# ATF Curve Editor User and Programming Guide

Public.

## Revision History

This guide provides information about using Curve Editor, a graphical editor for authoring curves.

| ATF Version | Revision Date | Author(s) | Changes |
|---|---|---|---|
| 3.6 | Jan 2014 | Gary Staas | "Curve Editor Interface": Remove Cut, Copy, and Paste of points. |
| 3.6 | Oct 2013 | Gary Staas | Overall: Changed to new document format. Made numerous corrections and updates. "About This Guide": Added more info about Curve Editor. Added resources information, document list, and other info. "Authoring Curves": Moved this chapter first. Added info on Basic and Advanced mode. Described curve limit rectangle. "Curve Editor Interface": Added **Options** menu. Added Fit Button. Explained in more detail curve types and auto snap tools. "Curve Editor Classes and Interfaces": New chapter describing CurveEditor component API, including example. |
| 2.8.1 | Dec 2009 | Risa Galant, Laura Lemay | Initial version. |

# Table of Contents

# About This Guide

The Authoring Tools Framework (ATF) Curve Editor is a graphical environment for creating and editing curves and splines, including working with control points and tangents. Use the Curve Editor to create different types of curves, including linear, stepped, clamped, and smooth curves. The Curve Editor is also a Managed Extensibility Framework (MEF) component that you easily add to applications and can interact with programmatically.

A curve is an object that has attributes such as the list of control points, the curve name, and maximum and minimum values.

The Curve Editor interface has a mode similar to Maya's Graph editor with some differences in terminology, such as *points* or *control points* instead of the Maya term *keys*.

The splines used by the Curve editor are Cubic Hermite splines (cspline). For more information, see the Cubic Hermite Spline Wikipedia page.

The following figure shows the Curve Editor panel.

**Figure 1  Curve Editor Panel**
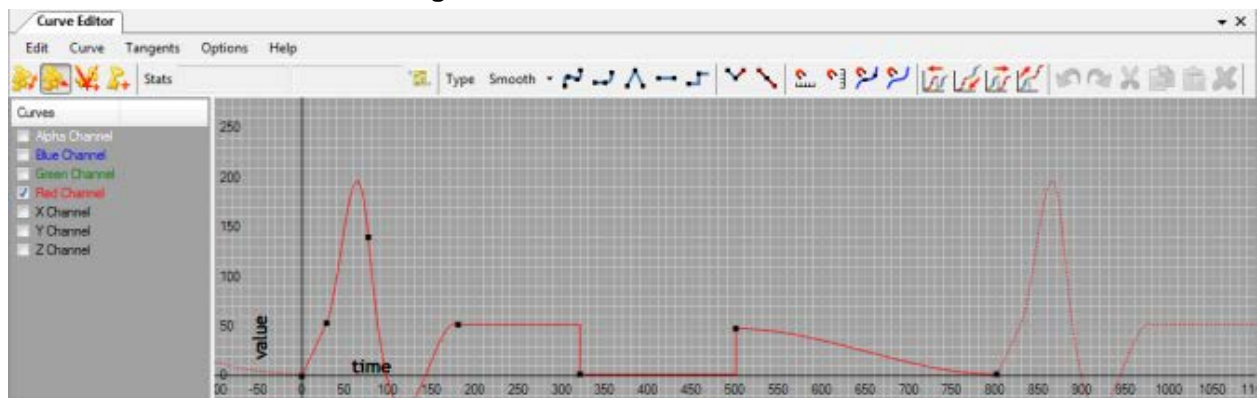


- The Menu Bar and Toolbar components provide commands and tools for manipulating curves and control points in the graph panel of the Curve Editor, including the Stats, Curve Type, Snap, Infinity, and Edit tools.
- The **Curve List Panel** lists the names of all the curves and enables showing or hiding a curve, as well as adding a point to a curve. You cannot add or change the name of a curve with the Curve Editor user interface; you do this using the Curve Editor classes and interfaces. For details, see "Curve Editor Classes and Interfaces".
- The **Graph Panel** shows all the control points and the shapes of the curve(s). You can use the graph panel to add and insert new control points. Note that if you delete all the control points for a curve, you can use the **Curve List Panel** to add a control point and continue working with control points in the **Graph Panel**.

You can work in the way that best suits you: use the toolbar buttons, menus, keyboard controls, or a combination of these methods.

## Who Should Use This Guide

The ATF Curve Editor User Guide is for designers and animators who want to author curves within their applications. You should already be familiar with ATF editors such as the Level Editor and Circuit Editor, game development concepts and terminology, the way games are constructed, and the components that are involved.

## What this Guide Contains

This guide contains these chapters:

"About This Guide" (this chapter): Tells you about this guide and other ATF documentation, what you need to know to work with the Curve Editor, and available resources.

 "Authoring Curves": This chapter shows how to use the Curve Editor. You learn how to select control points and curves, add and move control points, edit tangents, and set curve characteristics.

"Curve Editor Interface": This describes in detail how to use the menu items and tools. It also explains the pre- and post-infinity settings and the tangent curve types.

 "Curve Editor Classes and Interfaces": This overview describes the interfaces and classes associated with the Curve Editor. It shows how to create curves using these classes with an example from the ATF DOM Tree Editor Sample.

## ATF Document Set

The ATF document set is described in Table 1.

