procedure iterating calculations
Sub VisibilityCalc()
    Dim ixL, ixU, iyL, iyU 'lower and upper bounds of the array
    Dim ix, iy, jx, jy
    ixL = LBound(Points, 1)
    ixU = UBound(Points, 1)
    iyL = LBound(Points, 2)
    iyU = UBound(Points, 2)
    'visibility iteration over each point of view
    For ix = ixL To ixU
        For iy = iyL To iyU
            With Points(ix, iy)
                .vp = 0
                .f = 0
                .oa = 0
            End With
            'for points with potential visibility
            If Points(ix, iy).h < ObsH Then
            Else
                'visibility check over each point
                For jx = ixL To ixU
                    For jy = iyL To iyU
                        If ix = jx And iy = jy Then
                            'self visible
                            With Points(ix, iy)
                        .vp = .vp + 1
                        .oa = .oa + .a
                        If jx = ixL Or jx = ixU Or _
                            jy = iyL Or jy = iyU Then
                           'point on the limits
                           .f = .f + 1
                        End If
                     End With
                  Else
                     If VisCheck(ix, iy, jx, jy) Then
                        'add visible point data
                        With Points(ix, iy)
                           .vp = .vp + 1
                           .oa = .oa + Points(jx, jy).a
                           If jx = ixL Or jx = ixU Or jy = iyL Or _
                              jy = iyU Then
                              .f = .f + 1
                           End If
                        End With
                     End If
                  End If
               Next jy
            Next jx
         End If
      Next iy
   Next ix
End Sub


Private Function VisCheck(xS, yS, xD, yD) As Boolean
'arguments: idx of source and dest points, false if invisible
   Dim Q
   Dim ix, iy, st
   If Abs(xS - xD) <= 1 And Abs(yS - yD) <= 1 Then 'direct sight
      VisCheck = True
      Exit Function
   End If

   'quadrants 0:upper,1:right,2:lower,3:left
   Q = 0
   If Abs(xS - xD) > Abs(yS - yD) Then 'left or right
      Q = 1
   End If
   If Q = 1 Then
      If xS > xD Then
         Q = 3
      End If
   Else
      If yS > yD Then
         Q = 2
      End If
```

```
    End If
    If Q = 2 Or Q = 3 Then
        st = -1
    Else
        st = 1
    End If
    VisCheck = True
    If Q = 0 Or Q = 2 Then
        For iy = yS To yD Step st
            ix = xS + (yS - iy) * (xS - xD) / (yS - yD)
            If Not VisCalc(xS, yS, xD, yD, ix, iy) Then
                VisCheck = False
                Exit Function
            End If
        Next iy
    Else
        For ix = xS To xD Step st
            iy = yS + (xS - ix) * (yS - yD) / (xS - xD)
            If Not VisCalc(xS, yS, xD, yD, ix, iy) Then
                VisCheck = False
                Exit Function
            End If
        Next ix
    End If

End Function

Private Function VisCalc(xS, yS, xD, yD, ix, iy) As Boolean
'false if visibility obscured
    Dim zS, zD, zP, zR
    Dim z1, z2
    If ix <> Int(ix) Then 'interpolation
        z1 = Point(Int(ix), iy).z + Point(Int(ix), iy).h
        z2 = Point(Int(ix) + 1, iy).z + Point(Int(ix) + 1, iy).h
        zP = z1 + (z2 - z1) * (ix - Int(ix))
    ElseIf iy <> Int(iy) Then
        z1 = Point(ix, Int(iy)).z + Point(ix, Int(iy)).h
        z2 = Point(ix, Int(iy) + 1).z + Point(ix, Int(iy) + 1).h
        zP = z1 + (z2 - z1) * (iy - Int(iy))
    Else
        zP = Point(ix, iy).z + Point(ix, iy).h
    End If
    zS = Point(xS, yS).z + ObsH 'point of sight elevation
    zD = Point(xD, yD).z + Point(xD, yD).h
    zR = zS + (zD - zS) * (xS - ix) * (yS - iy) / (xS - xD) / (yS - yD)
    If zR >= zP Then
        VisCalc = False
    Else
        VisCalc = True
    End If
End Function
```