ComponentOne

TileView for UWP

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of GrapeCity, Inc. All other trademarks used herein are the properties of their respective owners.

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

Table of Contents

TileView for UWP 2
Help with UWP Edition 2
TileView for UWP Key Features 3
TileView for UWP Quick Start 4
Step 1 of 3: Creating the C1TileView Application 4
Step 2 of 3: Customizing the C1TileView Control 4-5
Step 3 of 3: Running the C1TileView Application 5-6
Working with TileView for UWP Edition 7
TileViewItem Elements 7
TileViewItem States 7-8
Columns and Rows 8
Minimized Item Position 8
Drag-and-Drop Interaction 8
TileView for UWP Task-Based Help 9
Adding C1TileView to the Application 9
Adding Items to C1TileView 9-10
Disabling Drag-and-Drop Functionality 10
Customizing the Header's Appearance 11
Creating Minimized and Maximized Item Templates 11-12

Copyright © 2019 GrapeCity, Inc. All rights reserved.

TileView for UWP

Interactively browse through your data with **TileView for UWP**. Expand and collapse tiles to view more or less information. Show-off the true touch-first and fast-and-fluid nature of Windows 10 in your apps with this highly visual and interactive control. Create dashboards, detail views, photo galleries and more!

Help with UWP Edition

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with ComponentOne Studio UWP Edition.

Copyright © 2019 GrapeCity, Inc. All rights reserved.

TileView for UWP Key Features

TileView for UWP includes the following key features:

- **Gesture-based Interaction and Animation**

  The C1TileView control is a very interactive control. Each tile can be viewed in three different states and users can toggle between states by simply tapping the tile header. Users can also rearrange tiles by simply sliding or flicking tiles in any direction. Smooth animations occur as tiles transition from one state to another.

- **Three Tile States**

  Each tile can be viewed in three states: Maximized, Minimized and Default. Show more or less information for each tile easily using any of the item templates. Default state is when all tiles are displayed at the same size.

- **Minimize Position**

  Tiles can be minimized to the top, left, bottom or right side of the C1TileView control by just setting one property. You can also specify the number of rows and columns when in the default state.

- **Flexible Data Binding**

  C1TileView is an items control which can be bound to any collection of business objects. Specify element bindings inside item templates as you would for any items control.

- **UI Virtualization**

  The C1TileView control supports UI virtualization so it can load and display hundreds of items without affecting performance.

Copyright © 2019 GrapeCity, Inc. All rights reserved.

TileView for UWP Quick Start

Interactively browse through your data with **TileView for UWP**. Expand and collapse tiles to view more or less information. Show-off the true touch-first and fast-and-fluid nature of Windows 10 in your apps with this highly visual and interactive control. Create dashboards, detail views, photo galleries and more!

Step 1 of 3: Creating the C1TileView Application

In this step, you create a XAML application using **TileView for UWP**. When you add a C1TileView control to your application, you have a interface that you can display content in. To set up your project and add a C1TileView control to your application, complete the following steps:

1. In Visual Studio select **File | New | Project**.
2. In the New Project dialog box, select Templates | Visual C# | Windows | Universal. From the templates list, select **Blank App (Universal Windows)**.

   The MainPage.xaml page will open with the cursor between the <Grid> and </Grid> tags.
3. Navigate to the Toolbox and drag the C1TileView icon to the page to add the control to the grid. This will add the reference and XAML namespace automatically. The XAML markup resembles the following:

   Markup

   ```
   <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
           <TileView:C1TileView HorizontalAlignment="Left" Height="100"
   VerticalAlignment="Top" Width="100"/>
       </Grid>
   ```
4. Inside the Grid, initialize the C1TileView control and give it a name by adding x:Name="C1TileView1" to the <TileView:C1TileView> tag so that it appears similar to the following:

   Markup

   ```
   <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
           <TileView:C1TileView x:Name="C1TileView1" HorizontalAlignment="Left"
   Height="100" VerticalAlignment="Top" Width="100"/>
       </Grid>
   ```

   This will add a C1TileView control named "C1TileView1" to the application.

