Imports System.ComponentModel

Imports System.Diagnostics.Tracing

Imports System.Drawing

Imports System.Drawing.Text

Imports System.IO

Imports Microsoft.VisualBasic.Devices

Imports Windows.Win32.UI

Public Class Form2

Private Sub form2\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Form1.Visible = False

End Sub

'Game Running

Dim BoolIsRunning As Boolean = True

'Pacman as Pacman

Dim PacMan As PacMan

'Map Size

Dim IntMapWidth As Integer

Dim IntMaphight As Integer

'2D Array And Map display

Dim mapA(27, 30) As Integer

Dim IntResWidth As Integer

Dim IntResHeight As Integer

Dim IntTileSize As Integer = 22

Dim TilesNotEmpty As Integer = 0

'Graphics Variables

Dim GFX As Graphics

Dim BackBufferGFX As Graphics

Dim BackBuffer As Bitmap

Dim r As Rectangle

'Fps Counter & other

Dim InttSeconds As Integer = TimeOfDay.Second

Dim InttTicks As Integer = 0

Dim IntMaxTicks As Integer = 0

Dim Inttleft As Integer = 120

'Pathfinding

Private Sub LblClose\_Click(sender As Object, e As EventArgs) Handles LblClose.Click

BoolIsRunning = False

PnlExit.Visible = True

End Sub

Private Sub LblYes\_Click(sender As Object, e As EventArgs) Handles LblYes.Click

Form1.Visible = True

Close()

End Sub

Private Sub LblNo\_Click(sender As Object, e As EventArgs) Handles LblNo.Click

PnlExit.Visible = False

BoolIsRunning = True

End Sub

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Me.Show()

Me.Focus()

PnlExit.Visible = False

'Reminder the map size will be declared in another form when UI is completed for now its decrared here

IntMapWidth = MapDisplay1.Size.Width

IntMaphight = MapDisplay1.Size.Height

IntResHeight = IntMaphight

IntResWidth = IntMapWidth

'Declareing pacman

PacMan = New PacMan(13, 23)

PacMan.IX = 13

PacMan.IY = 23

'Hard coded Map1

mapA = {

{2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2},

{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 5, 5, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 4, 4, 4, 4, 4, 4, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{-1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 4, 4, 4, 4, 4, 4, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 4, 4, 4, 4, 4, 4, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2},

{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2},

{2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2},

{2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2},

{2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2, 2, 2},

{2, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2},

{2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2},

{2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2},

{2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2},

{2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2}

}

'Possably add more maps in the future

'Graphics objects

GFX = MapDisplay1.CreateGraphics

BackBuffer = New Bitmap(IntResWidth, IntResHeight)

Controle.Start()

Timer.Start()

End Sub

Private Sub drawgraphics()

'copying BackBuffer to graphics object

Using GFX As Graphics = Graphics.FromImage(BackBuffer)

'fix overdraw BackBuffer

GFX.Clear(Color.Wheat)

' Draw based on mapA array

For x = 0 To 27

For y = 0 To 30

r = New Rectangle(x \* IntTileSize, y \* IntTileSize, IntTileSize, IntTileSize)

' Find tiles and draws

Select Case mapA(y, x) 'I HATE THIS F###ING SH## MAP!!3 F###ING HOURES I SPENT TRYING TO SOLVE THIS AND F###ING INVERTING X AND Y SOLVED!!IT KILL ME NOW !!!!

Case 0 'Empty space

GFX.FillRectangle(Brushes.Black, r)

GFX.DrawRectangle(Pens.Black, r)

Case 1 'Pellet

GFX.FillRectangle(Brushes.Black, r)

Dim pelletRect As New Rectangle(r.X + IntTileSize \ 4, r.Y + IntTileSize \ 4, IntTileSize \ 2, IntTileSize \ 2)

GFX.FillEllipse(Brushes.Yellow, pelletRect)

GFX.DrawRectangle(Pens.Black, r)

Case 2 'Wall

GFX.FillRectangle(Brushes.Blue, r)

GFX.DrawRectangle(Pens.Black, r)

Case -1 'Pellet portal

GFX.FillRectangle(Brushes.Red, r)

Dim pelletRect As New Rectangle(r.X + IntTileSize \ 4, r.Y + IntTileSize \ 4, IntTileSize \ 2, IntTileSize \ 2)

GFX.FillEllipse(Brushes.Yellow, pelletRect)

GFX.DrawRectangle(Pens.Black, r)

Case -2 'Empty portal

GFX.FillRectangle(Brushes.Red, r)

GFX.DrawRectangle(Pens.Black, r)

Case 4 'Ghost spawn

GFX.FillRectangle(Brushes.Black, r)

GFX.DrawRectangle(Pens.Black, r)

Case 5 'Ghost gate

GFX.FillRectangle(Brushes.Gray, r)

GFX.DrawRectangle(Pens.Black, r)

End Select

Next

Next

' Draws Pac-Man

PacMan.Draw(GFX, IntTileSize)

