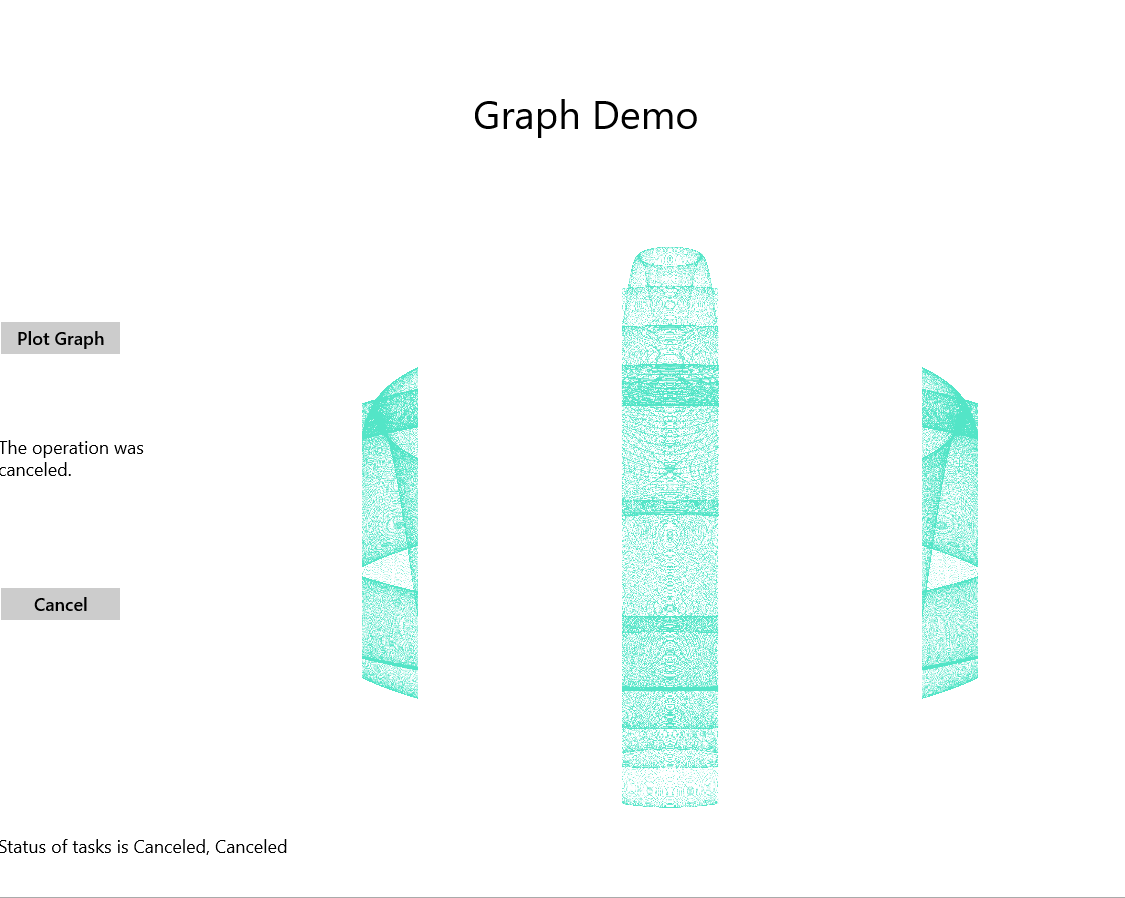
Student: Brian Johnston

Class COP 2362

Professor Hamilton

Chapter 23

Print Screens



Code:

using System;

using System.Diagnostics;

using System.IO;

using Windows.UI.ViewManagement;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Media.Imaging;

using Windows.UI.Xaml.Navigation;

using System.Runtime.InteropServices.WindowsRuntime;

using System.Threading.Tasks;

using System.Threading;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238

namespace GraphDemo

{

/// <summary>

/// An empty page that can be used on its own or navigated to within a Frame.

/// </summary>

public sealed partial class GraphWindow : Page

{

// Reduce pixelWidth and pixelHeight if there is insufficient memory available

private int pixelWidth = 12000;

private int pixelHeight = 8000;

private WriteableBitmap graphBitmap = null;

private int bytesPerPixel = 4;

private byte[] data;

private byte redValue, greenValue, blueValue;

private CancellationTokenSource tokenSource = null;

public GraphWindow()

{

this.InitializeComponent();

Window.Current.SizeChanged += App.WindowSizeChanged;

int dataSize = bytesPerPixel \* pixelWidth \* pixelHeight;

data = new byte[dataSize];

graphBitmap = new WriteableBitmap(pixelWidth, pixelHeight);

}

private async void plotButton\_Click(object sender, RoutedEventArgs e)

{

Random rand = new Random();

redValue = (byte)rand.Next(0xFF);

greenValue = (byte)rand.Next(0xFF);

blueValue = (byte)rand.Next(0xFF);

tokenSource = new CancellationTokenSource();

CancellationToken token = tokenSource.Token;

Stopwatch watch = Stopwatch.StartNew();

// Two tasks for dual-core computers

Task first = Task.Run(() => generateGraphData(data, 0, pixelWidth / 4, token), token);

Task second = Task.Run(() => generateGraphData(data, pixelWidth / 4, pixelWidth / 2, token), token);

//Task.WaitAll(first, second);

try

{

await first;

await second;

duration.Text = string.Format("Duration (ms): {0}", watch.ElapsedMilliseconds);