You've successfully set up your application's user interface, but if you run your application now you see that the C1TileView control currently contains no content. In the next steps you'll add content to the C1TileView control, and then you'll observe some of the run-time interactions possible with the control.

Step 2 of 3: Customizing the C1TileView Control

In the previous step you created a XAML application and added the C1TileView control to your project. To customize your application, complete the following steps:

1. Add AllowDrop="True" within the C1TileView tags on the page to allow users to perform drag-and-drop operations with items in the control. The XAML markup will appear similar to the following:

Copyright © 2019 GrapeCity, Inc. All rights reserved.

Markup

```
<TileView:C1TileView x:Name="C1TileView1" AllowDrop="True"/>
```

2. Add three C1TileViewItems within the C1TileView tags so the XAML markup appears similar to the following:

Markup

```
<TileView:C1TileView x:Name="C1TileView1" AllowDrop="True">
    <TileView:C1TileViewItem></TileView:C1TileViewItem>
    <TileView:C1TileViewItem></TileView:C1TileViewItem>
    <TileView:C1TileViewItem></TileView:C1TileViewItem>
</TileView:C1TileView>
```

3. Add Background and Header properties to each of the C1TileViewItems, so the markup appears like the following:

Markup

```
<TileView:C1TileView x:Name="C1TileView1" AllowDrop="True">
            <TileView:C1TileViewItem Background="Red" Header="Red">
</TileView:C1TileViewItem>
            <TileView:C1TileViewItem Background="Blue" Header="Blue">
</TileView:C1TileViewItem>
            <TileView:C1TileViewItem Background="Yellow" Header="Yellow">
</TileView:C1TileViewItem>
        </TileView:C1TileView>
```

Each item will now appear in a different color and have text in the header.

In this step you added content to the C1TileView control. In the next step you'll view some of the run-time interactions possible in the control.

Step 3 of 3: Running the C1TileView Application

Now that you've created a XAML application and customized the C1TileView control, the only thing left to do is run your application. To run your application and observe **TileView for UWP**'s run-time behavior, complete the following steps:

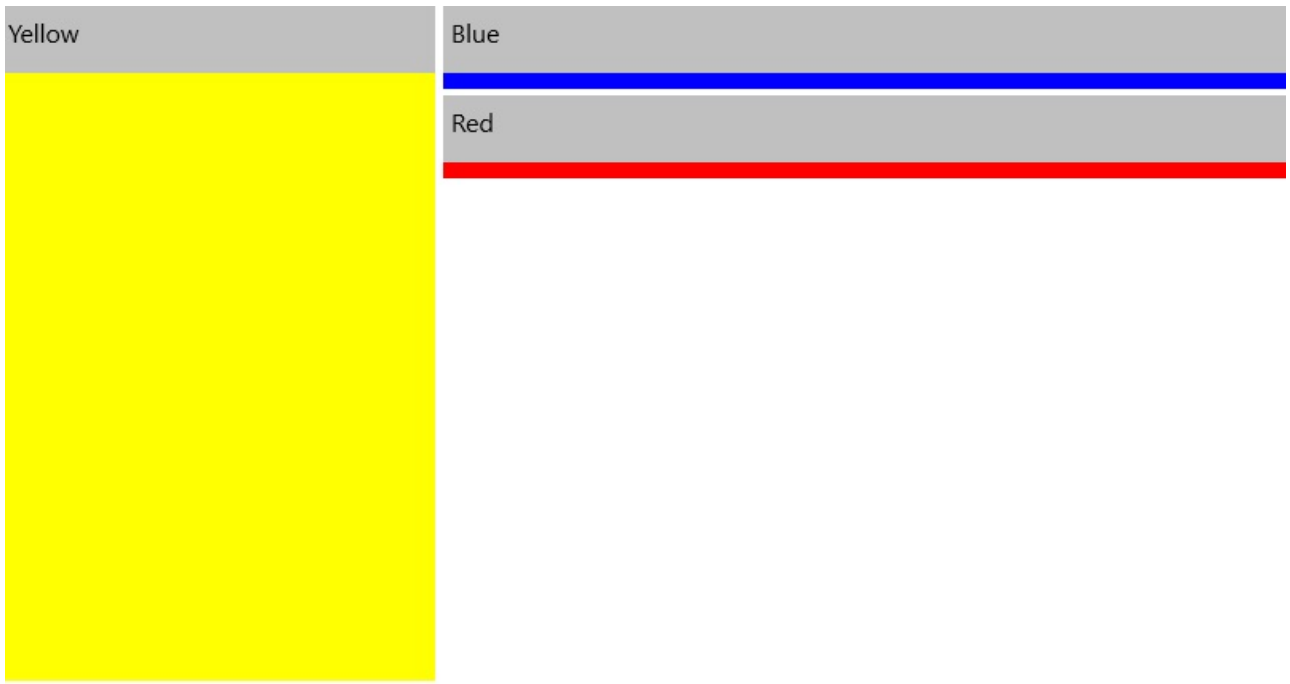
1. From the Debug menu, select Start Debugging to view how your application will appear at run time.