End Using

'Draws BackBuffer to panel

Using BackBufferGFX As Graphics = MapDisplay1.CreateGraphics

BackBufferGFX.DrawImage(BackBuffer, 0, 0, IntResWidth, IntResHeight)

End Using

End Sub

Private Sub tickcounter() 'Finds Fps

If InttSeconds = TimeOfDay.Second And BoolIsRunning = True Then

InttTicks = InttTicks + 1

Else

IntMaxTicks = InttTicks

InttSeconds = TimeOfDay.Second

InttTicks = 0

End If

End Sub

Private Sub lblControl()

If BoolIsRunning = True Then

lblMaxtick.Text = "FPS: " & IntMaxTicks 'FPS

lbltTick.Text = "Ticks: " & InttTicks 'Tick speed

lblPacmanlocation.Text = "Pacman (x,y): (" & PacMan.IX & "," & PacMan.IY & ")" 'Pacman quords

Lblscore.Text = "Score: " & PacMan.Score 'Score

lbltimer.Text = "time: " & Inttleft 'Time left

If PacMan.Direction = 37 Then

lblDirection.Text = "Direction: Left"

ElseIf PacMan.Direction = 38 Then

lblDirection.Text = "Direction: Up"

ElseIf PacMan.Direction = 39 Then

lblDirection.Text = "Direction: Right"

ElseIf PacMan.Direction = 40 Then

lblDirection.Text = "Direction: Down"

Else

lblDirection.Text = "Direction: Unknown"

End If

If PacMan.NextDirection = 37 Then

lblNextDirection.Text = "NextDirection: Left"

ElseIf PacMan.NextDirection = 38 Then

lblNextDirection.Text = "NextDirection: Up"

ElseIf PacMan.NextDirection = 39 Then

lblNextDirection.Text = "NextDirection: Right"

ElseIf PacMan.NextDirection = 40 Then

lblNextDirection.Text = "NextDirection: Down"

Else

lblNextDirection.Text = "NextDirection: Unknown"

End If

End If

End Sub

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

If BoolIsRunning = True Then

'Keep app responsive

Application.DoEvents()

'Move Pac-Man

PacMan.Move(mapA)

'Add pathinding (future)

'Draw Graphics

drawgraphics()

'Add sound and effects (future)

'Measures Fps

tickcounter()

lblControl()

End If

End Sub

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

If BoolIsRunning = True Then

Inttleft = Inttleft - 1

Dim icountx As Integer = 0

Dim icounty As Integer = 0

End If

End Sub

Private Sub Form1\_KeyDown(sender As Object, e As KeyEventArgs) Handles Me.KeyDown 'Movement for pacman

Select Case e.KeyCode

Case Keys.Up 'key value = 38

PacMan.NextDirection = Keys.Up

If IsNodeEmpty(PacMan.IX, PacMan.IY - 1) Then

PacMan.Direction = Keys.Up

End If

Case Keys.Down 'key value = 40

PacMan.NextDirection = Keys.Down

If IsNodeEmpty(PacMan.IX, PacMan.IY + 1) Then

PacMan.Direction = Keys.Down

End If

Case Keys.Left 'key value = 37

PacMan.NextDirection = Keys.Left

If IsNodeEmpty(PacMan.IX - 1, PacMan.IY) Then

PacMan.Direction = Keys.Left

End If

Case Keys.Right 'key value = 39

PacMan.NextDirection = Keys.Right

If IsNodeEmpty(PacMan.IX + 1, PacMan.IY) Then

PacMan.Direction = Keys.Right

End If

End Select

End Sub

'Function to check if a node is empty

Private Function IsNodeEmpty(x As Integer, y As Integer) As Boolean

'Check node not wall

If x >= 0 AndAlso x < mapA.GetLength(1) AndAlso y >= 0 AndAlso y < mapA.GetLength(0) Then

If mapA(y, x) < 2 Then

Return True 'Node empty

End If

End If

Return False 'Node not empty

End Function

End Class

'Object Oreinted

Public Class Entity

Public Property IX As Integer

Public Property IY As Integer

Public verticalportaltravel As Boolean = False

Public horisontalportaltravel As Boolean = False

Private Mcount As Integer = 0

Private Mspeed As Integer = 9

Public Property Direction As Keys 'Stores pacman direction

Public Property NextDirection As Keys

Public Property x As Integer

Set(value As Integer)

IX = value

End Set

Get

Return IX

End Get

End Property

Public Property y As Integer

Set(value As Integer)

IY = value

End Set

Get

Return IY

End Get

End Property

Public Sub New(startX As Integer, startY As Integer)

IX = startX

IY = startY

End Sub

'Collision detector

Public Overridable Sub Move(mapA(,) As Integer)

