Contents

Chapter 1

Namespace CircularMenu

Namespace Contents *Page*

1.1 Interfaces

1.1.1 Interface **IExtraSpaceToolTipRenderer**

Defines a tool-tip renderer that requires more menu space than is allocated when accounting solely for icon sizes and positions. The rectangle returned by the method will be passed to the and methods when rendering.

Declaration

```
public interface IExtraSpaceToolTipRenderer
  :
```

Methods

- *GetMaximumRenderArea*

  `public` **System.Drawing.Rectangle GetMaximumRenderArea**`( )`

  Determines the space required for this renderer in order to display tool tips.

  - **Parameters**
    - `g` - A graphics context that can be used to measure font heights and widths.
    - `menuCenter` - A point within defaultMenuBoundaries that defines the central position of a circular menu.
    - `defaultMenuSize` - The boundaries allocated for the menu, accounting for icon sizes and widths.

Extended Information

- Assembly: `CircularMenu`

1.1.2 Interface **IFrameLayoutManager**

Defines a class that can position menu options within an animation frame.

Declaration

public interface IFrameLayoutManager
:

Methods

- *GetOptionPosition*

  `public` **System.Drawing.Point GetOptionPosition**`( )`

  Returns the position of the central pixel of the given option's bitmap for the given frame.

  - **Parameters**
    - `radius` - The preferred distance between the central location (0, 0) and the center of option images.
    - `optionIndex` - The index of the option whose position is to be determined.
    - `optionCount` - The total number of options to be rendered.
    - `frameIndex` - The index of the frame within which the option is to be psitioned.
    - `frameCount` - The total number of frames to be rendered.

Extended Information

- Assembly: `CircularMenu`

1.1.3 Interface **IFrameModifier**

Defines an option that can modify an existing frame image during an animation. These modifiers are applied to every frame in an animation with the exception of the final frame. For this reason, frame modifiers should attempt to render frames so that they approach a non-modified.

Declaration

```
public interface IFrameModifier
    :
```

Methods

- *ModifyFrame*

  `public` **CircularMenu.FrameOptionData ModifyFrame**`( )`

  Modifies the image from the provided menu option for the given animation frame. Note that this method will not be called for the final frame.

  - **Parameters**
    - `option` - The data for the menu option currently being modified.
    - `frameIndex` - The index of the frame currently being modified.
    - `frameCount` - The total number of frames to be rendered.

Extended Information

- Assembly: `CircularMenu`

1.1.4 Interface **IToolTipRenderer**

This interface describes an object that is capable of rendering menu tool tips.

Declaration

public interface IToolTipRenderer
:

Methods

- *Render*

  `public` **void Render**`( )`

  Instructs the renderer to draw the provided tool tip to the provided graphics context.

  – **Parameters**
    * `g` - The graphics context to use for rendering.
    * `toolTip` - The tool tip to render. Should not be null.
    * `menuCenter` - Defines the point where the menu's center pixel is located.
    * `renderArea` - Defines the rendering area that the tool tip should be placed within.

- *RenderEmpty*

  `public` **void RenderEmpty**`( )`

  Instructs the renderer to indicate that there is no tool tip currently available (either because the user is not hovering over an option or because the option the user is hovering over does not define a tool tip.

  – **Parameters**
    * `g` - The graphics context to use for rendering.
    * `menuCenter` - Defines the point where the menu's center pixel is located.
    * `renderArea` - Defines the rendering area that the tool tip should be placed within.

Extended Information

- Assembly: `CircularMenu`

1.2 Classes

1.2.1 Class **BelowMenuToolTipRenderer**

Provides a tool-tip renderer that is draws tool tips below the menu, instead of centrally (the default position).

Declaration

```
public class BelowMenuToolTipRenderer
  : StandardToolTipData
```

Constructors

- *.ctor*

  `public` **BelowMenuToolTipRenderer**`( )`

  Initializes a new instance of the BelowMenuToolTipRenderer class.

Methods

- *GetMaximumRenderArea*

  `public` **System.Drawing.Rectangle GetMaximumRenderArea**`( )`

  Determines the space required for the extra tool tips.

  – **Parameters**

  * `g` -
  * `menuCenter` -
  * `defaultMenuSize` -

- *Render*

  `public` **void Render**`( )`

  Renders the tool tip for the given menu option.

  – **Parameters**

  * `g` -
  * `toolTip` -
  * `menuCenter` -

* `renderArea` -

- *RenderEmpty*

  `public` **void RenderEmpty**`( )`

  If is true, behaves like when toolTip is the empty string (""). Otherwise, performs no action.

  – **Parameters**

  * `g` -
  * `menuCenter` -
  * `renderArea` -

Extended Information

- Assembly: `CircularMenu`

1.2.2 Class **BurnInFrameModifier**

A modified animation in which menu options "burn into" the screen. All white menu option masks are first faded in for the first 1/3 of the animation, and then the white masks are faded to the options themselves in the last two thirds. This is a slower modifier, so it should be used only if it is expected that the animation cache will stay consistent through many popups.

Declaration

```
public class BurnInFrameModifier
  : Object
```

Constructors

- *.ctor*

  `public` **BurnInFrameModifier**`( )`

  Initializes a new instance of the BurnInFrameModifier class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this modifer.

- *ModifyFrame*

  `public` **CircularMenu.FrameOptionData ModifyFrame**`( )`

  Applies the burn-in modification to the provided frame option.

  – **Parameters**

  * `option` -
  * `frameIndex` -
  * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.3 Class **CircularLayoutManager**

Defines a super type that can assist subtypes in creating circular arrangements.

Declaration

```
public class CircularLayoutManager
  : Object
```

Constructors

- *.ctor*

  `public` **CircularLayoutManager**`( )`

  Initializes a new instance of the CircularLayoutManager class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this layout manager.

- *GetOptionPosition*

  `public` **System.Drawing.Point GetOptionPosition**`( )`

  Returns the position of the provided option for the provided frame. Options are laid out in a clockwise, circular manner ending directly to the right of the clicked position.

  – **Parameters**
    * `radius` -
    * `optionIndex` -
    * `optionCount` -
    * `frameIndex` -
    * `frameCount` -

- *ModifyRadius*

  `protected` **float ModifyRadius**`( )`

  Simply returns the radius without modification.

  - **Parameters**
    * `radius` -
    * `optionIndex` -
    * `optionCount` -
    * `frameIndex` -
    * `frameCount` -

- *ModifyTheta*

  `protected` **float ModifyTheta**`( )`

  Simply returns the theta value without modification.