   The application will appear similar to the following:

Copyright © 2019 GrapeCity, Inc. All rights reserved.

Notice that the C1TileView control appears with three C1TileViewItems.

2. Click on the red item's header and drag the item to towards the blue item. The items will trade places.
3. Click on the yellow item's header to maximize the item. Note that the other two items are minimized:



Congratulations!
You've completed the **TileView for UWP** quick start and created a simple UWP application, added and customized a **TileView for UWP** control, and viewed some of the run-time capabilities of the control.

Copyright © 2019 GrapeCity, Inc. All rights reserved.

Working with TileView for UWP Edition

TileView for UWP includes the C1TileView control, a panel that slows you to interactively browse through your data. When you add the C1TileView control to a XAML window it exists as a blank container control that can be customized and include loaded content.

TileViewItem Elements

The C1TileViewItem control consists of two parts: a header and a content area. The image below identifies the toolbar and content area:

Any content that you add to the C1TileViewItem will be visible in the content area. You can add a caption bar title to the header area. The button in the upper right corner will either maximize or minimize the C1TileView control.

TileViewItem States

Each C1TileViewItem includes three different states – minimized, maximized, and the default state (which is neither minimized nor maximized). For example, in the following images all three of the C1TileViewItems in the C1TileView control appear in the default state:

In the following image, the red C1TileViewItem is maximized and the other two items are minimized:

Copyright © 2019 GrapeCity, Inc. All rights reserved.

When one item is maximized, the other items are minimized and appear as specified by the MinimizedItem property. The default state uses the Columns and Rows properties to determine the layout. The minimized/maximized states use the MinimizedItemsPosition property to determine layout.

Columns and Rows

The C1TileView.Columns and C1TileView.Rows properties get or set the number of columns and rows the C1TileViewItems are laid in respectively. If the value is zero, the minimum number that doesn't require scrolling is used. If both Columns and Rows are zero, the items are laid in a square.

The default state uses the Columns and Rows properties to determine the layout. The minimized/maximized states use the MinimizedItemsPosition property to determine layout. For more information about states, see TileViewItem States.

Minimized Item Position

The MinimizedItemsPosition property allows you to determine where minimized items will appear within the C1TileView control. Options include **Left**, **Right**, **Top**, and **Bottom**. By default, minimized items appear at the right of the panel.

Note that the default C1TileView state uses the Columns and Rows properties to determine the layout. The minimized/maximized states use the MinimizedItemsPosition property to determine layout. For more information about states, see TileViewItem States.

Drag-and-Drop Interaction

You can easily determine whether drag-and-drop operations are allowed within the C1TileView control by setting the CanUserReorder property. By default, this property is set to **True** and users can reorder items at run time. If this property is set to **False**, instead, users will no longer be able to reorder items at run time. See Disabling Drag-and-Drop Functionality for an example.

Copyright © 2019 GrapeCity, Inc. All rights reserved.

TileView for UWP Task-Based Help

The following task-based help topics assume that you are familiar with Visual Studio and Expression Blend and know how to use the C1TileView control in general. If you are unfamiliar with the **TileView for UWP** product, please see the TileView for UWP Quick Start first.