Mcount += 1

If Mspeed <= Mcount Then

Dim newX = IX

Dim newY = IY

Dim nextX = IX

Dim nextY = IY

Dim NextXYIsValid As Boolean = False

Select Case NextDirection

Case Keys.Up '38

nextY = nextY - 1

Case Keys.Down '40

nextY = nextY + 1

Case Keys.Left '37

nextX = nextX - 1

Case Keys.Right '39

nextX = nextX + 1

End Select

If mapA(nextY, nextX) < 2 Then

NextXYIsValid = True

verticalportaltravel = False

horisontalportaltravel = False

End If

If NextXYIsValid = True Then

newX = nextX

newY = nextY

Direction = NextDirection

Else

'Direction property determines movement

Select Case Direction

Case Keys.Up '38

newY = newY - 1

Case Keys.Down '40

newY = newY + 1

Case Keys.Left '37

newX = newX - 1

Case Keys.Right '39

newX = newX + 1

End Select

End If

'Collision

'Checks walls

If mapA(newY, newX) < 2 Then

IX = newX

IY = newY

End If

'Checks portal

If IX = 0 Then

IX = 26

verticalportaltravel = True

End If

If IX = 27 Then

IX = 1

verticalportaltravel = True

End If

If IY = 0 Then

IY = 29

horisontalportaltravel = True

End If

If IY = 30 Then

IY = 1

horisontalportaltravel = True

End If

Mcount = 0

End If

End Sub

'GXF for pacman

Public Overridable Sub Draw(g As Graphics, tileSize As Integer)

Dim entityRect As New Rectangle(x \* tileSize, y \* tileSize, tileSize, tileSize)

g.FillEllipse(Brushes.Gray, entityRect)

End Sub

End Class

Public Class PacMan

Inherits Entity

'Score

Public Property Score As Integer

Public TilesNotEmpty As Integer = 0

Public Sub New(startX As Integer, startY As Integer)

MyBase.New(startX, startY)

Score = 0

End Sub

'Eats pellets

Public Overrides Sub Move(mapA(,) As Integer)

MyBase.Move(mapA)

If verticalportaltravel = True Then

If mapA(IY, IX - 1) = -1 Or mapA(IY, IX + 1) = -1 Then

verticalportaltravel = False

mapA(IY, 0) = -2

mapA(IY, 27) = -2

Score = Score + 20

TilesNotEmpty = 0

For icountx = 0 To 27 Step +1

For icounty = 0 To 30 Step +1

If mapA(icounty, icountx) = 1 Or mapA(icounty, icountx) = -1 Then

TilesNotEmpty = TilesNotEmpty + 1

End If

Next

Next

End If

ElseIf horisontalportaltravel = True Then

If mapA(IY + 1, IX) = -1 Or mapA(IY - 1, IX) = -1 Then

horisontalportaltravel = False

mapA(30, IX) = -2

mapA(0, IX) = -2

Score = Score + 20

TilesNotEmpty = 0

For icountx = 0 To 27 Step +1

For icounty = 0 To 30 Step +1

If mapA(icounty, icountx) = 1 Or mapA(icounty, icountx) = -1 Then

TilesNotEmpty = TilesNotEmpty + 1

End If

Next

Next

End If

ElseIf mapA(IY, IX) = 1 Then

mapA(IY, IX) = 0

Score = Score + 10

TilesNotEmpty = 0

For icountx = 0 To 27 Step +1

For icounty = 0 To 30 Step +1

If mapA(icounty, icountx) = 1 Or mapA(icounty, icountx) = -1 Then

TilesNotEmpty = TilesNotEmpty + 1

End If

Next

Next

End If

If TilesNotEmpty = 0 Then

IX = 13

IY = 23

Direction = 0

NextDirection = 0

verticalportaltravel = False

For icountx = 0 To 27 Step +1

For icounty = 0 To 30 Step +1

If mapA(icounty, icountx) = 0 Then

mapA(icounty, icountx) = 1

End If

If mapA(icounty, icountx) = -2 Then

mapA(icounty, icountx) = -1

End If

Next

Next

End If

End Sub

'Render pacman

Public Overrides Sub Draw(g As Graphics, tileSize As Integer)

Dim PacManRect As New Rectangle(IX \* tileSize, IY \* tileSize, tileSize, tileSize)

g.FillEllipse(Brushes.Yellow, PacManRect)

End Sub

End Class

Public Class Form1

Private Sub BtnStart\_Click(sender As Object, e As EventArgs) Handles BtnStart.Click

PnlOptions.Visible = False

PnlStats.Visible = False

Form2.Visible = True

End Sub

Private Sub BtnClose\_Click(sender As Object, e As EventArgs) Handles BtnClose.Click

Close()

End Sub

Private Sub BtnXOptions\_Click(sender As Object, e As EventArgs) Handles BtnXOptions.Click

PnlOptions.Visible = False

End Sub

Private Sub BtnOptiosns\_Click(sender As Object, e As EventArgs) Handles BtnOptiosns.Click

PnlOptions.Visible = True

End Sub

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

PnlOptions.Visible = False

PnlStats.Visible = False

End Sub

Private Sub BtnStats\_Click(sender As Object, e As EventArgs) Handles BtnStats.Click

PnlStats.Visible = True

End Sub

Private Sub BtnXStats\_Click(sender As Object, e As EventArgs) Handles BtnXStats.Click

PnlStats.Visible = False

End Sub

End Class