  - **Parameters**
    * `theta` -
    * `optionIndex` -
    * `optionCount` -
    * `frameIndex` -
    * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.4 Class **CircularMenuPopup**

Provides functionality for the display of a "circular" menu, an iconic menu that appears as a circle around the user's mouse position. The circular menu class provides six important properties. provides a that collects the actual items that will be displayed in the menu. and provide the animations for the menu, while defines the overall size of the menu. Finally, exposes a set of options for controlling how the menus tool tips are displayed. In addition to these, there are also two properties that allow you to override the default tool-tip behavior with the or with a custom tool-tip implementation. To override the default behavior, set the property to a non-null object that implements the interface. When is null (Nothing in Visual Basic), the default tool-tip behavior as defined by the property is used instead. You can obtain a reference to the actual tool-tip renderer used by checking the value of the property. CircularMenuPopup exposes a number of useful methods. The most important of these is undoubtedly the method. This method, which provides four overrides, is called when you wish the pop-up menu to be displayed and the user to select an option. You can provide either the exact screen coordinates of the menus central pixel, or you can provide the coordinates of that location relative to the edge of a control (such as your main form). You can control the cached menu animations via the and methods. The first of these builds and stores the menu animations, while the second clears them. Use to preload the animations when your application starts, instead of the first time the pop-up is shown. Use to reset the caches when you change an option that would change how the menu is rendered.

Declaration

```
public class CircularMenuPopup
  : Component
```

Properties

- *ActualRenderer*

  `public` **CircularMenu.IToolTipRenderer ActualRenderer** { `get;` }

  Defines the actual tool tip renderer to use.

- *ClosingAnimation*

  `public` **CircularMenu.MenuAnimation ClosingAnimation** { `get; set;` }

  The animation to play when closing this menu. Cannot be null.

- *IsPopupShown*

  `public` **bool IsPopupShown** { `get; }`
  Determines if the popup menu is currently shown.

- *MenuOptions*

  `public` **CircularMenu.MenuOptionCollection MenuOptions** { `get; set; }`
  The options that can be displayed on this menu. Cannot be null.

- *OpeningAnimation*

  `public` **CircularMenu.MenuAnimation OpeningAnimation** { `get; set; }`
  The animation to play when opening this menu. Cannot be null.

- *Radius*

  `public` **int Radius** { `get; set; }`
  The radius of the menu's layout.

- *ToolTip*

  `public` **CircularMenu.StandardToolTipRenderer ToolTip** { `get; set; }`
  Provides options related to the default rendering of tool tips within the menu.

- *ToolTipOverride*

  `public` **CircularMenu.IToolTipRenderer ToolTipOverride** { `get; set; }`
  A tool-tip renderer to use instead of the default one provided by the property. If null, the renderer will be used. Otherwise, this renderer will be used.

Constructors

- *.ctor*

  `public` **CircularMenuPopup**`( )`
  Initializes a new instance of the CircularMenuPopup class.

METHODS

- *CacheAnimations*

  `public` **void CacheAnimations**`( )`

  Builds the cached animations for both the opening and closing animations. Clears caches before rebuilding them.

- *ClearAnimationCaches*

  `public` **void ClearAnimationCaches**`( )`

  Clears the cached animation for both menu animations.

- *Popup*

  `public` **void Popup**`( )`

  Shows the menu popup centered at the given screen position. If the popup is currently shown (see), this method does nothing.

  – **Parameters**

    * `screenPos` - The screen position around which the popup menu should be centered.

- *Popup*

  `public` **void Popup**`( )`

  Shows the menu popup centered at the given screen position. If the popup is currently shown (see), this method does nothing.

  – **Parameters**

    * `screenX` -
    * `screenY` -

- *Popup*

  `public` **void Popup**`( )`

  Shows the menu popup centered at the given point on the given control. If a popup is currently shown (see), this method does nothing.

  – **Parameters**

    * `control` - The form on which the popup will be displayed. Cannot be null.
    * `centerPos` - The position, relative to the corner's position, where the popup should be displayed.

- *Popup*

  `public` **void Popup**`( )`

  Shows the menu popup centered at the given point on the given control.

  – **Parameters**
    * `control` - The form on which the popup will be displayed. Cannot be null.
    * `centerX` - The x-coordinate on control where the popup menu should be centered.
    * `centerY` - The y-coordinate on control where the popup menu should be centered.

Extended Information

- Assembly: `CircularMenu`

1.2.5 Class **CircularMenuWindow**

Summary description for CircularMenuWindow.

Declaration

```
public class CircularMenuWindow
  : Form
```

Properties

- *CreateParams*

  `protected` **System.Windows.Forms.CreateParams CreateParams** { `get;` }

  Sets some specific parameters we'll use to control rendering of our window.

- *SelectedOption*

  `public` **CircularMenu.MenuOption SelectedOption** { `get;` }

  This property can be used to test the selected menu option. It will be null if no option was selected (the menu was canceled).

Constructors

- *.ctor*

  `public` **CircularMenuWindow**`( )`

  Initializes a new circular menu window for the provided menu.

  – **Parameters**

  * `menu` - The menu to animate. Cannot be null.
  * `position` - The screen coordinates of central pixel of the animation. The window will position itself around this point while ensuring that the entire menu can still be shown on the screen.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    - `disposing` -

- *GetScreen*

  `public` **System.Windows.Forms.Screen GetScreen**`( )`

  Gets and returns the screen containing the given pixel, or null if no such screen exists.

  - **Parameters**
    - `position` -

- *WndProc*

  `protected` **void WndProc**`( )`

  Monitors Windows system messages for the message the indicates a window was selected. If the selected window wasn't this window, we'll close ourselves.

  - **Parameters**
    - `m` -

Extended Information

- Assembly: `CircularMenu`

1.2.6 Class **DropShadowOptions**

Provides a centralized collection of options used to render image drop shadows. Also provides a method, , that returns a drop shadow for a given image based on the settings defined for the class. This class has been designed to take advantage of the .NET design-time features. The default type converter is the , and the UI type editor is the .

Declaration

```
public class DropShadowOptions
  : Object
```

Properties

- *BlurRadius*

  `public` **int BlurRadius** { `get;` `set;` }

  The number of pixels by which each shadow pixel should be blurred. Must be at least 0.

- *MaximumOpacity*

  `public` **int MaximumOpacity** { `get;` `set;` }

  Defines the maximum opacity of any pixel in the rendered shadow. Must be between 0 and 255.

- *OpacityStep*

  `public` **int OpacityStep** { `get;` `set;` }

  The amount by which the alpha channel of each pixel in the drop shadow is increased by for each blur. Smaller values for this property tend to produce more subtle shadows, while larger values will produce shaper shadows. Must be at least 1.

- *ShadowColor*

  `public` **System.Drawing.Color ShadowColor** { `get;` `set;` }

  The color to use when rendering the drop shadow.

  – **Usage**

    * Changing this value can cause any connected parent to re-render itself.

- *ShadowOffsetX*

  `public` **int ShadowOffsetX** { `get;  set;` }

  Controls the horizontal offset of the shadow from the original image.

- *ShadowOffsetY*

  `public` **int ShadowOffsetY** { `get;  set;` }

  Controls the vertical offset of the shadow from the original image.

Constructors

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Creates a new drop shadow settings object and sets the properties to their defaults.

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Initializes a drop shadow options set and assigns the provided shadow color.

  – **Parameters**

    * `shadowColor` -

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Initializes an option set and applies the provided offset.

  – **Parameters**

    * `xOffset` -
    * `yOffset` -

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Initializes an option set and applies the provided offset.

  - **Parameters**
    - `offset` -

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Initializes an option set and applies the provided blur radius.