Each topic in this section provides a solution for specific tasks using the **TileView for UWP** product. Most task-based help topics also assume that you have created a new UWP project and added a C1TileView control to the project.

Adding C1TileView to the Application

In this topic you'll add a C1TileView control to your application. Complete the following steps:

1. In Visual Studio select **File | New | Project**.
2. In the **New Project** dialog box, select Templates | Visual C# | Windows | Universal. From the templates list, select **Blank App (Universal Windows)**. Enter a **Name** and click **OK** to create your project.
3. Open MainPage.xaml if it isn't already open, place the cursor between the <Grid> and </Grid> tags, and click once.
4. Navigate to the Toolbox and double-click the C1TileView icon to add the control to the grid. This will add the reference and XAML namespace automatically. The XAML markup resembles the following:

Markup

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <TileView:C1TileView HorizontalAlignment="Left" Height="100"
VerticalAlignment="Top" Width="100"/>
    </Grid>
```

What You've Accomplished

You've successfully set up your application's user interface, but if you run your application now you'll see that the C1TileView control currently contains no content. See the Adding Items to C1TileView topic for more information.

> If the C1TileView control was installed to the Visual Studio Toolbox, simply dragging the control onto a page will automatically perform all the steps above.

Adding Items to C1TileView

In this topic you'll add C1TileViewItems to the C1TileView control. Note that this topic assumes you have already added an empty C1TileView control to your application.

Edit the <TileView:C1TileView x:Name="C1TileView1" /> tag to add several C1TileViewItems. The XAML will appear similar to the following:

Markup

```
<TileView:C1TileView Name="C1TileView1">
    <TileView:C1TileViewItem Background="Red" Header="Red"></TileView:C1TileViewItem>
    <TileView:C1TileViewItem Background="Orange" Header="Orange">
</TileView:C1TileViewItem>
    <TileView:C1TileViewItem Background="Yellow" Header="Yellow">
</TileView:C1TileViewItem>
    <TileView:C1TileViewItem Background="Green" Header="Green">
```

Copyright © 2019 GrapeCity, Inc. All rights reserved.

```
</TileView:C1TileViewItem>
    <TileView:C1TileViewItem Background="Blue" Header="Blue">
</TileView:C1TileViewItem>
    <TileView:C1TileViewItem Background="Purple" Header="Purple">
</TileView:C1TileViewItem>
</TileView:C1TileView>
```

You've successfully added six C1TileViewItems to the C1TileView control.

Disabling Drag-and-Drop Functionality

By default drag-and-drop functionality is enabled allowing users to re-order C1TileViewItem elements at run time. If you choose, however, you can disable drag-and-drop functionality by setting the C1TileView.CanUserReorder property to False.

At Design Time

To disable drag-and-drop functionality in the C1TileView control in the Properties window at design time, complete the following steps:

1. Click the C1TileView control once to select it.
2. Navigate to the Properties window, and locate the CanUserReorder property.
3. Click the drop-down arrow next to the CanUserReorder property and select False.

This will disable drag-and-drop functionality.

In XAML

To disable drag-and-drop functionality in the C1TileView control in XAML add CanUserReorder="False" to the <TileView:TileView> tag so that it appears similar to the following:

Markup

```
<TileView:C1TileView Name="C1TileView1" CanUserReorder="False">
```

In Code

Right-click the window and select **View Code** to open the Code Editor. Add code to the main class, so it appears similar to the following:

Visual Basic

```
Public Sub New()
InitializeComponent()
    Me.C1TileView1.CanUserReorder = False
End Sub
```

C#

```
public MainPage()
{
InitializeComponent();
    this.C1TileView1.CanUserReorder = false;
}
```

Run your project and observe:
You will not be able to perform drag-and-drop operations at run time.

Copyright © 2019 GrapeCity, Inc. All rights reserved.

Customizing the Header's Appearance

C1TileView includes several properties that enable you to change the appearance of the **C1TileViewItem's Header**. These properties include: **Header**, HeaderBackground, **HeaderFontFamily**, **HeaderFontSize**, **HeaderFontStretch**, **HeaderFontStyle**, **HeaderFontWeight**, HeaderForeground, **HeaderPadding**, and **HeaderTemplate**.

For example, the following markup sets several of these properties:

Markup