**Table 1  ATF Document Set**

| Volume | Version | Comments |
|---|---|---|
| *Getting Started with ATF* | 3.5 | Introduces ATF concepts and ATF samples. |
| *ATF Programmer's Guide* | 3.6 | Shows using ATF to create applications and tools. It includes a Glossary of ATF technical terms. On the ATF wiki. |
| *ATF Programmer's Guide: Document Object Model (DOM)* | 3.5 | Shows how to use the ATF DOM for application data. |
| *ATF Curve Editor User and Programming Guide* | 3.6 | This document. A guide to using the ATF Curve Editor component and programming it. |
| *ATF Refactor User Guide* | 3.3 | Guide to using the ATF Refactor tool. |

In addition, complete API class and method reference documentation for ATF is available as comments in the source code itself and as part of Visual Studio's Intellisense.

## Supported Tools

ATF supports the following tools:

- Visual Studio 2012 and 2010. Visual Studio 2010 and 2012 projects are interchangeable.
- All of the core code in the foundation compiles with C# 3.0. Over time, there will be additional support for C# 4.0 covariance and contravariance, using the `in` and `out` keywords, but a C# 3.0 equivalent will be available.

## Migration From Earlier Releases

The ATF 2.8 components, data models, and so on, are supported in ATF 3. You can continue to build your tools as usual, while taking advantage of new components and features that ATF 3 provides.

There are many class and method name changes from ATF 2.8 to ATF 3. The `AtfRefactor` tool automates the first step in the migration process by updating those class and method names that have well-defined mapping rules and that affect very few lines in the original source code. For more information, see the *ATF Refactor User Guide*.

## Resources

### Other Reading

Some other publications provide useful information:

- *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries (2nd Edition)* by Krzysztof Cwalina and Brad Abrams. This is an excellent book whose guidelines ATF follows in developing a reusable extensible framework. MSDN has much of the book's content as well as the naming guidelines.
- *Programming C# 5.0, Building Windows 8, Web, and Desktop Applications for the .NET 4.5 Framework* by Ian Griffiths, O'Reilly Media.

## Typographical Conventions

This document follows the typographical conventions in Table 2.

**Table 2  Document typographical conventions**

| Font | Used for | Examples |
|------|----------|----------|
| monospace | Paths and file names | `wws_atf\Samples\CircuitEditor` |
| | Scripts | `function ShouldRevive()`<br>`    return m_shouldRevive`<br>`end` |

| Font | Used for | Examples |
|---|---|---|
| | Code elements and code | ```public Command(string description)    {       m_description = description;    }``` |
| | Formatted text, such as XML | ```<?xml version="1.0" encoding="utf-8" standalone="yes"?> <StringLocalizationTable>   <StringItem id="Cut" context="" translation="Cut" />``` |
| bold | User interface elements | **Add** button |
| | Menu item and other paths | **Build > Build Solution** |
| italic | New technical terms | *Document Object Model* |
| | Emphasis | Do *not* do this. |
| | Document titles | *Getting Started with ATF* |
| bold italic | Special emphasis | ***MEF Catalog*** |
| blue | URLs (external) | ATF project home |
| | Links inside a document | "Application Shell Framework" |
| quoted | Exact text | "ActiproSoftare SyntaxEditor" |
| | Names | "Add Layer" |

# 1 Authoring Curves

This chapter describes how to perform common curve authoring tasks using the Curve Editor.

## Basic and Advanced Mode

The Curve Editor has two operation modes: **Basic** and **Advanced**, set by the **Options > Input Mode** menu:

- Use Basic mode for manipulating control points and curves. In this mode, all operations are done with either the **left** or **right mouse buttons**. The mouse wheel is used for zooming. It is simpler than Advanced mode.
- Use Advanced mode for manipulating control points and curves. Input handling for this mode is similar to Maya's Graph Editor. It is more powerful, but it is harder to learn and use compared to Basic mode.

## Selection and Navigation

This section describes how to select control points, tangents, and curves, and how to navigate the Curve Editor's graph panel view.

### Center, Pan, and Zoom

The following table describes the Curve Editor's graph panel view.

**Table 3  View manipulation tasks**

| Task | Operation |
|------|-----------|
| Zoom the view | **Alt + right mouse button** and drag, or rotate the mouse wheel. |
| Pan the view | **Alt + middle mouse button** and drag. In Basic mode, you can also pan with **Alt + left mouse button** and drag. In Advanced mode, **Ctrl + Alt + Right mouse button** and drag. |
| Center the view around the origin | "C" key. |
| Frame the view around the selection or all curves if no selection | "F" key, or **Fit** tool bar button, . |
| Frame the view around all the visible curves, not just selected | "A" key. |

**Select Control Points, Tangents, and Curves**

The following table describes how to select control points, tangents, and curves in the graph panel of the Curve Editor. The Input mode—Basic or Advanced—affects selection operations.

**Table 4  Selection Tasks and Operations**

| Task | Operation |
|---|---|
| Select a single point | Click **left mouse button** on point, or drag a selection rectangle around the point. |
| Select multiple points | Click **left mouse button** and drag around points. |
| Clear selection | Click **left mouse button** on an empty area. |
| Add a point to a selection | Basic mode: **Ctrl + left mouse button** on point. <br> Advanced mode: **Ctrl + Shift + left mouse button** on point. |
| Remove a point from a selection | **Ctrl + left mouse button** on point. |
| Toggle a point's selection state | Basic mode: **Ctrl + left mouse button** on point. <br> Advanced mode: **Shift + left mouse button** on point. |
| Add multiple points to a selection | Advanced mode: **Shift + Ctrl + left mouse button** and drag around points to be added. |
| Remove multiple points from a selection | Advanced mode: **Ctrl + left mouse button** and drag around points. |
| Toggle selection state for multiple points | Advanced mode: **Shift + left mouse button** and drag around points. |
| Select an entire curve | Click **left mouse button** on the curve. <br> A curve limit rectangle is displayed around the curve when it is selected. Control points cannot be beyond these limits. You can drag the curve limit rectangle to change the limits, but not inside the existing control points. |
| Select a tangent | Select a point, and then click its tangent with the **left mouse button**. This also deselects the point. |
| Add tangents to a selection | Basic mode: Click **Ctrl + left mouse button** first on the control point and then on the tangent. <br> Advanced mode: Click **Ctrl + Shift + left mouse button** first on the control point and then on the tangent. |
| Remove a tangent from a selection | **Ctrl + left mouse button** on the tangent. |
| Remove multiple tangents from a selection | Advanced mode: **Ctrl + left mouse button** and drag around the tangents. |
| Select tangent In and tangent Out for the same point | Not supported. |

## Using the Curve List Panel

Use the **Curve List Panel** to show or hide a curve and add control points to a curve.