  - **Parameters**
    - `blurRadius` - The blur radius for the new option set. Must be at least 0.

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Initializes a new option set.

  - **Parameters**
    - `blurRadius` -
    - `xOffset` -
    - `yOffset` -

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Initializes a new option set.

  - **Parameters**
    - `blurRadius` -
    - `offset` -

- *.ctor*

  `public` **DropShadowOptions**`( )`

  Creates a new, independent copy of the provided settings object.

  - **Parameters**
    - `copyFrom` - The object to copy settings from. If null, all settings will be initialized to their defaults.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Returns a new, independent copy of this settings object. Changes to the clone will not cause the parent to reanimate.

- *GetDropShadow*

  `public` **System.Drawing.Bitmap GetDropShadow**`( )`

  Returns a drop shadow for the provided image, rendered using the settings currently applied to this object.

  – **Parameters**

    ∗ `image` -

- *GetImageWithDropShadow*

  `public` **System.Drawing.Bitmap GetImageWithDropShadow**`( )`

  Returns the provided image composited with its drop shadow, based on the settings defined in this class.

  – **Parameters**

    ∗ `image` - The image you wish to composite. Cannot be null.

Extended Information

- Assembly: `CircularMenu`

1.2.7 Class **DropShadowOptionsEditor**

Provides a class that launches a graphical user interface for editing the values of a object.

Declaration

```
public class DropShadowOptionsEditor
  : UITypeEditor
```

Constructors

- *.ctor*

  `public` **DropShadowOptionsEditor**`( )`

  Initializes a new instance of the DropShadowOptionsEditor class.

Methods

- *EditValue*

  `public` **object EditValue**`( )`

  Edits the provided value in a modal dialog.

  – **Parameters**

  * `context` -
  * `provider` -
  * `value` -

- *GetEditStyle*

  `public` **System.Drawing.Design.UITypeEditorEditStyle GetEditStyle**`( )`

  Indicates that this is a modal editor.

  – **Parameters**

  * `context` -

Extended Information

- Assembly: `CircularMenu`

1.2.8 Class **DropShadowOptionsEditorUi**

Defines a dialog that can edit objects.

Declaration

```
public class DropShadowOptionsEditorUi
  : Form
```

Properties

- *Options*

  `public` **CircularMenu.DropShadowOptions Options** { `get; set;` }

  Defines the set of options currently being edited by this dialog.

Constructors

- *.ctor*

  `public` **DropShadowOptionsEditorUi**`( )`

  Initializes a new editor for a default set of options.

- *.ctor*

  `public` **DropShadowOptionsEditorUi**`( )`

  Initializes a new editor and assigns it to edit the provided set of options, which cannot be null.

  – **Parameters**

    ∗ `options` -

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

  - **Parameters**
    * `disposing` -

- *DrawIconExample*

  `public` **void DrawIconExample**`( )`

  Repaints the iconic example (shows an actual image and its drop shadow). This method can take a long time.

- *DrawLayoutExample*

  `public` **void DrawLayoutExample**`( )`

  Repaints the layout example (showing only the sample regions). This method is fast, compared with .

- *RefreshDisplaySettings*

  `public` **void RefreshDisplaySettings**`( )`

  Synchronizes the GUI setting elements to reflect the current values of the options object, and repaints the examples.

Extended Information

- Assembly: `CircularMenu`

1.2.9 Class **FadeInFrameModifier**

Defines a frame modifier that slowly fades in menu options during the couse of the animation. Frame options approach a final opacity of 255 (fully opaque).

Declaration

```
public class FadeInFrameModifier
  : Object
```

Constructors

- *.ctor*

  `public` **FadeInFrameModifier**`( )`

  Initializes a new instance of the FadeInFrameModifier class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this modifier.

- *ModifyFrame*

  `public` **CircularMenu.FrameOptionData ModifyFrame**`( )`

  Modifies the provided option image, given the current position in the animation.

  – **Parameters**

    * `originalOption` -
    * `frameIndex` -
    * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.10 Class **FadeInZoomFrameModifier**

A combination of the and classes.

Declaration

```
public class FadeInZoomFrameModifier
  : Object
```

Constructors

- *.ctor*

  `public` **FadeInZoomFrameModifier**`( )`

  Initializes a new instance of the FadeInZoomFrameModifier class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this modifier.

- *ModifyFrame*

  `public` **CircularMenu.FrameOptionData ModifyFrame**`( )`

  Modifies the frame by fading it in and transitioning it from a one-pixel image to its normal size.

  – **Parameters**

  * `option` -
  * `frameIndex` -
  * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.11 Class **FinalFrame**

Defines the final frame of an animation. This frame is referenced when the system is rendering the menu for the user, and thus has some constraints that are not present in "normal" frames.

Declaration

```
public class FinalFrame
  : Frame
```

Constructors

- *.ctor*

  `public` **FinalFrame**`( )`

  Initializes the data for this frame.

  - **Parameters**
    - `options` - The options that are to be rendered into this frame. Cannot be null, nor can it contain less than one option.
    - `radius` - The radius to apply to the menu layout.
    - `frameIndex` - Ignored.
    - `frameCount` - The number of frames comprising the animation, including this frame. Must be at least one.
    - `layout` - The object to use for performing layout of the frame. Cannot be null.

Methods

- *HitTest*

  `public` **CircularMenu.MenuOption HitTest**`( )`

  Determines the menu option whose image contains the provided mouse coordinate when rendered, if any. If no such option exists, this method returns null.

  - **Parameters**
    - `xOffset` -
    - `yOffset` -
    - `mouseX` -

        * `mouseY` -

- *Render*

  `public` **void Render**`( )`

  Renders this frame, taking the current mouse position into account.

  – **Parameters**

    * `g` -
    * `xOffset` -
    * `yOffset` -
    * `mouseX` -
    * `mouseY` -
    * `mousePressed` -

Extended Information

- Assembly: `CircularMenu`

1.2.12 Class **FinalFrameOptionData**

Defines auxillary information for an option appearing in the final frame of an animation.

Declaration

```
public class FinalFrameOptionData
  : FrameOptionData
```

Properties

- *BoundingBox*

  `public` **System.Drawing.Rectangle BoundingBox** { `get;` }

  Returns the bounding box of this option (using the normal mode bitmap).

- *Option*

  `public` **CircularMenu.MenuOption Option** { `get;` }

  Gets the menu option rendered.

Constructors

- *.ctor*

  `public` **FinalFrameOptionData**`( )`

  Initializes a new final frame option data set.

  – **Parameters**
    * `centerPixelLocation` - The location of the bitmap's central pixel.
    * `option` - The menu option associated with this frame. Cannot be null.

Methods

- *RenderTo*

  `public` **void RenderTo**`( )`

  Renders this option to the provided graphics context.

– **Parameters**

* `g` - The graphics context to render to. Cannot be null.
* `xOffset` - An origin offset to apply to the x position.
* `yOffset` - An origin offset to apply to the y position.
* `optionState` - The state of the option.

Extended Information

- Assembly: `CircularMenu`

1.2.13 Class **ForwardFrameCollection**

Defines a frame collection that animates its elements from the initial state to the final state.

