Inside F#

Brian's thoughts on F# and .NET

- Home
- About

- **Categories**
  - Uncategorized
- **Archives**
  - January 2011
  - December 2010
  - November 2010
  - October 2010
  - September 2010
  - August 2010
  - July 2010
  - June 2010
  - May 2010
  - April 2010
  - March 2010
  - February 2010
  - December 2009
  - November 2009
  - October 2009
  - May 2009
  - April 2009
  - March 2009
  - February 2009
  - November 2008
  - October 2008
  - September 2008
  - August 2008
  - July 2008
  - June 2008
  - May 2008
  - April 2008
  - March 2008
  - February 2008
  - December 2007
  - November 2007
- Search
- **Blogroll**
  - Documentation
  - Plugins
  - Suggest Ideas
  - Support Forum
  - Themes
  - WordPress Blog
  - WordPress Planet
- **Meta**
  - Register
  - Log in

- **Subscribe**
  - Entries (RSS)
  - Comments (RSS)

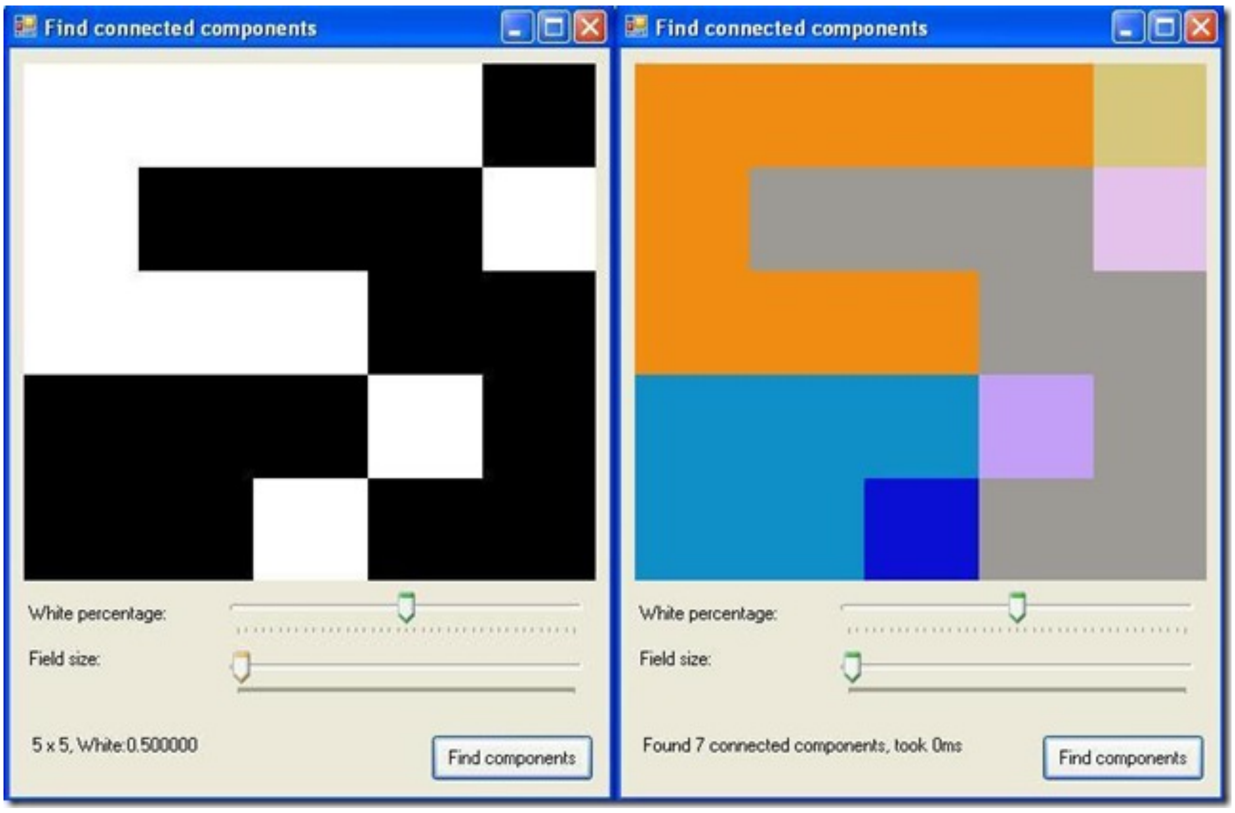
« "FSI is the new perl"
DebuggerVisualizers in F# »

Connected component labeling in F#

Posted by Brian on May 12, 2008

Kirill recently posted a blog about connected component labeling in C#. He also asked for solutions in other languages, so of course I had to code it in F#.