If you delete all the control points for a curve in the **Graph Panel**, you can use the **Curve List Panel** to add a new control point, after which you can continue working with control points as described in "Working with Control Points".
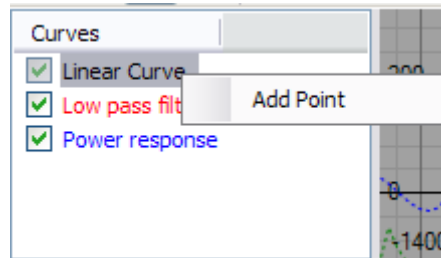
**Show or Hide a Curve**

- To show a curve, check the checkbox next to the curve name.
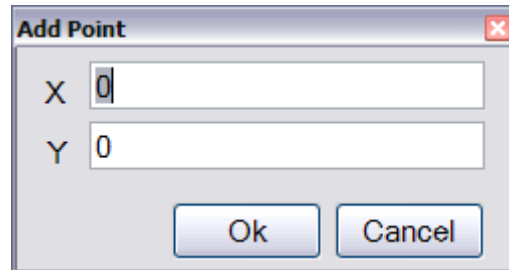- To hide a curve, uncheck the checkbox next to the curve name.

Double-clicking the curve name toggles its visibility state.

**Add a control point to a curve**

(1) Place the mouse over the curve name.

(2) Click the **right mouse button**. The **Add Point** context menu appears.



(3) Select **Add Point**. The **Add Point** dialog appears.



(4) Enter x- and y-coordinates into the **Add Point** dialog, and then click **Ok**.

The new control point appears on the curve in the graph panel. The curve adjusts to include the new point.

## Working with Control Points

You can add a point, insert a point, move or translate points, and scale points around a pivot. See "Selection and Navigation" for information about selecting control points. Some of these operations are only available in Advanced Mode.

**Edit Control Points and Curves**

You can use the **Edit** menu items to edit control points on curves. For details, see "Edit Tools".

**Add a New Point**

In Basic mode, to add a point that is not necessarily on the existing curve:

    (1)    Select curve(s).

    (2)    **Shift** click either the **left** or **middle mouse button** at the new point location.

In Advanced mode, to add a new point, do the following:

    (1)    Click the **Add Control Point** toolbar button, , to change to Add mode.

    (2)    Select the curve(s) to which you want to add a control point (if you have not done so already). To select a curve in this mode, you only need to select one of the points within the curve. If only one curve is displayed, you do not need to select it.

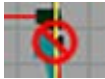    (3)    Click the **middle mouse button** or **Ctrl + right mouse button** to add a new control point.

The new point is inserted at the mouse position for each selected curve, and each curve is adjusted to go through the new point.

**Insert a New Point**

In Advanced mode, to insert a new point onto the existing curve, do the following:

    (1)    Click the **Insert Control Point** toolbar button, , to change to Insert mode.

    (2)    Select the curve to which you want to add a control point (if you have not done so already). To select a curve in this mode, you need only select one of the points within that curve. If only one curve is displayed, you do not need to select it.

    (3)    Click the **middle mouse button** or **Ctrl + right mouse button** to insert a new control point onto a curve.

The new point is inserted at the intersection of each selected curve and the vertical line through the insertion point. Each curve's shape does not change.

If the point cannot be inserted at the specified location, a red "no action" symbol, , appears at the intersection point.

**Move or Translate Points with the Mouse**

In Basic mode, click on a point and drag it with the **left mouse button**. To move several selected points, click on one of the selected points and drag it with the **left mouse button**.

In Advanced mode, do the following to move a selected point:

    (1)    Click the Move Selected Control Point toolbar button, , to change to Move mode.

    (2)    Select the points you want to move. You can select points on more than one curve.

(3)  Drag the selection with the **middle mouse button** to move the selected points. Hold down the **Shift** key before pressing the button to constrain the movement along a single axis. You can also drag the selection with **Ctrl + right mouse button**.

If **Auto Snap to next major x/y tick** is enabled, point(s) are moved to the nearest major grid mark on the x- and/or y-axis.

Control points cannot be moved outside the curve limits. These limits are displayed as a rectangle around the curve when it is selected, as seen in the figure. The rectangle can be resized, but not inside the curve's control points.