Declaration

```
public class ForwardFrameCollection
    : FrameCollection
```

Constructors

- *.ctor*

  `public` **ForwardFrameCollection**`( )`

  Initializes an animation for the provided collection.

  - **Parameters**
    - `options` - The options to animate. This parameter cannot be null, nor can the result of its method be empty.
    - `radius` - The radius to apply to the menu layout.
    - `frameCount` - The number of frames to render. Must be at least one.
    - `layout` - The object used to lay out the menu options during the animation.
    - `modifier` - An object that transforms image options during the animation.

Extended Information

- Assembly: `CircularMenu`

1.2.14 Class **ForwardMenuAnimation**

A menu animation that animates forward.

Declaration

```
public class ForwardMenuAnimation
  : MenuAnimation
```

Constructors

- *.ctor*

  `public` **ForwardMenuAnimation**`( )`

  Initializes a new instance of the ForwardMenuAnimation class.

Methods

- *GetClone*

  `protected` **CircularMenu.MenuAnimation GetClone**`( )`

  Returns a new forward menu animation.

- *GetUncachedAnimation*

  `public` **CircularMenu.FrameCollection GetUncachedAnimation**`( )`

  Returns a new set of frames for this animation, and does not cache the result.

  – **Parameters**

  * `menuOptions` - The menu options to animate. Cannot be null.
  * `radius` - The radius to apply to the menu layout.

Extended Information

- Assembly: `CircularMenu`

1.2.15 Class **Frame**

Defines an individual animation frame.

Declaration

public class Frame : Object

Fields

- *m_bounds*

  `familyorassembly` **System.Drawing.Rectangle m_bounds**

  The overall boundaries of this frame.

- *m_options*

  `familyorassembly` **System.Collections.ArrayList m_options**

  The options contained in this frame.

Properties

- *Bounds*

  `public` **System.Drawing.Rectangle Bounds** { `get;` }

  Gets the bounding box for this frame.

Constructors

- *.ctor*

  `public` **Frame**`( )`

  Initializes the data for the given frame.

  - **Parameters**
    - `options` - The options that are to be rendered into this frame. Cannot be null.

* `radius` - The radius to apply to the menu layout.
* `frameIndex` - The index of the frame to render. Must be at least zero and must be less than frameCount - 1.
* `frameCount` - The number of frames to render. Must be at least 1.
* `layout` - The object to use for performing layout of the frame. Cannot be null.
* `modifier` - The modifications to apply to the rendered options.

Methods

- *Render*

  `public` **void Render**`( )`

  Renders this frame to the provided graphics context.

  – **Parameters**

  * `g` -
  * `xOffset` -
  * `yOffset` -

Extended Information

- Assembly: `CircularMenu`

1.2.16 Class **FrameCollection**

A collection of frames to be rendered in an animation.

Declaration

public class FrameCollection : Object

Fields

- *m_finalFrame*

  `familyorassembly` **CircularMenu.FinalFrame m_finalFrame**
  The final frame in this colleciton.

- *m_frames*

  `familyorassembly` **System.Collections.ArrayList m_frames**
  The list of frames in this collection.

Properties

- *Count*

  `public` **int Count** { `get;` }
  Returns the number of frames defined in this animation.

- *FinalFrame*

  `public` **CircularMenu.FinalFrame FinalFrame** { `get;` }
  Returns the final frame of this animation.

- *Item*

  `public` **CircularMenu.Frame Item** { `get;` }
  Returns a reference to the frame at the provided position.

  – **Parameters**
    * `index` -

Constructors

- *.ctor*

  `familyorassembly` **FrameCollection**`( )`

  Protect access to the constructor.

Methods

- *GetEnumerator*

  `public` **System.Collections.IEnumerator GetEnumerator**`( )`

  Returns an object that can be used to enumerate the frames in this animation.

Extended Information

- Assembly: `CircularMenu`

1.2.17 Class **FrameOptionData**

Provides data that intermittently describes a menu option during the opening animation.

Declaration

```
public class FrameOptionData
  : Object
```

Properties

- *CenterPixelPosition*

  `public` **System.Drawing.Point CenterPixelPosition** { `get;` }

  Defines the location of the option's central pixel.

- *HasBitmap*

  `public` **bool HasBitmap** { `get;` }

  Determines if this option defines a bitmap to be painted.

- *OptionBitmap*

  `public` **System.Drawing.Bitmap OptionBitmap** { `get;` }

  Defines the bitmap to paint for the option. If null, no bitmap is to be painted.

Constructors

- *.ctor*

  `public` **FrameOptionData**`( )`

  Initializes the option data.

  – **Parameters**
    * `centerPosition` - The position of the central point of the option.
    * `option` - The bitmap representing the option. If null, the option will not be rendered.

Methods

- *RenderTo*

  `public` **void RenderTo**`( )`

  Renders this option to the provided graphics context.

  - **Parameters**
    - `g` - The graphics context to render to. Cannot be null.
    - `xOffset` - An origin offset to apply to the x position.
    - `yOffset` - An origin offset to apply to the y position.

- *RenderTo*

  `protected` **void RenderTo**`( )`

  Renders the provided image to the provided graphics context at the location specified by this class.

  - **Parameters**
    - `image` -
    - `g` -
    - `xOffset` -
    - `yOffset` -

Extended Information

- Assembly: `CircularMenu`

1.2.18 Class **MenuAnimation**

Provides options that control a menu animation. Both the and properties of the class accept instances of the MenuAnimation class. This class provides basic properties that control both the final and animated layouts of the menu, as well as special effects that are applied during the animation. The property of this class provides access to the object that controls the special effects in the animation. Set this property to an object that implements the interface to specify the special effect you desire. The built-in options are: : This special effect produces white-masks of the option images which it first fades in to full white. From there, the white images fade towards the normal images. : This special effect fades the option images in from fully transparent to their normal settings. : Combines the and effects. : This special type performs no action on the menu images. You must use this type to turn off special effects, since the property cannot be set to null. : This effect enlarges the images from 1x1 to their full size. Similarly, the property holds a reference to an object that implements the interface. Objects of this type are responsible for the layout of menu options during and after an animation. There are six built-in layout managers: : This class is the base class for the other options, and places menu options in a circular formation around the click position. However, this class does not animate the options in any way. : Animates options moving along the perimeter of the circle defined by . All options start at the same position and move with different speeds towards their final position. This provides an unfolding effect. : Similarly to the , this manager animates options moving along the perimeter of the circle. Unlike the other form, however, options start off in separate positions (180 degrees from their final location), and all move with the same speed towards their final location. : While this manager doesnt animate the degree positions of the options, it does animate them moving along their radii from the center of the circle out towards the edge. : Combines the effects of the spin layout manager and the starburst layout manager. : Combines the effects of the perimeter unfold layout manager and the starburst layout manager. You can control the length (and graininess) of the animation via the property. The final frame in the animation is adopted as the layout for the menu while the user is interacting with it. Because of this, this property has a minimum value of one. Frames are rendered once every thirty milliseconds, which produces a frame rate of about 30 frames per second. This means that an animation of 15 frames will take half a second to complete, a 30-frame animation takes about a second, and a 60-frame animation takes two seconds. Note that the animations may move slower when using more menu options and on slower computers. The complexity of the layout manager and frame modifier can also slow down an animation. Because of this, it is best to work with fewer frames. Finally, you can use the , , and methods to control the cached menu animation. The first of these always regenerates the animation and returns the result without replacing or setting the actual cached animation. The other effects work with the cached animation.