```
<TileView:C1TileViewItem Header="News" HeaderPadding="10 5 5 5"
HeaderForeground="#FF507494" HeaderFontFamily="Trebuchet MS" HeaderFontSize="16">
   <TileView:C1TileViewItem.HeaderBackground>
      <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
        <GradientStop Color="#FFE9ECF0" Offset="0" />
            <GradientStop Color="#FFDDE1E7" Offset="0.2" />
            <GradientStop Color="#FFCCD3DC" Offset="0.2" />
            <GradientStop Color="#FFFAFAFB" Offset="0.647" />
      </LinearGradientBrush>
    </TileView:C1TileViewItem.HeaderBackground>
</TileView:C1TileViewItem>
```

Creating Minimized and Maximized Item Templates

You can customize how items in the C1TileView control are displayed when minimized and maximized. For example, you may wish to display an icon in the content area of a minimized item indicating the type of content that item contains. You can use the C1TileViewItem.ContentMinimized and C1TileViewItem.ContentMaximized properties to set a display template. If these properties are not set, the **Content** is used.

For example, the following markup adds **C1TileViewItem.ContentMinimized** and **C1TileViewItem.ContentMaximized** templates:

Markup

```
<TileView:C1TileView Name="C1TileView1">
            <TileView:C1TileViewItem Background="Red" Header="Red">
                <TileView:C1TileViewItem.ContentMinimized>
                    <TextBlock Text="Red Minimized" Height="28" Name="TextBlock1"
Foreground="White"/>
                </TileView:C1TileViewItem.ContentMinimized>
                <TileView:C1TileViewItem.ContentMaximized>
                    <TextBlock Text="Red Maximized" Height="28" Name="TextBlock2"
Foreground="White"/>
                </TileView:C1TileViewItem.ContentMaximized>
            </TileView:C1TileViewItem>
            <TileView:C1TileViewItem Background="Orange" Header="Orange">
                <TileView:C1TileViewItem.ContentMinimized >
                    <TextBlock Text="Orange Minimized" Height="28" Name="TextBlock3"
Foreground="White"/>
                </TileView:C1TileViewItem.ContentMinimized>
                <TileView:C1TileViewItem.ContentMaximized >
                    <TextBlock Text="Orange Maximized" Height="28" Name="TextBlock4"
```

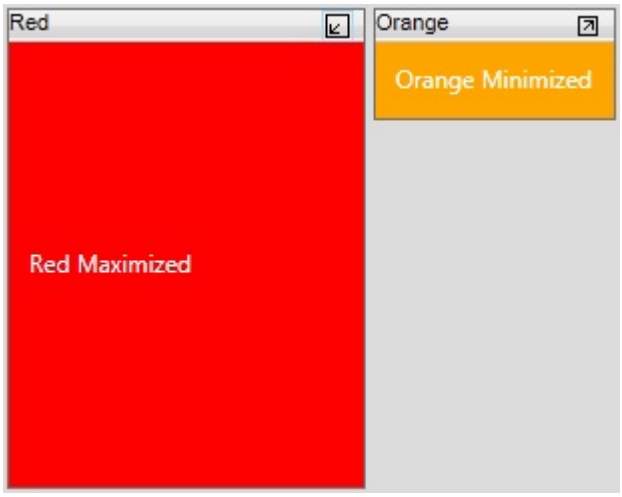
Copyright © 2019 GrapeCity, Inc. All rights reserved.

```
Foreground="White"/>
                </TileView:C1TileViewItem.ContentMaximized>
            </TileView:C1TileViewItem>

        </TileView:C1TileView>
```

What You've Accomplished

You've added templates for the minimized and maximized C1TileView states. Run your application and maximize one of the items, observe that the content of both the minimized and maximized items has changed. Maximize the minimized items, the content of each item changes again.



Copyright © 2019 GrapeCity, Inc. All rights reserved.