**Figure 2  Curve Limit Rectangle**
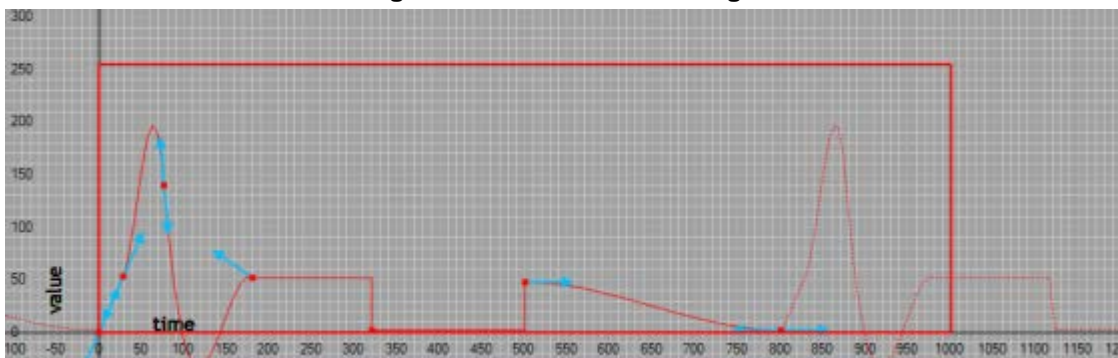


### Move or Translate a Single Point with the Stats Fields

(1)  Select a single point with the **left mouse button**. Its x- and y-coordinates display in the **Stats** fields.

(2)  Edit the x- or y-coordinates of the point in the **Stats** fields and press **Enter** to move the point.

You can also use assignment operations in the **Stats** fields for a single point, as described in the next section. For information on the **Stats** fields, see "Stats Fields".

Control points cannot be moved outside the curve limits. These limits are displayed as a rectangle around the curve when it is selected.

### Move or Translate Multiple Points with the Stats Fields

(1)  Select multiple points.

(2)  Edit the x- or y-coordinates of the selection in the **Stats** field. You can change x-coordinates only for points on different curves.

For information on the **Stats** fields, see "Stats Fields".

Changing the y-coordinate value in the **Stats** field changes the y-coordinate of *all* selected points, altering their curves accordingly. Changing the x- coordinate aligns the selected points of different curves along the x-axis, altering their curves accordingly.

You can also use simple assignment operations in the **Stats** fields for both single and multiple points, as listed in the following table. You can use a negative number for the value.

**Table 5  Stats Fields Assignment Operations**

| Operation | Function |
|---|---|
| value | The entered value replaces the existing value. |
| +=value | The entered value is added to the existing value. |
| *=value | The entered value multiplies the existing value. |
| -=value | The entered value is subtracted from the existing value. |
| /=value | The entered value divides the existing value. |

Control points cannot be moved outside the curve limits. These limits are displayed as a rectangle around the curve when it is selected.

### Scale Points around a Pivot

In Advanced mode, to scale points around a pivot point:

(1)  Click the **Scale Selected Control Points** toolbar button, , to change to Scale mode.

(2)  Select the points you want to scale.

(3)  Drag the selection with the **middle mouse button** to scale uniformly along both axes. You can also use **Ctrl + right mouse button** and drag.

Hold down the **Shift** key before moving the cursor to constrain scaling along a single axis. Note that scaling occurs around a pivot point at the cursor position when the pivoting begins. If **Auto Snap to next major x/y tick** is enabled, the pivot point is the nearest major grid mark in x and/or y to the beginning cursor position.

## Working with Tangents

See "Selection and Navigation" for information about selecting tangents.

### Adjust the Tangent for a Curve

In Basic mode, to adjust the tangent:

(1)  Select control point(s) and then select their tangent(s).

(2)  Click one of the selected tangents and drag to adjust selected tangent(s).

In Advanced mode:

(1)  Click the **Move Selected Control Point** toolbar button, , to change to Move mode.

(2)  Select the point(s) and then the tangent(s) you want to adjust.

(3)   Drag the selection with the **middle mouse button** to move the selected tangent(s). You can also use **Ctrl + right mouse button** and drag.

### Break Tangents for One or More Points

Breaking a tangent means that the in and out tangents are no longer coupled, so you can adjust them separately.

(1)   Select point(s).

(2)   Click the **Break Tangents** toolbar button,  .

### Unify tangents for one or more points

Unifying tangents couples the in and out tangents, so they adjust together. The angle between the tangents is maintained thereafter when adjusting them.

(1)   Select point(s).

(2)   Click the **Unify Tangents** toolbar button,  .

### Set the Tangent Type

You can set the tangent for a point, which determines the kind of curve coming into and leaving the point.

(1)   Select one or more points.

(2)   Select the tangent type immediately under the **Tangents** menu to set both the in and out tangents. To set only the in or out tangent, select the tangent type under either the **Tangents > In Tangent** or **Tangents > Out Tangent** menu.

You can also click one of the Tangent type toolbar buttons to set both the in and out tangents for the selected point(s).

For more information about tangents and tangent types, see "Tangents Tools".

## Set Curve Pre and Post Infinities

Setting the **Pre-Infinity** and **Post-Infinity** settings for a curve determines what the curve does before the first and after the last control point.

(1)   Select one or more curves. You can select the entire curve or just one point on that curve.

(2)   Select an infinity value from the **Curve > Pre-Infinity** or **Curve > Post-Infinity** menus, or from the toolbar buttons.

The infinity values are **Constant**, **Cycle**, **CycleWithOffset**, **Oscillate**, and **Linear**. For a description of these settings, see "Pre and Post Infinities".

# 2 Curve Editor Interface

This section describes the user interface and its curve tools.

## Edit Tools

Edit tools under the **Edit** menu are standard clipboard editing commands: **Delete**, **Undo**, and **Redo**. These commands apply to control points and curves, rather than text. All the editing commands can also be accessed from the Curve Editor context menu and by using standard shortcut keys, for example, **Ctrl+Z** for Undo.

The following table describes the Edit menu commands.