Declaration

public class MenuAnimation
: Object

Properties

- *FrameImageEffect*

  `public` **CircularMenu.IFrameModifier FrameImageEffect** { `get; set;` }

  Defines an object that is responsible for modifying the option images during the course of an animation. Cannot be null.

- *FramesToRender*

  `public` **int FramesToRender** { `get; set;` }

  The number of frames to render for the animation. Must be at least one.

  - **Usage**
    - Frames are rendered once every thirty milliseconds, which produces a frame rate of about 30 frames per second. This means that an animation of 15 frames will take half a second to complete, a 30-frame animation takes about a second, and a 60-frame animation takes two seconds. Note that the animations may move slower when using more menu options and on slower computers. The complexity of the layout manager and frame modifier can also slow down an animation. Because of this, it is best to work with fewer frames.

- *LayoutAnimator*

  `public` **CircularMenu.IFrameLayoutManager LayoutAnimator** { `get; set;` }

  Defines the object that is responsible for laying out menu options during and at the end of an animation. Cannot be null.

Constructors

- *.ctor*

`protected` **MenuAnimation**`( )`

Initializes a new instance of the MenuAnimation class.

Methods

- *ClearAnimation*

  `public` **void ClearAnimation**`( )`

  Ensures that the cached set of frames for this animation is cleared.

- *Clone*

  `public` **object Clone**`( )`

  Returns a new, independent copy of this animation. Note that when an animation is cloned, the clone will not have a cached animation.

- *GetAnimation*

  `public` **CircularMenu.FrameCollection GetAnimation**`( )`

  Returns a set of frames for this animation.

  – **Parameters**

  * `menuOptions` - The menu options to animate. Cannot be null.
  * `radius` - The radius to apply to the menu layout.

- *GetAnimation*

  `public` **CircularMenu.FrameCollection GetAnimation**`( )`

  Returns a set of frames for this animation.

  – **Parameters**

  * `menuOptions` - The menu options to animate. Cannot be null.
  * `radius` - The radius to apply to the menu layout.
  * `cacheResult` - If true, the result of this method will be saved for quick reference later. To clear a cached result, use the method.

- *GetClone*

  `protected` **CircularMenu.MenuAnimation GetClone**`( )`

  Returns a new instance of the implementing class.

- *GetUncachedAnimation*

  `public` **CircularMenu.FrameCollection GetUncachedAnimation`( )`**

  Returns a new set of frames for this animation, and does not cache the result.

  – **Parameters**

    * `menuOptions` - The menu options to animate. Cannot be null.
    * `radius` - The radius to apply to the menu layout.

Extended Information

- Assembly: `CircularMenu`

1.2.19 Class **MenuAnimationEditor**

A class that provides a modal graphical user interface for editing a instance.

Declaration

```
public class MenuAnimationEditor
  : UITypeEditor
```

Constructors

- *.ctor*

  `public` **MenuAnimationEditor**`( )`

  Initializes a new instance of the MenuAnimationEditor class.

Methods

- *EditValue*

  `public` **object EditValue**`( )`

  Edits the provided value.

  - **Parameters**
    - `context` -
    - `provider` -
    - `value` -

- *GetEditStyle*

  `public` **System.Drawing.Design.UITypeEditorEditStyle GetEditStyle**`( )`

  Indicates that this editor uses a modal dialog for editing.

  - **Parameters**
    - `context` -

Extended Information

- Assembly: `CircularMenu`

1.2.20 Class **MenuAnimationEditorUI**

Summary description for MenuAnimationEditorUI.

Declaration

```
public class MenuAnimationEditorUI
  : Form
```

Properties

- *Animation*

  `public` **CircularMenu.MenuAnimation Animation** { `get;  set;` }
  Defines the animation edited by this dialog.

Constructors

- *.ctor*

  `public` **MenuAnimationEditorUI**`( )`
  Creates a new editor dialog and edits the provided value.
  - **Parameters**
    - `edit` -

- *.ctor*

  `public` **MenuAnimationEditorUI**`( )`
  Creates a new dialog and edits a default animation.

Methods

- *Dispose*

  `protected` **void Dispose**`( )`
  Clean up any resources being used.
  - **Parameters**
    - `disposing` -

Extended Information

- Assembly: `CircularMenu`

1.2.21 Class **MenuEventArgs**

This class provides event arguments specific to menu-level events.

Declaration

```
public class MenuEventArgs
  : EventArgs
```

Properties

- *MenuPos*

  `public` **System.Drawing.Point MenuPos** { `get;` }

  The position of the shown menu, in screen coordinates.

- *SelectedOption*

  `public` **CircularMenu.MenuOption SelectedOption** { `get;` }

  Returns the option selected from the menu, or null if not applicable or if no option was selected.

Constructors

- *.ctor*

  `public` **MenuEventArgs**`( )`

  Creates a new instance of the menu event arguments class.

  – **Parameters**
    * `menuPos` - The position of the shown menu, in screen coordinates.
    * `selectedOption` - The option from the menu that was selected. Can be null if not applicable or if no option was selected.

Extended Information

- Assembly: `CircularMenu`

1.2.22 Class **MenuOption**

Defines an individual option in a circular menu. An options property is rendered whenever the user is not currently hovering over the option, and the option is enabled. Conversely, when the user is hovering over the option but not actually depressing it, the property is shown. The is shown when the user is depressing an enabled option, and the is shown regardless of the mouse state when an option is not enabled. You can view the actual images shown for each of the particular states by accessing the property for the normal state, the property for the hover state, and the property for the pressed state. Each of these properties either returns the property or their counterpart. It is via instances of the MenuOption class that a client application can actually track when options are clicked in a shown pop-up. This is accomplished via the , , and events. These events are raised when the user selects a menu option, hovers the mouse over the option, or moves the mouse away from the option, respectively. Each event provides an instance to the source menu option instance and as arguments.

Declaration

```
public class MenuOption
  : Component
```

Properties

- *CachedPrimaryHoverImage*

  `public` **System.Drawing.Bitmap CachedPrimaryHoverImage** { `get; }`

  Returns the bitmap to be drawn when the option is in its hover state. Returns the cached bitmap of either or .

- *CachedPrimaryImage*

  `public` **System.Drawing.Bitmap CachedPrimaryImage** { `get; }`

  Returns the bitmap to be drawn when the option is in its normal state (e.g, not hovered and not pressed). Returns the cached bitmap of either or .

- *CachedPrimaryPressedImage*

  `public` **System.Drawing.Bitmap CachedPrimaryPressedImage** { `get; }`

Returns the bitmap to be drawn when the option is in its pressed state. Returns the cached bitmap of either or .

- *DisabledImage*

  `public` **CircularMenu.MenuOptionImage DisabledImage** { `get; set;` }

  The image to render when the option is disabled.

