

Forum:[Hack Forums](#)

Thread: [Maze Generator](#)

Barret - February 8, 2017, 3:00 pm

Very small and compact maze generator class that works best with asynch thread and a callback.
Let me state that i don't own nor thought of this, it is public domain. Credits goes to a Phd teacher in a university from Polen.

How to use

Class constructor

Code Maze.Generator(Size, Color, PenSize)- Size Is the final bitmap size it generates
- Color, Pen size Well that speaks for itself...

Maze.Generator.StartAsynch()

Code Maze.Generator(Size, Color, PenSize).StartAsynch(GridSize, Seed, Complexity, Delegate(Callback))- Gridsize Is how large the maze is going to be, i suggest you keep this lower then 256x256 (even my 8-core will bitch slap me for this size)

- Seed Is for the randomizer
- Complexity This determines how complex it is going to be, values between 0~64, small values generates an easy maze
- Delegate Where to call back when the generate is done

The generator source

Spoiler ([Click to View](#))

PHP Code

<?php

Public Class Generator

Private Property Size As Size

Private Property Data As Byte(,)

Private Property Base As Integer()

Private Property Seed As Integer

Private Property Stack As Stack

Private Property Complexity As Integer

Private Property ComplexityOffset As Integer

Private Property BitmapSize As Size

Private Property BitmapColor As Color

Private Property BitmapPen As Integer

Sub New(BitmapSize As Size, BitmapColor As Color, BitmapPen As Integer)

Me.BitmapSize = BitmapSize

Me.BitmapColor = BitmapColor

Me.BitmapPen = BitmapPen

End Sub

Public Sub StartAsynch(Size As Size, Seed As Integer, Complexity As Integer, Callback As Action(

Of Bitmap))

Me.Size = Size

Me.Seed = Seed

Me.Stack = New Stack

Me.Complexity = Complexity

Me.ComplexityOffset = &H20

Me.Base = New Integer(Me.Size.Width * Me.Size.Height - 1) {}

Me.Data = New Byte(Me.Size.Width - 1, Me.Size.Height - 1) {}

Me.Initialize(New State(Me.Randomizer.Next Mod Me.Size.Width, Me.Randomizer.

Next Mod Me.Size.Height, 0))

Callback.Invoke(Me.Generate(Me.BitmapSize.Width, Me.BitmapSize.Height, Me.BitmapColor, Me.BitmapPen))

End Sub

Private Function Generate(width As Integer, height As Integer, PenColor As Color, PenSize As Integer) As Bitmap

Dim x As Single = Convert.ToSingle(width / Me.Size.Width)

Dim y As Single = Convert.ToSingle(height / Me.Size.Height)

Using bm As New Bitmap(width + 1, height + 1)

Using g As Graphics = Graphics.FromImage(bm)

Using pen As New Pen(PenColor, PenSize)

g.Clear(Color.WhiteSmoke)

g.DrawRectangle(pen, 0, 0, bm.Width - 1, bm.Height - 1)

For i As Integer = 0 To Me.Size.Width - 1

For j As Integer = 0 To Me.Size.Height - 1

If (Me.Data(i, j) And Convert.ToByte(1)) = 0 Then

g.DrawLine(pen, New PointF(x * i, y * j), New PointF(x * (i + 1), y * j))

Elseif (Me.Data(i, j) And Convert.ToByte(2)) = 0 Then

g.DrawLine(pen, New PointF(x * i, y * j), New PointF(x * i, y * (j + 1)))

End If

Next

Next

Return CType(bm.Clone, Bitmap)

End Using

End Using

End Using

End Function

Private Sub Initialize(State As State)

Dim abort As Boolean = False, found As Boolean

Dim src As Integer, dst As Integer, heading As Integer = 0, last As Integer = 0

For i As Integer = 0 To Me.Size.Width - 1

For j As Integer = 0 To Me.Size.Height - 1

Me.Data(i, j) = 0

Me.Base(Me.Indexer(i, j)) = -1

Next

Next

While True

If State.Heading = 15 Then

While State.Heading = 15

If (Me.Stack.Count > 0) Then

State = CType(Me.Stack.Pop(), State)