**Table 6  Edit Menu**

| Command | Use |
|---------|-----|
| **Undo** | Undo the last Edit command. |
| **Redo** | Redo the last Edit command that was undone. |
| **Delete** | Delete the selected point(s). Select point(s), and then select **Delete**. |

## Curve Menu

### Pre and Post Infinities

Infinities extrapolate a curve beyond the first and last control points of the curve. The **Curve** menu's **Pre-Infinity** and **Post-Infinity** settings allow you to specify how a set of control points cycle backward and forward through time infinitely, for example, to show a spinning gear or to play a background sound such as ocean waves. The **Pre-Infinity** setting defines the curve's behavior before the first (smallest x-coordinate) control point. The **Post-Infinity** setting defines the curve's behavior after the last (largest x-coordinate) control point. For directions on how to set **Pre-Infinity** and **Post-Infinity**, see "Set Curve Pre and Post Infinities".

The following table describes the **Cycle** (Infinity) toolbar buttons.

**Table 7  Cycle (Infinity) Toolbar Buttons**

| Toolbar Button | Effect |
|----------------|--------|
|  | **Cycle Before**: Copy and repeat the curve infinitely before the first control point. Same as the **Curve > Pre-Infinity > Cycle** menu item. |
|  | **Cycle Before with Offset**: Copy and repeat the curve infinitely before the first control point, offsetting the copied curve to join the original curve, creating a continuous curve. Same as the **Curve > Pre-Infinity > CycleWithOffset** menu item. |

| Toolbar Button | Effect |
|---|---|
|  | **Cycle After**: copy and repeat the curve infinitely after the last control point. Same as the **Curve > Post-Infinity > Cycle** menu item. |
|  | **Cycle After with Offset**: Copy and repeat the curve infinitely after the last control point, offsetting the copied curve to join the original curve, creating a continuous curve. Same as the **Curve > Post-Infinity > CycleWithOffset** menu item. |

The Curve menu **Pre-Infinit**y and **Post-Infinity** items have the following appearance:

**Table 8  Curve Menu Pre-Infinity and Post-Infinity Appearance**

| Setting | Effect |
|---|---|
| **Constant** | Maintain the y-coordinate value of the end control point.<br> |
| **Cycle** | Copy and repeat the curve infinitely. Same as the **Cycle Before/After** toolbar buttons.<br> |
| **CycleWithOffset** | Copy and repeat the curve, offsetting the copied curve to join the original curve, creating a continuous curve. Same as the **Cycle Before/After with Offset** toolbar buttons.<br> |

| Setting | Effect |
|---|---|
| Oscillate | Copy and repeat the curve by reversing its values at the end of each cycle and then repeating the original curve, to create a continuous alternate forwards and backwards effect. |
| Linear | Maintain the tangent of the end control point. Use the control point's tangent information to project a linear curve infinitely beyond the original curve. |

## Tangents Tools

A tangent describes the curve segment tangent to a control point. You can define the tangent on both sides of a control point: the tangent leading into a control point (the *in tangent*) and the tangent leading out of a control point (the *out tangent*). The **Tangents** menu provides commands that allow you to manipulate in or out tangents. You can define the shape of a tangent using the **Tangents** menu items or by clicking the Curve toolbar buttons. For directions on how to set the tangent for a control point, see "Set the Tangent Type".

The following table lists and describes the **Tangents** tools and menu items. It also describes the curve types: spline, linear, clamped, stepped, and flat.

Selecting a tangent type from either immediately under the **Tangents** menu or the Curve toolbar sets *both* the in and out tangents of selected points. Use a menu item from either the **Tangents > In Tangent** or **Tangents > Out Tangent** to set the tangent on only one side.

Generally the shape of a curve between two points is affected by both tangents between them—except for a **Stepped** tangent. When a point has a **Stepped** tangent, its out curve is a horizontal segment until the next control point, regardless of the in tangent of that next control point.

**Table 9  Tangents Tools**

| Menu Item | Toolbar Button | Effect |
|---|---|---|

| Menu Item | Toolbar Button | Effect |
|---|---|---|
| **Spline** | | Change the tangent type for selected control points to **Spline**. **Spline** is a curve that is smooth between the point before and the point after the selected control point. The tangents at the end of the curve are both at the same angle, which ensures that the curve smoothly enters and exits the control point. |
| **Linear** | | Change the tangent type for the selected control points to **Linear**. **Linear** is a curve that is a straight line joining two control points. |
| **Clamped** | | Change the tangent type for the selected control points to **Clamped**. **Clamped** is similar to **Spline**. A control point's tangents are splines unless the values of two adjacent control points are very close. In this case, the out tangent of the first control point and the in tangent of the second control point are linear: the curve is clamped linear. |
| **Stepped** | | Change the tangent type for the selected control points to **Stepped**. **Stepped** is a curve where the out tangent is a horizontal segment. No interpolation is applied: the y-coordinate changes at control points without gradation. |
| **Flat** | | Change the tangent type for the selected control points to **Flat**. **Flat** creates a flat tangent: horizontal at the control point. |
| None | | Break the tangents of the selected control points. This enables manipulation of the in and out tangent handles individually, so you can adjust the curve segment entering the point without affecting its opposite handle. |
| None | | Unify the tangents of the selected control points. This causes the manipulation of an in or out tangent handle to affect its opposite handle equally. Unify tangents retains the relative position of the tangent handles, even after tangents are individually adjusted. |
| **In Tangent** | None | Change the tangent leading *into* a control point on a curve. Select one of **Spline**, **Linear**, **Clamped**, or **Flat** tangency. |
| **Out Tangent** | None | Change the tangent leading *out* of a control point on a curve. Select one of **Spline**, **Linear**, **Clamped**, or **Flat** tangency. |

## Stats Fields

The **Stats** fields show a selected control point's x- and y-coordinate values. For example, the following figure shows the coordinates of the selected point at x (10) and y (-10).