- *Enabled*

  `public` **bool Enabled** { `get; set;` }

  Determines if this menu option is enabled.

- *HoverImage*

  `public` **CircularMenu.MenuOptionImage HoverImage** { `get; set;` }

  The image rendered when the user hovers the mouse over the image.

- *Image*

  `public` **CircularMenu.MenuOptionImage Image** { `get; set;` }

  The image rendered when the option is in its normal state.

- *PressedImage*

  `public` **CircularMenu.MenuOptionImage PressedImage** { `get; set;` }

  The image to render when the option is depressed.

- *ToolTip*

  `public` **string ToolTip** { `get; set;` }

  A small "tool tip" that is drawn whenever the user hovers over the menu. Cannot be null, but if "", no tool tip will be drawn.

- *Visible*

  `public` **bool Visible** { `get; set;` }

  Determines if this menu option is visible.

Constructors

- *.ctor*

  `public` **MenuOption**`( )`

  Initializes a new instance of the MenuOption class.

Extended Information

- Assembly: `CircularMenu`

1.2.23 Class **MenuOptionCollection**

Defines a collection of menu options.

Declaration

```
public class MenuOptionCollection
  : CollectionBase
```

Properties

- *Item*

  `public` **CircularMenu.MenuOption Item** { `get;  set;` }

  Returns the menu option at the provided index.

  – **Parameters**

    * `index` -

Constructors

- *.ctor*

  `public` **MenuOptionCollection**`( )`

  Initializes a new collection.

Methods

- *Add*

  `public` **CircularMenu.MenuOption Add**`( )`

  Creates a new menu option and adds it to the list. Returns the created option.

- *Add*

  `public` **int Add**`( )`

  Adds a menu option to the collection.

  - **Usage**
    * Null options are ignored.
  - **Parameters**
    * `option` -

- *AddRange*

  `public` **void AddRange**`( )`

  Adds a range of options to the collections. None of the options can be null.

  - **Parameters**
    * `c` -

- *Contains*

  `public` **bool Contains**`( )`

  Determines if this collection contains the provided option.

  - **Parameters**
    * `option` -

- *GetVisibleOptions*

  `public` **CircularMenu.MenuOptionCollection GetVisibleOptions**`( )`

  Returns a subset of this collection that contains only its visible elements.

- *IndexOf*

  `public` **int IndexOf**`( )`

  Determines the index of the provided option.

  - **Parameters**
    * `option` -

- *Remove*

  `public` **void Remove**`( )`

  Removes the provided option.

  - **Parameters**
    * `option` -

Extended Information

- Assembly: `CircularMenu`

1.2.24 Class **MenuOptionImage**

Provides options for one of the various images associated with a menu option, and supports rendering that image and caching the result. The MenuOptionImage class consolidates the options for each image into a single instance. These options include the image itself, its transparency key, overall transparency, and drop shadow. The base image, specified via the property, is an instance of the .NET frameworks class. This can be an image of any format supported by the framework, including Windows Bitmap, GIF, JPEG, and PNG, or it can be an image that your rendered manually via code. Use the property to specify a color from the that is not drawn when the pop-up menu is rendered. For example, if you wanted your option to be circular in shape, you could place the circle on a bright green background, and set to the background color. If you are using an image format that intrinsically supports transparency, such as PNG, you should set this property to . Similarly, the property controls image semi-transparency. This is different from in that it is applied to every pixel in the image, regardless of color. Furthermore, whereas the transparency key renders a pixel as either fully transparent (0) or fully opaque (255), this property allows you to make a pixel semi-transparent, so that its color will be blended with the background color. This value can be set to an integer between 0 and 255, inclusive. The smaller the number, the more of the background shows; the larger the number, the more of the image itself shows. 0 renders as fully transparent, and 255 renders as fully opaque. Finally, the and options control the visibility of the images drop shadow. The class allows you to configure the details of the drop shadow algorithm. The image is rendered by applying the options in the following order: –gt; –gt; and –gt; Because MenuOptionImage instances represent a number of operations that are applied to a basic image, these objects cache a pre-rendered version of the option that can be accessed rapidly whenever the option needs to be drawn. Menu option images automatically clear their image cache whenever any of their properties change, but you can take direct control over this via the and methods. You can obtain a reference to the cached, fully rendered image, via the property.

Declaration

```
public class MenuOptionImage
  : Object
```

Properties

- *CachedImage*

  `public` **System.Drawing.Bitmap CachedImage** { `get; }`

  Returns a bitmap that is the end product of rendering the property with all of the settings defined in this instance. This image will be created if

required, and then saved for later use. Changing any of the other properties in this instance will clear the cached bitmap.

- *DropShadow*

  `public` **CircularMenu.DropShadowOptions DropShadow** { `get; set;` }

  The drop shadow options for this image. Cannot be null.

- *Image*

  `public` **System.Drawing.Bitmap Image** { `get; set;` }

  The base image for this option image.

- *MaximumOpacity*

  `public` **byte MaximumOpacity** { `get; set;` }

  Determines the maximum value for the alpha-channel of any pixel in the cached bitmap.

- *TransparencyKey*

  `public` **System.Drawing.Color TransparencyKey** { `get; set;` }

  If set, any pixel in of this color will be made 100% transparent.

- *UseDropShadow*

  `public` **bool UseDropShadow** { `get; set;` }

  Determines if a drop shadow should be applied to the image.

Constructors

- *.ctor*

  `public` **MenuOptionImage( )**

  Initializes a new image with default settings.

- *.ctor*

  `public` **MenuOptionImage( )**

Initializes a new image by cloning data from the old image. If the provided image is null, this constructor behaves identically to the default constructor.

– **Parameters**

* `copyFrom` -

Methods

- *ClearCache*

  `public` **void ClearCache**`( )`

  Disposes and clears the cached bitmap, forcing it to be recreated the next time it is required.

- *Clone*

  `public` **object Clone**`( )`

  Returns a new, independent copy of this menu option image.

- *CreateCache*

  `public` **void CreateCache**`( )`

  Renders the cached bitmap if required.

Extended Information

- Assembly: `CircularMenu`

1.2.25 Class **MenuOptionImageEditor**

Provides a class that can display a UI for editing menu option images.

Declaration

```
public class MenuOptionImageEditor
  : UITypeEditor
```

Constructors

- *.ctor*

  `public` **MenuOptionImageEditor**`( )`

  Initializes a new instance of the MenuOptionImageEditor class.

Methods

- *EditValue*

  `public` **object EditValue**`( )`

  Displays a modal dialog to edit the provided value.

  – **Parameters**

  * `context` -
  * `provider` -
  * `value` -

- *GetEditStyle*

  `public` **System.Drawing.Design.UITypeEditorEditStyle GetEditStyle**`( )`

  Indicates that we edit with a modal dialog.

  – **Parameters**

  * `context` -

- *GetPaintValueSupported*

`public` **bool GetPaintValueSupported**`( )`

Indicates that we support painting a preview image of our edited type to a property listing.

– **Parameters**

* `context` -

- *PaintValue*

`public` **void PaintValue**`( )`

Paints the image for the provided value.