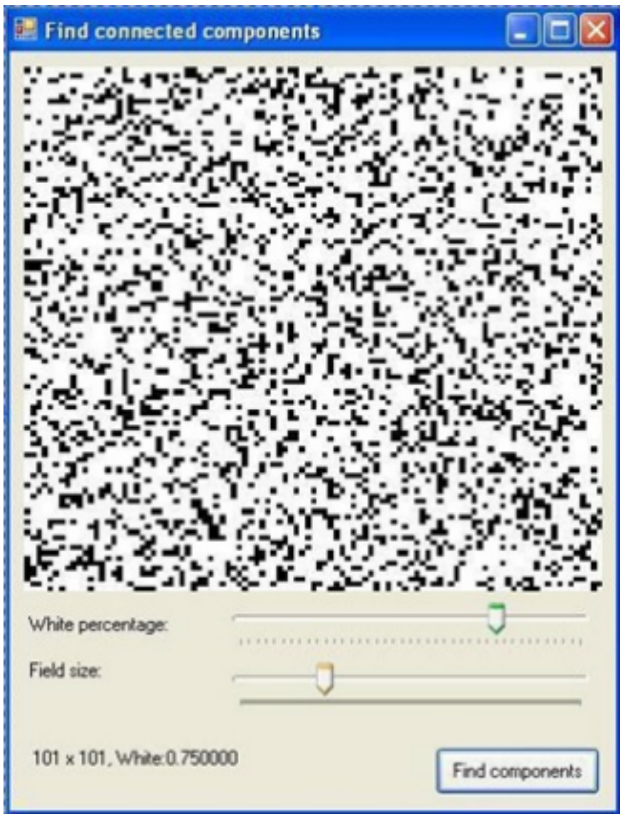
You can read his blog for more info, but the gist is, given a black and white grid, do a "flood fill" of each section with a different random color. Here's a sample before-and-after screenshot:



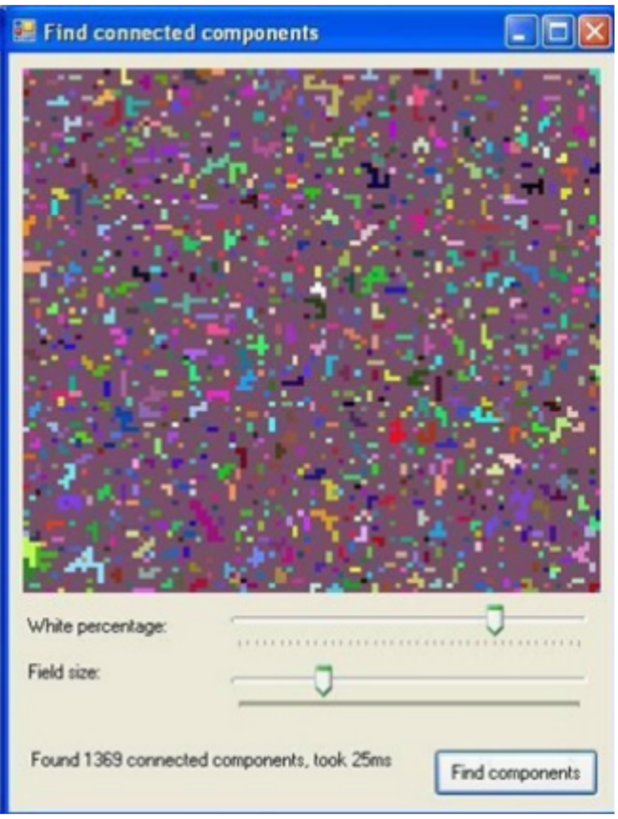
5 x 5, White:0.500000

Found 7 connected components, took 0ms

There are slider controls which let you control both the size of the initial black-and-white grid and the "white percentage". So here's a bigger example that starts off mostly white:



101 x 101, White:0.750000



Found 1369 connected components, took 25ms

When you move the sliders you get a new black-and-white grid, and then when you click the button, it colors it. Get the idea? Cool and fun.