**Figure 3  Stats fields showing point coordinates**
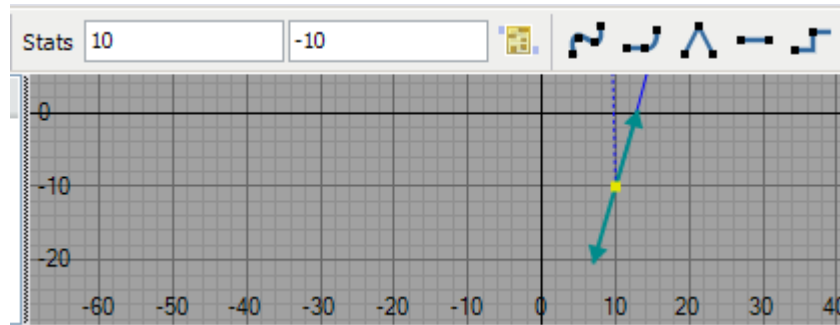


You can edit a selected control point's coordinate values by entering values directly from the keyboard. See "Move or Translate a Single Point With The Stats Fields" or "Move or Translate Multiple Points With The Stats Fields" for more information.

## Fit Button

Clicking the Fit button, , or pressing the "F" key frames the view around the selection or all curves if nothing is selected.

## Curves Type

The curves **Type** drop down menu on the toolbar can be used to temporarily change all curves' appearances:

**Figure 4  Curves Type Drop Down Menu**
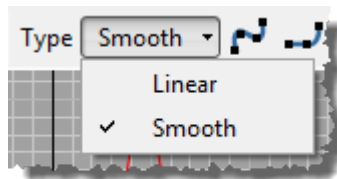


The drop down has two settings described in the table.

**Table 10  Curves Types**

| Command | Use |
|---------|-----|
| **Linear** | Change all curves to have **Linear** tangents. |
| **Smooth** | Do not change curve appearance (default). |

## Auto Snap Tools

Auto Snap forces control points in the graph panel to the next major x- or y-axis grid tick, to other points, or to curves. The following table lists and describes the Auto Snap toolbar buttons.

**Table 11  Auto Snap Toolbar Buttons**

| Button | Use |
| --- | --- |
| | Auto Snap to next major X tick when either moving control points or scaling points around a pivot. |
| | Auto Snap to next major Y tick when either moving control points or scaling points around a pivot. |
| | Auto Snap to point. When moving a control point, snap to a point in another curve. |
| | Auto Snap to curve. When moving a control point, snap to another curve. |

## Options Menu

The Options menu lets you set Curve Editor options, described in the following table. All these options persist across sessions.

**Table 12  Curve Editor Options**

| Menu | Submenu | Use |
| --- | --- | --- |
| **Input Mode** Describes the use mode for Curve Editor. All shortcuts are the same for both modes. | **Basic** | Set Basic mode for manipulating control points and curves, which provides a simpler interface than Advanced mode. In this mode, all operations are done with either the **left** or **right mouse buttons**. The mouse wheel is used for zooming. |
| | **Advanced** | Set Advanced mode for manipulating control points and curves. Input handling for this mode is similar to Maya's Graph Editor. It is more powerful, but it is harder to learn and use compared to Basic mode. |
| **Lock Origin** Describes where the graph origin is locked | **Free** | Not locked. The graph can be panned along the x- and y-axes. (Default) |
| | **Center** | At the center of the graph window. Panning is disabled. |
| | **Left** | At the left of the graph window, preserving the current position of the x-axis. The y-axis is free to pan. |
| | **LeftTop** | At the left top of the graph window. Panning is disabled. |

| Menu | Submenu | Use |
|---|---|---|
| | **LeftMiddle** | At the left middle of the graph window. Panning is disabled. |
| | **LeftBottom** | At the left bottom of the graph window. Panning is disabled. |
| **Flip Y-Axis** | | Flip the y-axis for all curves. |

## Help Menu

The Help menu provides the following command:

**Table 13  Help Menu**

| Command | Use |
|---|---|
| **Quick Help** | Display the Curve Editor Quick Help file, which is a condensed version of this document. |

# 3 Curve Editor Classes and Interfaces

Curve Editor is implemented as the Managed Extensibility Framework (MEF) component `CurveEditor`, so an application simply needs to include it in its `TypeCatalog` to use it.

The other requirement is that the application provide a selectable object that has curves associated with it. A curve implements the `ICurve` interface, which describes curve properties. This also means that the application must provide a selection context.

Curve Editor is built on top of the `CurveEditingControl` class, which uses a `CurveCanvas` class object deriving from `Cartesian2dCanvas`. These powerful controls provide the capabilities needed to display curves.

## CurveEditor Operation

The `CurveEditor`'s `IInitializable.Initialize()` method subscribes to the `ActiveContextChanged` event. When the active context changes, `CurveEditor` attempts to adapt the context to an `ISelectionContext`, and if successful, subscribes to the `SelectionChanged` event for this context.

When an `SelectionChanged` event occurs, `CurveEditor` calls its `GetCurves()` method, which attempts to adapt the selection to `ICurveSet`, a set of `ICurve` objects. If this succeeds, the `CurveEditingControl`'s `Curves` property is set to this `ICurve` collection. Doing this gives `CurveEditingControl` what it needs to display and work with the given curves.

The [ATF DOM Tree Editor Sample](#) uses the `CurveEditor` component. This sample has an "Animation" type that has curves as child types. The DOM adapter for the "Curve" type, `Curve`, implements the `ICurve` interface. The "Animation" type has the DOM adapter `UIAnimation`, which implements `ICurveSet`. This allows a selected "Animation" object to be adapted to the `ICurveSet` used by `CurveEditingControl`.