– **Parameters**

* `e` -

Extended Information

- Assembly: `CircularMenu`

1.2.26 Class **MenuOptionImageEditorUI**

Summary description for MenuOptionImageEditorUI.

Declaration

```
public class MenuOptionImageEditorUI
  : Form
```

Properties

- *OptionImage*

  `public` **CircularMenu.MenuOptionImage OptionImage** { `get; set;` }

  The image being edited. Setting this to null will be ignored.

Constructors

- *.ctor*

  `public` **MenuOptionImageEditorUI**`( )`

  Initializes a new image editor and edits a default menu option image.

- *.ctor*

  `public` **MenuOptionImageEditorUI**`( )`

  Initializes a new image editor and edits the provided menu option.

  – **Parameters**

  * `edit` -

Methods

- *Dispose*

  `protected` **void Dispose**`( )`

  Clean up any resources being used.

– **Parameters**

* `disposing` -

Extended Information

- Assembly: `CircularMenu`

1.2.27 Class **NoOpFrameModifier**

A frame modifier that performs no action.

Declaration

```
public class NoOpFrameModifier
  : Object
```

Constructors

- *.ctor*

  `public` **NoOpFrameModifier**`( )`

  Initializes a new instance of the NoOpFrameModifier class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this modifier.

- *ModifyFrame*

  `public` **CircularMenu.FrameOptionData ModifyFrame**`( )`

  Simply returns option.

  – **Parameters**

  * `option` -
  * `frameIndex` -
  * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.28 Class **PerimeterUnfoldLayoutManager**

A layout manager that animates options "unfolding" along the circle's perimeter from 0 radians to their final positions.

Declaration

```
public class PerimeterUnfoldLayoutManager
  : CircularLayoutManager
```

Constructors

- *.ctor*

  `public` **PerimeterUnfoldLayoutManager**`( )`

  Initializes a new instance of the PerimeterUnfoldLayoutManager class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this layout manager.

- *ModifyTheta*

  `protected` **float ModifyTheta**`( )`

  Modifies the theta value by moving it slowly from zero to the normal value as the animation advances.

  – **Parameters**

  * `theta` -
  * `optionIndex` -
  * `optionCount` -
  * `frameIndex` -
  * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.29 Class **ReverseFrameCollection**

Defines a frame collection that animates its elements from the final state to the initial state.

Declaration

```
public class ReverseFrameCollection
  : FrameCollection
```

Constructors

- *.ctor*

  `public` **ReverseFrameCollection**`( )`

  Initializes an animation for the provided collection.

  - **Parameters**
    - `options` - The options to animate. This parameter cannot be null, nor can the result of its method be empty.
    - `radius` - The radius to apply to the menu layout.
    - `frameCount` - The number of frames to render. Must be at least one.
    - `layout` - The object used to lay out the menu options during the animation.
    - `modifier` - An object that transforms images during the animation.

Extended Information

- Assembly: `CircularMenu`

1.2.30 Class **ReverseMenuAnimation**

A menu animation that animates backwards.

Declaration

```
public class ReverseMenuAnimation
  : MenuAnimation
```

Constructors

- *.ctor*

  `public` **ReverseMenuAnimation**`( )`

  Initializes a new instance of the ReverseMenuAnimation class.

Methods

- *GetClone*

  `protected` **CircularMenu.MenuAnimation GetClone**`( )`

  Returns a new reverse-frame animation.

- *GetUncachedAnimation*

  `public` **CircularMenu.FrameCollection GetUncachedAnimation**`( )`

  Returns a new set of frames for this animation, and does not cache the result.

  – **Parameters**

  * `menuOptions` - The menu options to animate. Cannot be null.
  * `radius` - The radius to apply to the menu layout.

Extended Information

- Assembly: `CircularMenu`

1.2.31 Class **SpinLayoutManager**

A layout manager that animates options spinning rapidly along the circle's edge and slowing towards their final points. This effect is particularly cool when used with a fade-in frame modifier.

Declaration

```
public class SpinLayoutManager
  : CircularLayoutManager
```

Constructors

- *.ctor*

  `public` **SpinLayoutManager**`( )`

  Initializes a new instance of the SpinLayoutManager class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this layout manager.

- *ModifyTheta*

  `protected` **float ModifyTheta**`( )`

  Modifies the theta value by moving it along the perimeter.

  – **Parameters**

    * `theta` -
    * `optionIndex` -
    * `optionCount` -
    * `frameIndex` -
    * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.32 Class **SpinningStarburstLayoutManager**

A combination of the and managers.

Declaration

```
public class SpinningStarburstLayoutManager
  : CircularLayoutManager
```

Constructors

- *.ctor*

  `public` **SpinningStarburstLayoutManager**`( )`

  Initializes a new instance of the SpinningStarburstLayoutManager class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this layout manager.

- *ModifyRadius*

  `protected` **float ModifyRadius**`( )`

  Performs the starburst modification.

  – **Parameters**

  * `radius` -
  * `optionIndex` -
  * `optionCount` -
  * `frameIndex` -
  * `frameCount` -

- *ModifyTheta*

  `protected` **float ModifyTheta**`( )`

  Performs the spin modification.

– **Parameters**

* `theta` -
* `optionIndex` -
* `optionCount` -
* `frameIndex` -
* `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.33 Class **StandardToolTipData**

Provides settings for common tool tip data. StandardToolTipData provides a number of options for controlling the display of tool-tip information. The tool-tip background is controlled via the , , and properties; text is defined by the , , and properties; the tool-tip border by the , , and properties. Each of the opacity properties controls the transparency of the element and is analogous to the property of the class.

Declaration

public class StandardToolTipData **:** Object

Properties

- *BackgroundColor*

  `public` **System.Drawing.Color BackgroundColor** { `get; set;` }
  The color used to render tool-tip backgrounds.

- *BackgroundOpacity*

  `public` **byte BackgroundOpacity** { `get; set;` }
  Provides the opacity of the background.

- *BorderColor*

  `public` **System.Drawing.Color BorderColor** { `get; set;` }
  The color of the background's border.

- *BorderOpacity*

  `public` **byte BorderOpacity** { `get; set;` }
  Provides the opacity of the tool-tip border.

- *BorderThickness*

  `public` **int BorderThickness** { `get; set;` }
  Provides the thickness of the border.

- *Font*

  `public` **System.Drawing.Font Font** { `get;  set;` }

  The font used to render the text.

- *ForegroundColor*

  `public` **System.Drawing.Color ForegroundColor** { `get;  set;` }

  Provides the color used to render text.

- *ForegroundOpacity*

  `public` **byte ForegroundOpacity** { `get;  set;` }

  Provides the opacity of the foreground text.

- *RenderBackgroundOnEmpty*

  `public` **bool RenderBackgroundOnEmpty** { `get;  set;` }

  Determines if the tool-tip background will be rendered when there is no tool tip.

- *RenderingBackgroundColor*

  `public` **System.Drawing.Color RenderingBackgroundColor** { `get;` }

  Provides the background color with the alpha value applied.

- *RenderingBorderColor*

  `public` **System.Drawing.Color RenderingBorderColor** { `get;` }

  Provides the border color with the alpha value applied.