Anyway, I shamelessly stole all Kirill's UI code, transliterated it to F#, and then implemented the Union-Find algorithm Kirill had mentioned, and it seems to be blazingly fast. So I present for your enjoyment, without further commentary, the F# code:

```
open System

// A partition is a mutable set of values, where one arbitrary value in the set
// is chosen as the canonical representative for that set.
type Partition<'a>(orig : 'a) as this =
    [<DefaultValue(false)>] val mutable parent : Partition<'a>
    [<DefaultValue(false)>] val mutable rank : int
    let rec FindHelper(x : Partition<'a>) =
        if Object.ReferenceEquals(x.parent, x) then
            x
        else
            x.parent <- FindHelper(x.parent)
            x.parent
    do this.parent <- this
    // The representative element in this partition
    member this.Find() =
        FindHelper(this)
    // Merges two partitions
    member this.Union(other : Partition<'a>) =
        let thisRoot = this.Find()
        let otherRoot = other.Find()
        if thisRoot.rank < otherRoot.rank then
            otherRoot.parent <- thisRoot
        elif thisRoot.rank > otherRoot.rank then
            thisRoot.parent <- otherRoot
        elif not (Object.ReferenceEquals(thisRoot, otherRoot)) then
            otherRoot.parent <- thisRoot
            thisRoot.rank <- thisRoot.rank + 1
    // The original value of this element
    member this.Value = orig

open System.Diagnostics
open System.Windows.Forms
open System.Drawing

let random = new Random()
type Info() =
    let mutable iMax = 1
    let mutable jMax = 1
    // The original grid (true = white)
    let mutable grid = Array2D.create iMax jMax true
    // Connected components
    let mutable colorField = Array2D.create iMax jMax (new Partition<_>(Color.White))
    // Initialize() resets the data and returns a white/black array
    member this.Initialize pctWhite size =
        iMax <- size
        jMax <- size
        grid <- Array2D.init iMax jMax (fun _ _ -> float (random.Next(100)) < 100.0 * pctWhite)
        colorField <- Array2D.init iMax jMax (fun _ _ ->
            new Partition<_>(Color.FromArgb(random.Next(256), random.Next(256), random.Next(256))))
        Array2D.init iMax jMax (fun i j -> if grid.[i,j] then Color.White else Color.Black)
    // Connect() connects components, and returns a tuple (numConnectedComponents, newColorArray)
    member this.Connect() =
        // connect components...
        for i in 0 .. iMax-1 do
            for j in 0 .. jMax-1 do
                if i <> 0 then
                    if grid.[i-1,j] = grid.[i,j] then
                        colorField.[i-1,j].Union(colorField.[i,j])
                if j <> 0 then
                    if grid.[i,j-1] = grid.[i,j] then
                        colorField.[i,j-1].Union(colorField.[i,j])
                if i <> iMax-1 then
                    if grid.[i+1,j] = grid.[i,j] then
                        colorField.[i+1,j].Union(colorField.[i,j])
                if j <> jMax-1 then
                    if grid.[i,j+1] = grid.[i,j] then
                        colorField.[i,j+1].Union(colorField.[i,j])
        // ... count how many there are, and pick a color for each component
        let h = new System.Collections.Generic.HashSet<_>()
        let theField = Array2D.init iMax jMax (fun i j ->
            let rep = colorField.[i,j].Find()
            h.Add(rep) |> ignore
            rep.Value  // color of representative element
        )
        (h.Count, theField)

// the UI
type Form1() as this =
    inherit Form()
    let Drawing = new PictureBox(Anchor = (AnchorStyles.Top ||| AnchorStyles.Bottom
                                           ||| AnchorStyles.Left ||| AnchorStyles.Right),
                                 Location = new System.Drawing.Point(12, 12),
                                 Name = "Drawing",
                                 Size = new System.Drawing.Size(485, 405),
                                 TabIndex = 0,
                                 TabStop = false)
    let FindComponents = new Button(Anchor = (AnchorStyles.Bottom ||| AnchorStyles.Right),
                                    Location = new System.Drawing.Point(358, 538),
                                    Name = "FindComponents",
                                    Size = new System.Drawing.Size(139, 37),
                                    TabIndex = 2,
                                    Text = "Find components",
                                    UseVisualStyleBackColor = true)
    let PercentageSlider = new TrackBar(Anchor = (AnchorStyles.Bottom ||| AnchorStyles.Left
                                                  ||| AnchorStyles.Right),
                                        LargeChange = 2,
                                        Location = new System.Drawing.Point(176, 423),
                                        Maximum = 40,
                                        Name = "PercentageSlider",
                                        Size = new System.Drawing.Size(321, 53),
                                        TabIndex = 3,
                                        Value = 20)
    let label1 = new Label(Anchor = (AnchorStyles.Bottom ||| AnchorStyles.Left),
                           AutoSize = true,
                           Location = new System.Drawing.Point(12, 434),
                           Name = "label1",
                           Size = new System.Drawing.Size(124, 17),
                           TabIndex = 4,