## Curve Interfaces

The file `CurveInterfaces.cs` defines several curve interfaces.

### IControlPoint

A curve is mainly defined by its control points and their attributes. `IControlPoint`'s main properties describe a control point:

- `Parent`: Get the curve (`ICurve`) holding the control point.
- `X`, `Y`: Get or set the point's x- and y-coordinates.

- `TangentIn`, `TangentOut`: Get or set the tangent in and out vector as a `Vec2F`. The tangent vector is the first derivative of the curve. The *tangent in* is the derivative before the control point; the *tangent out* the derivative after the control point. Thus the vector's direction is tangent to the curve and its magnitude is the curve's slope. The curve renderer uses curve evaluators to draw the curve, and the evaluators use the tangents to compute the curve's y-coordinate for a given x-coordinate.
- `TangentInType`, `TangentOutType`: Get or set the tangent in and out `CurveTangentTypes`.
- `BrokenTangents`: Get or set whether the tangent is broken, that is, the tangent in and out are not coupled to each other, so that the tangents can be adjusted separately.

The `Clone()` method copies a control point.

### ICurve

The interface `ICurve` mostly specifies curve properties, such as the following:

- `ControlPoints`: Get the curve's control points as a `ReadOnlyCollection<IControlPoint>`.
- `MinX`, `MaxX`: Get or set the minimum and maximum x-coordinate values.
- `MinY`, `MaxY`: Get or set the minimum and maximum y-coordinate values.
- `CurveColor`: Get or set the curve `Color`.
- `PreInfinity`, `PostInfinity`: Get or set the `CurveLoopTypes` to specify the curve before and after the end control points.

`ICurve` also contains useful curve construction methods:

- `CreateControlPoint()`: Create a control point.
- `AddControlPoint()`: Add a given control point to a curve.
- `InsertControlPoint()`: Insert the control point with the specified index into a curve.
- `RemoveControlPoint()`: Remove the given control point.
- `Clear()`: Remove all the curve's control points.

### ICurveSet

This interface describes a set of `ICurve` objects. Its `Curves` property gets an `IList<ICurve>` of all the set's curves. Note that `CurveEditingControl` needs an `ICurveSet` for the curves it displays and works with.

## Curve Enumerations

The `CurveEnums.cs` file lists various types used with control points and curves.

### CurveLoopTypes

These types describe the **Pre-Infinity** and **Post-Infinity** settings for curves. The values correspond precisely to the ones used in the Curve Editor. For a description, see "Pre and Post Infinities".

### CurveTangentTypes

The values in this enumeration correspond to the tangent types described in "Tangents Tools", except for these additional ones:

- `Fixed`: The tangent does not change when the curve changes as control points are added or moved; the tangent and its angle are fixed. The tangent type is set to `Fixed` after the tangent is adjusted, as described in "Adjust the Tangent for a Curve". This allows the user to set the tangent angle precisely as desired.
- `SteppedNext`: Similar to `Stepped`, but the curve immediately steps to the *next* control point's y-coordinate value, instead of the current point's value. Not implemented.
- `Plateau`: Similar to `Spline`, but the curve is flattened so that the minimum and maximum y-coordinate values are at the control points. Not implemented.

### PointSelectionRegions

This enumeration describes selections related to control points:

- `None`: No control point selected.
- `TangentIn`: Control point's tangent in selected.
- `TangentOut`: Control point's tangent out selected.
- `Point`: Control point selected.

### InterpolationTypes

Curve editor uses two kinds of curves between points: splines and linear curves:

- `None`: No defined curve.
- `Hermite`: Cubic hermite spline, also known as a cspline.
- `Linear`: Linear curve.

## Curve Classes

The `CurveEditor` component is supported by several other classes. You do not need to directly instantiate these classes: `CurveEditor` takes care of that. `CurveUtils` provides static methods useful for applications.

### CurveEditingControl

This control serves as a container for a `CurveCanvas` control that the `CurveEditingControl` constructor creates. Its `Init()` method sets up nearly all the

controls seen in the Curve Editor, including the **Curve**, **Tangents**, **Options**, and **Help** menus and their items; the tool buttons, the **Stats** fields, and so on. It handles all the event actions associated with these UI elements. It also handles the **Help** and the **Add Point** dialogs.

### CurveCanvas

`CurveCanvas` is the control that actually displays in Curve Editor. It provides a generic canvas on which to draw curves. `CurveCanvas` provides the **Edit** menu and its menu items, as well as their implementation. It derives from `Cartesian2dCanvas`. It handles curve operations, such as tangents, **Pre-Infinity** and **Post-Infinity** curve drawing, and snapping. It handles selections, and its history context handles undo and redo operations. It performs scaling points around a pivot. `CurveCanvas` also provides the Basic and Advanced modes for the user interface, including handling the Advanced Mode's tool buttons for scaling, moving, inserting, and adding control points. Its `OnPaint()` method uses the `CurveRenderer` class to actually render curves.

### CurveRenderer

The `CurveRenderer` class handling curve drawing and picking. It uses its `DrawCurve()`, `DrawControlPoints()`, and `DrawArrow()` methods to draw curves, control points, and tangents using GDI. `DrawCurve()` uses either a `LinearCurveEvaluator` or `HermiteCurveEvaluator` to interpolate the curve. `LinearCurveEvaluator` simply draws linear line segments between points. `HermiteCurveEvaluator` computes y-coordinates for a sequence of points along the x-axis with a small step between them, and then draws lines between the points to approximate the curve.