- *RenderingForegroundColor*

  `public` **System.Drawing.Color RenderingForegroundColor** { `get;` }

  Provides the foreground text color with the alpha value applied.

Constructors

- *.ctor*

  `public` **StandardToolTipData**`( )`

  Initializes a new instance of the StandardToolTipData class.

Extended Information

- Assembly: `CircularMenu`

1.2.34 Class **StandardToolTipRenderer**

This class provides basic functionality for describing and rendering tool tips.

Declaration

public class StandardToolTipRenderer : StandardToolTipData

Properties

- *BackgroundRadius*

  `public` **int BackgroundRadius** { `get; set;` }

  The radius of the tool tip's background.

Constructors

- *.ctor*

  `public` **StandardToolTipRenderer**`( )`

  Initializes a new instance of the StandardToolTipRenderer class.

Methods

- *Render*

  `public` **void Render**`( )`

  Renders the tool tip bacground centered on the provided center point, and draws the tool tip text on top of that.

  – **Parameters**

    * `g` -
    * `toolTip` -
    * `menuCenter` -
    * `renderArea` -

- *RenderEmpty*

  `public` **void RenderEmpty**`( )`

  If is true, behaves like when toolTip is the empty string (""). Otherwise, performs no action.

  – **Parameters**

    * `g` -
    * `menuCenter` -
    * `renderArea` -

Extended Information

- Assembly: `CircularMenu`

1.2.35 Class **StarburstLayoutManager**

A layout manager that animates options "bursting" rapidly from the origin and slowing towards their final points.

Declaration

```
public class StarburstLayoutManager
  :  CircularLayoutManager
```

Constructors

- *.ctor*

  `public` **StarburstLayoutManager**`( )`

  Initializes a new instance of the StarburstLayoutManager class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this layout manager.

- *ModifyRadius*

  `protected` **float ModifyRadius**`( )`

  Increases the radius from zero towards its normal value.

  - **Parameters**
    - `radius` -
    - `optionIndex` -
    - `optionCount` -
    - `frameIndex` -
    - `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.36 Class **UnfoldingStarburstLayoutManager**

A combination of the and types.

Declaration

```
public class UnfoldingStarburstLayoutManager
  : CircularLayoutManager
```

Constructors

- *.ctor*

  `public` **UnfoldingStarburstLayoutManager**`( )`

  Initializes a new instance of the UnfoldingStarburstLayoutManager class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this layout manager.

- *ModifyRadius*

  `protected` **float ModifyRadius**`( )`

  Performs the starbursh modification.

  – **Parameters**

  * `radius` -
  * `optionIndex` -
  * `optionCount` -
  * `frameIndex` -
  * `frameCount` -

- *ModifyTheta*

  `protected` **float ModifyTheta**`( )`

  Performs the unfold modifcation.

– **Parameters**

* `theta` -
* `optionIndex` -
* `optionCount` -
* `frameIndex` -
* `frameCount` -

Extended Information

- Assembly: `CircularMenu`

1.2.37 Class **ZoomInFrameModifier**

A frame modifier that "zooms in" on an image, resizing it from minimum width and height to its current width and height.

Declaration

```
public class ZoomInFrameModifier
  : Object
```

Constructors

- *.ctor*

  `public` **ZoomInFrameModifier**`( )`

  Initializes a new instance of the ZoomInFrameModifier class.

Methods

- *Clone*

  `public` **object Clone**`( )`

  Creates and returns an independent copy of this modifier.

- *ModifyFrame*

  `public` **CircularMenu.FrameOptionData ModifyFrame**`( )`

  Returns a new option that is the result of the image resized by a percentage that is dependent on the current position of the animation.

  – **Parameters**
    * `option` -
    * `frameIndex` -
    * `frameCount` -

Extended Information

- Assembly: `CircularMenu`

Chapter 2

Namespace PixelEffects

Namespace Contents *Page*

Interfaces

Classes

2.1 Interfaces

2.2 Classes

2.2.1 Class **Effects**

This non-instantiable class provides static method for applying graphical effects to images, such as a drop shadow or glow.

Declaration

public class Effects : Object

Methods

- *AugmentPixel*

  `public` **System.Drawing.Color AugmentPixel**`( )`

  Increases the opacity of the provided color, taking care not to increase beyond the provided maximum.

  – **Parameters**
    * `currentValue` - The current pixel value.
    * `step` - The amount to increase the current pixel's value by. Must be at least zero.
    * `maximumValue` - The maximum value for the pixel's alpha channel. Must be between 0 and 255.

- *GetAssemblyImageResource*

  `public` **System.Drawing.Bitmap GetAssemblyImageResource**`( )`

  Attempts to extract the provided resource from the provided assembly as an image. Throws an exception if the image cannot be extracted.

  – **Parameters**
    * `assembly` - The assembly to search for the resource within.
    * `resourceName` - The name of the resource to extract.

- *GetImageDropShadow*

  `public` **System.Drawing.Bitmap GetImageDropShadow**`( )`

Calculates and produces a bitmap representing a drop-shadow for the provided bitmap.

- **Parameters**
  - `original` - The bitmap whose drop-shadow is to be generated.
  - `dropShadowColor` - The color to use for the drop shadow. Cannot be null.
  - `blurRadius` - This value defines the amount of "blur" placed in the drop shadow. Each pixel will cast a shadow having this radius. Must be at least 0. Note that larger values will significantly increase rendering time.
  - `maximumShadowOpacity` - Determines the maximum value for any pixel in the drop shadow's alpha channel. Must be between 0 and 255.
  - `shadowOpacityStep` - Determines the amount by which alpha channel transparency in the drop shadow is augmented by. Larger values will produce a sharper shadow, smaller values will produce a very subtle shadow. Must be at least 1.

- *GetImageDropShadow*

  `public` **System.Drawing.Bitmap GetImageDropShadow**`( )`

  Calculates and returns a drop-shadow image that can be used with the provided bitmap.

  - **Parameters**
    - `image` - The image whose drop-shadow is to be created. Cannot be null.

- *GetScreenCapture*

  `public` **System.Drawing.Bitmap GetScreenCapture**`( )`

  Captures a section of the screen to a new bitmap resource and returns the bitmap.

  - **Parameters**
    - `region` - The specific region of the screen to capture.

- *GetScreenCapture*

  `public` **System.Drawing.Bitmap GetScreenCapture**`( )`

  Returns a screen capture for the entire display area, across all monitors.

- *MakeColorTransparent*

  `public` **void MakeColorTransparent**`( )`

  Makes the provided color fully transparent wherever it appears in the given bitmap. Ignores the alpha channel for comparisson.

  – **Parameters**
    * `image` - The image to mask. Cannot be null.
    * `color` - The color to make transparent.

- *SetImageOpacity*

  `public` **void SetImageOpacity**`( )`

  Adjusts the opacity for every pixel in the provided image so that it is no greater than the provided maximum opacity level.

  – **Parameters**
    * `image` - The image whose opacity levels are to be adjusted. Cannot be null.
    * `maximumOpacity` - The maximum alpha value for any pixel in the image.

Extended Information

- Assembly: `PixelEffects`