                           Text = "White percentage:")
    let label2 = new Label(Anchor = (AnchorStyles.Bottom ||| AnchorStyles.Left),
                           AutoSize = true,
                           Location = new System.Drawing.Point(12, 470),
                           Name = "label2",
                           Size = new System.Drawing.Size(71, 17),
                           TabIndex = 6,
                           Text = "Field size:")
    let FieldSizeSlider = new TrackBar(Anchor = (AnchorStyles.Bottom ||| AnchorStyles.Left
                                               ||| AnchorStyles.Right),
                                       LargeChange = 2,
                                       Location = new System.Drawing.Point(176, 470),
                                       Maximum = 100,
                                       Minimum = 1,
                                       Name = "FieldSizeSlider",
                                       Size = new System.Drawing.Size(321, 53),
                                       TabIndex = 5,
                                       Value = 5)
    let Status = new Label(Anchor = (AnchorStyles.Bottom ||| AnchorStyles.Left),
                           AutoSize = true,
                           Location = new System.Drawing.Point(15, 538),
                           Name = "Status",
                           Size = new System.Drawing.Size(0, 17),
                           TabIndex = 7)

    let mutable field = Array2D.create 1 1 Color.White  // the array we will Draw

    let Draw (canvas : Control) (graphics : Graphics) =
        let width  = float32 canvas.ClientSize.Width
        let height = float32 canvas.ClientSize.Height
        let iMax = Array2D.length1 field
        let jMax = Array2D.length2 field
        let iMaxFloat = float32 iMax
        let jMaxFloat = float32 jMax
        for i in 0 .. iMax-1 do
            for j in 0 .. jMax-1 do
                let w = width / iMaxFloat
                let h = height / jMaxFloat
                use brush = new SolidBrush(field.[i, j])
                graphics.FillRectangle(brush, w * float32 i, h * float32 j, w, h)

    let info = new Info()

    do this.InitializeComponent()

    member this.Repaint() = Drawing.Invalidate()

    member private this.Form1_Resize sender e =
        let maxFieldSize = Math.Max(5, Math.Min(Drawing.ClientSize.Width, Drawing.ClientSize.Height))
        FieldSizeSlider.Maximum <- maxFieldSize
        this.Repaint()

    member private this.FindComponents_Click sender e =
        let stopwatch = new Stopwatch()
        stopwatch.Start()
        let count, newField = info.Connect()
        field <- newField
        stopwatch.Stop()
        Status.Text <- sprintf "Found %d connected components, took %dms" count stopwatch.ElapsedMilliseconds
        this.Repaint()

    member this.Regenerate() =
        let size = FieldSizeSlider.Value
        let pct = float PercentageSlider.Value / float PercentageSlider.Maximum
        field <- info.Initialize pct size
        Status.Text <- sprintf "%d x %d, White:%f" size size pct
        this.Repaint()
    member private this.InitializeComponent() =
        (Drawing :> System.ComponentModel.ISupportInitialize).BeginInit()
        (PercentageSlider :> System.ComponentModel.ISupportInitialize).BeginInit()
        (FieldSizeSlider :> System.ComponentModel.ISupportInitialize).BeginInit()
        this.SuspendLayout()
        Drawing.Paint.AddHandler(fun s e -> Draw (Drawing :> Control) e.Graphics)
        FindComponents.Click.AddHandler(new EventHandler(this.FindComponents_Click))
        PercentageSlider.Scroll.AddHandler(fun s e -> this.Regenerate())
        FieldSizeSlider.Scroll.AddHandler(fun s e -> this.Regenerate())
        this.AcceptButton <- FindComponents
        this.AutoScaleDimensions <- new System.Drawing.SizeF(float32 8, float32 16)
        this.AutoScaleMode <- System.Windows.Forms.AutoScaleMode.Font
        this.ClientSize <- new System.Drawing.Size(509, 587)
        this.Controls.Add(Status)
        this.Controls.Add(label2)
        this.Controls.Add(FieldSizeSlider)
        this.Controls.Add(label1)
        this.Controls.Add(PercentageSlider)
        this.Controls.Add(FindComponents)
        this.Controls.Add(Drawing)
        this.DoubleBuffered <- true
        this.Name <- "Form1"
        this.StartPosition <- System.Windows.Forms.FormStartPosition.CenterScreen
        this.Text <- "Find connected components"
        this.Resize.AddHandler(new System.EventHandler(this.Form1_Resize))
        (Drawing :> System.ComponentModel.ISupportInitialize).EndInit()
        (PercentageSlider :> System.ComponentModel.ISupportInitialize).EndInit()
        (FieldSizeSlider :> System.ComponentModel.ISupportInitialize).EndInit()
        this.ResumeLayout(false)
        this.PerformLayout()
        this.Regenerate()

[<STAThread>]
do
    Application.EnableVisualStyles()
    Application.SetCompatibleTextRenderingDefault(false)
    Application.Run(new Form1())
```

Share this: Reddit

Like this: Like Be the first to like this post.

This entry was posted on May 12, 2008 at 3:30 pm and is filed under Uncategorized. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

Leave a Reply

Enter your comment here...

Guest Log In Log In Log In

Email (required) (Not published)

Name (required)

Notify me of follow-up comments via email.

Notify me of new posts via email.

Post Comment

« “FSI is the new perl”
DebuggerVisualizers in F# »