### CurveUtils

`CurveUtils` provides a set of static methods that applications can use to create curves. The most useful method is `ComputeTangent()`, which computes tangents for all control points in a given curve by calling `ComputeTangentOut()` and `ComputeTangentIn()` to compute these tangents for a given control point on the curve. As previously mentioned, the tangent vector is the first derivative of the curve at the point, and it may be different on each side of the point because the tangent types may be different, so the curves are different. The `AddControlPoint()` method is used by `CurveCanvas` to add control points, as ultimately directed by `CurveEditingControl`. Both `CurveCanvas` and `CurveRenderer` use `CreateCurveEvaluator()` to get an evaluator, which computes x- and y-coordinate values for curves so they can be drawn. `CreateCurveEvaluator()` in turn returns an instance of either the `LinearCurveEvaluator` or `HermiteCurveEvaluator` class to do the actual work.

## Example

The [ATF DOM Tree Editor Sample](#) provides a Curve Editor to show curves associated with its "Animation" type objects. There are also "Curve" and "Control point" types in the application's data model. The DOM adapter for the "Animation" type is `UIAnimation`,

and it implements `ICurveSet`. The DOM adapters for "Curve" and "Control point" types, `Curve` and `ControlPoint`, implement `ICurve` and `IControlPoint` respectively.

`UIAnimation`'s `OnNodeSet()` method is called when an "Animation" type `DomNode` gets initialized, and this occurs when an Animation object is first selected. Here's the section that creates a curve labeled "X Channel":

```
// add x channel
Curve curve = (new
    DomNode(UISchema.curveType.Type)).As<Curve>();
curve.Name = "X Channel";
curve.MinX = 0;
curve.MaxX = 1000;
curve.MinY = float.MinValue;
curve.MaxY = float.MaxValue;
curve.CurveColor = Color.Black;
curve.PreInfinity = CurveLoopTypes.Cycle;
curve.PostInfinity = CurveLoopTypes.Cycle;
curve.XLabel = "time";
curve.YLabel = "value";

IControlPoint cp = curve.CreateControlPoint();
cp.X = 0;
cp.Y = 1;
curve.AddControlPoint(cp);
cp = curve.CreateControlPoint();
cp.X = 500;
cp.Y = 10;
curve.AddControlPoint(cp);
CurveUtils.ComputeTangent(curve);
m_curves.Add(curve);
```

On the first line, the variable `curve` is adapted to `Curve`, so it supports the `ICurve` interface. The subsequent lines set `ICurve` properties. For example, the `PreInfinity` property is set to `CurveLoopTypes.Cycle`, using the `CurveLoopTypes` enumeration.

The next section adds control points to the curve. The `Curve.CreateControlPoint()` method creates a `DomNode` of type "Control point" that is therefore adapted to the `IControlPoint` interface. The point's x- and y-coordinates are set to 0 and 1. The control point is then added to the curve's list by calling `Curve.AddControlPoint()`, which has this simple implementation to add the point to a list:

```
public void AddControlPoint(IControlPoint cp)
{
    m_pointList.Add(cp);
}
```

Recall that the `ICurve` property `ControlPoints` gets the list of control points, and this property is implemented this way:

```
public ReadOnlyCollection<IControlPoint> ControlPoints
{
```

```
        get { return m_readonlyPointList; }
    }
```

And `m_readonlyPointList` is set like this:

```
m_readonlyPointList = new
    ReadOnlyCollection<IControlPoint>(m_pointList);
```

Thus `m_readonlyPointList` contains all the curve's points, and they can be accessed through the `ICurve.ControlPoints` property.

After adding a second control point and adding it to the curve, `CurveUtils.ComputeTangent()`is called to set up the tangents for all control points in a given curve.

Finally, the last line adds the configured curve to the `IList<ICurve> m_curves` variable. The important thing about `m_curves` is that it provides the value for the `ICurveSet.Curves` property that `UIAnimation` implements:

```
public IList<ICurve> Curves
{
    get { return m_curves; }
}
```

This is key, because the `CurveEditor` component needs to adapt the selected object to an `ICurveSet` to get the list of curves it ultimately provides to the other curve handling classes. Each curve in the set implements `ICurve`. Thus a selected "Animation" object can provide the `ICurveSet` curve list for the Curve Editor.

The `OnNodeSet()` method sets up several other curves in a similar way. It does something extra with the "Red Channel" curve:

```
cp = curve.CreateControlPoint();
cp.TangentInType = CurveTangentTypes.Fixed;
cp.TangentOutType = CurveTangentTypes.Fixed;
cp.X = 76.55121f;
cp.Y = 139.914139f;
cp.TangentIn = new Vec2F(0.100534618f, -0.994933546f);
cp.TangentOut = new Vec2F(0.100534618f, -0.994933546f);
curve.AddControlPoint(cp);
```

The code explicitly specifies the additional control point properties `TangentInType`, `TangentOutType`, `TangentIn`, and `TangentOut`. The `CurveUtils.ComputeTangent()` method does not compute tangents when the tangent type is `Fixed`. This allows the application to specify the exact tangent it wants, as in this case.

# Shortcut Reference

The following table lists the keyboard shortcuts available from the Curve Editor.

**Table 14  Keyboard Shortcuts**

| Operation | Shortcut |
|---|---|
| Undo | **Ctrl + Z** |
| Redo | **Ctrl + Y** |
| Move (translation) mode in Advanced mode | W |
| Scale mode in Advanced mode | R |
| Frame selected items | F |
| Frame all, not just selected | A |
| Pan to origin | C |
| Delete | **Delete** |