}

catch (OperationCanceledException oce)

{

duration.Text = oce.Message;

}

string message = string.Format("Status of tasks is {0}, {1}", first.Status, second.Status);

messages.Text = message;

Stream pixelStream = graphBitmap.PixelBuffer.AsStream();

pixelStream.Seek(0, SeekOrigin.Begin);

pixelStream.Write(data, 0, data.Length);

graphBitmap.Invalidate();

graphImage.Source = graphBitmap;

}

private void cancelButton\_Click(object sender, RoutedEventArgs e)

{

if (tokenSource != null)

{

tokenSource.Cancel();

}

}

private void generateGraphData(byte[] data, int partitionStart, int partitionEnd, CancellationToken token)

{

int a = pixelWidth / 2;

int b = a \* a;

int c = pixelHeight / 2;

for (int x = partitionStart; x < partitionEnd; x++)

{

int s = x \* x;

double p = Math.Sqrt(b - s);

for (double i = -p; i < p; i += 3)

{

//if (token.IsCancellationRequested)

//{

// return;

//}

token.ThrowIfCancellationRequested();

double r = Math.Sqrt(s + i \* i) / a;

double q = (r - 1) \* Math.Sin(24 \* r);

double y = i / 3 + (q \* c);

plotXY(data, (int)(-x + (pixelWidth / 2)), (int)(y + (pixelHeight / 2)));

plotXY(data, (int)(x + (pixelWidth / 2)), (int)(y + (pixelHeight / 2)));

}

}

}

private void plotXY(byte[] data, int x, int y)

{

int pixelIndex = (x + y \* pixelWidth) \* bytesPerPixel;

data[pixelIndex] = blueValue;

data[pixelIndex + 1] = greenValue;

data[pixelIndex + 2] = redValue;

data[pixelIndex + 3] = 0xBF;

}

}

}

Snapperview.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238

namespace GraphDemo

{

/// <summary>

/// An empty page that can be used on its own or navigated to within a Frame.

/// </summary>

public sealed partial class SnappedView : Page

{

public SnappedView()

{

this.InitializeComponent();

}

/// <summary>

/// Invoked when this page is about to be displayed in a Frame.

/// </summary>

/// <param name="e">Event data that describes how this page was reached. The Parameter

/// property is typically used to configure the page.</param>

protected override void OnNavigatedTo(NavigationEventArgs e)

{

}

}

}

App.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using Windows.ApplicationModel;

using Windows.ApplicationModel.Activation;

using Windows.Foundation;

using Windows.Foundation.Collections;

using Windows.UI.Core;

using Windows.UI.ViewManagement;

using Windows.UI.Xaml;

using Windows.UI.Xaml.Controls;

using Windows.UI.Xaml.Controls.Primitives;

using Windows.UI.Xaml.Data;

using Windows.UI.Xaml.Input;

using Windows.UI.Xaml.Media;

using Windows.UI.Xaml.Navigation;

// The Blank Application template is documented at http://go.microsoft.com/fwlink/?LinkId=234227

namespace GraphDemo

{

/// <summary>

/// Provides application-specific behavior to supplement the default Application class.

/// </summary>

sealed partial class App : Application

{

private static Frame MainWindowFrame = new Frame();

private static Frame SnappedViewFrame = new Frame();

/// <summary>

/// Initializes the singleton application object. This is the first line of authored code

/// executed, and as such is the logical equivalent of main() or WinMain().

/// </summary>

public App()

{

this.InitializeComponent();

this.Suspending += OnSuspending;

Application.Current.RequestedTheme = ApplicationTheme.Light;

}

public static void WindowSizeChanged(object sender, WindowSizeChangedEventArgs args)

{

ApplicationViewState viewState = ApplicationView.Value;

if (viewState == ApplicationViewState.Snapped)

{

Window.Current.Content = App.SnappedViewFrame;

}

else

{

if (Window.Current.Content != App.MainWindowFrame)

{

Window.Current.Content = App.MainWindowFrame;

}

}

Window.Current.Activate();

}

/// <summary>

/// Invoked when the application is launched normally by the end user. Other entry points

/// will be used when the application is launched to open a specific file, to display

/// search results, and so forth.

/// </summary>

/// <param name="args">Details about the launch request and process.</param>

protected override void OnLaunched(LaunchActivatedEventArgs args)

{

SnappedViewFrame.Navigate(typeof(SnappedView));

MainWindowFrame.Navigate(typeof(GraphWindow));

Frame rootFrame = Window.Current.Content as Frame;

// Do not repeat app initialization when the Window already has content,

// just ensure that the window is active

if (rootFrame == null)

{

// Create a Frame to act as the navigation context and navigate to the first page

rootFrame = MainWindowFrame;

if (args.PreviousExecutionState == ApplicationExecutionState.Terminated)

{

//**TODO: Load state from previously suspended application**

}

// Place the frame in the current Window

Window.Current.Content = rootFrame;

}

// Ensure the current window is active

Window.Current.Activate();

}

/// <summary>

/// Invoked when application execution is being suspended. Application state is saved

/// without knowing whether the application will be terminated or resumed with the contents

/// of memory still intact.

/// </summary>

/// <param name="sender">The source of the suspend request.</param>

/// <param name="e">Details about the suspend request.</param>

private void OnSuspending(object sender, SuspendingEventArgs e)

{

var deferral = e.SuspendingOperation.GetDeferral();

//**TODO: Save application state and stop any background activity**

deferral.Complete();

}

}

}