Else

abort = True

```

        Exit While
    End If
End While
If abort Then Exit While
Else
    Do
        last = heading
        heading = CInt(Math.Pow(2, Me.Randomizer.Next Mod 4))
        If (Me.Complexity >= Me.ComplexityOffset) Then
            Me.Complexity = Me.ComplexityOffset - 1
        End If
        If (Me.Randomizer.Next Mod Me.ComplexityOffset) < Me.Complexity Then
            If (State.Heading And last) = 0 Then
                heading = last
            End If
        End If

        found = (State.Heading And heading) <> 0

        Loop While found AndAlso State.Heading <> 15

        State.Heading = State.Heading Or heading

        src = Me.Indexer(State.Position.X, State.Position.Y)

        If (heading = 1 AndAlso State.Position.X > 0) Then
            dst = Me.Indexer(State.Position.X - 1, State.Position.Y)
            If Me.BaseCell(src) <> Me.BaseCell(dst) Then
                Me.Merge(src, dst)
                Me.Data(State.Position.X, State.Position.Y) = Me.Data(State.Position.X, State.
Position.Y) Or Convert.ToByte(2)
                Me.Stack.Push(New State(State))
                State.Position = New Point(State.Position.X - 1, State.Position.Y)
                State.Heading = 0
            End If
        ElseIf (heading = 2 AndAlso State.Position.X < Me.Size.Width - 1) Then
            dst = Me.Indexer(State.Position.X + 1, State.Position.Y)
            If Me.BaseCell(src) <> Me.BaseCell(dst) Then
                Me.Merge(src, dst)
                Me.Data(State.Position.X + 1, State.Position.Y) = Me.Data(State.Position.X + 1,
State.Position.Y) Or Convert.ToByte(2)
                Me.Stack.Push(New State(State))
                State.Position = New Point(State.Position.X + 1, State.Position.Y)
                State.Heading = 0
            End If
        ElseIf (heading = 4 AndAlso State.Position.Y > 0) Then
            dst = Me.Indexer(State.Position.X, State.Position.Y - 1)
            If Me.BaseCell(src) <> Me.BaseCell(dst) Then
                Me.Merge(src, dst)
                Me.Data(State.Position.X, State.Position.Y) = Me.Data(State.Position.X, State.
Position.Y) Or Convert.ToByte(1)
                Me.Stack.Push(New State(State))
                State.Position = New Point(State.Position.X, State.Position.Y - 1)
            End If
        End If
    End Do

```

```

        State.Heading = 0
    End If
    ElseIf (heading = 8 AndAlso State.Position.Y < Me.Size.Height - 1) Then
        dst = Me.Indexer(State.Position.X, State.Position.Y + 1)
        If Me.BaseCell(src) <> Me.BaseCell(dst) Then
            Me.Merge(src, dst)
            Me.Data(State.Position.X, State.Position.Y + 1) = Me.Data(State.Position.X, State.
Position.Y + 1) Or Convert.ToByte(1)
            Me.Stack.Push(New State(State))
            State.Position = New Point(State.Position.X, State.Position.Y + 1)
            State.Heading = 0
        End If
    End If
End If
End While
End Sub
Private Function Indexer(x As Integer, y As Integer) As Integer
    Return Me.Size.Width * y + x
End Function
Private Function BaseCell(index As Integer) As Integer
    While Me.Base(index) >= 0
        index = Me.Base(index)
    End While
    Return index
End Function
Private Sub Merge(x As Integer, y As Integer)
    Me.Base(Me.BaseCell(y)) = Me.BaseCell(x)
End Sub
Private Function Randomizer() As Random
    Static rnd As New Random(Me.Seed)
    Return rnd
End Function
End Class

```

```

Public Class State
    Public Property Position As Point
    Public Property Heading As Integer
    Sub New(State As State)
        Me.Heading = State.Heading
        Me.Position = State.Position
    End Sub
    Sub New(Position As Point)
        Me.Position = Position
    End Sub
    Sub New(Position As Point, h As Integer)
        Me.Heading = h
        Me.Position = Position
    End Sub
    Sub New(x As Integer, y As Integer, h As Integer)
        Me.Heading = h
        Me.Position = New Point(x, y)
    End Sub
End Class

```

```
Public Class Position
    Public Property x As Integer
    Public Property y As Integer
    Sub New(x As Integer, y As Integer)
        Me.x = x
        Me.y = y
    End Sub
End Class
```

For those that wonder how the 128x128 grid would look like, here you go:

(This was done in 201.7996 msecs)

Zilog80 - February 8, 2017, 4:09 pm

Nice

Once you have a maze, you can then have the program solve it too... ?

<https://channel9.msdn.com/coding4fun/blog/Getting-lost-and-found-with-the-C-Maze-Generator-and-Solver>

Bonus track #1: a lot info about mazes: <http://www.astrolog.org/labyrnth/algrithm.htm>

Bonus track #2: 'Visual' Depth First Search algorithm: <http://www.migapro.com/depth-first-search/>

VmX5 - February 8, 2017, 4:23 pm

Woah nice job, I've thought of attempting to do this a few times before but never got around to it, good job.

?-? ??? ???? - February 8, 2017, 4:27 pm

That's actually pretty sick dude, nice work :)

Ninja - February 8, 2017, 9:54 pm

This is awesome. It pleases my eyes.

But is it me or is there a lot of repetition in the patterns? Like, I see a lot of lines hook up, but never hook down.

MeltyNet - February 9, 2017, 1:06 am

Yo that's pretty fucking cool man. I can now go through mazes all day to entertain myself for hours!

marvinn - February 9, 2017, 1:44 am

Now make the solver, I'm curious on how you would do that

Barret - February 9, 2017, 6:41 am

(Feb 9, 2017 02:44 AM)marvinn Wrote: [\[CITE\]](#) Now make the solver, I'm curious on how you would do that

There was a solver but that code required some serious optimizing if i was to port it, i had to tune this code as well, teacher was smart about the algorithm but did not use the right design for certain elements.

(Feb 8, 2017 10:54 PM)Ninja Wrote: [\[CITE\]](#) This is awesome. It pleases my eyes.

But is it me or is there a lot of repetition in the patterns? Like, I see a lot of lines hook up, but never hook down.

The idea behind it is that all cells start closed (squares) and the pathways are randomly walked and by doing so opening the squares and creating a new road, the approach from that teacher was using headings, going north, west etc etc