# Creating Shape Templates in ShopTools

By Daniel Patterson

## Introduction

To completely replace the power tools in your shop like **Drill Press**, **Table Saw**, **Miter Saw**, and **Band Saw**, this version of ShopTools is only pre-loaded with the following templates.

* Move in a straight line without cutting, either using an end coordinate or an angle and a length.
* Cut in a straight line, either using an end coordinate or an angle and a length.
* Cut a two-segment line using coordinates.
* Drill a hole at a coordinate.

To emulate complex patterns along the lines of what can be achieved with a **Scroll Saw**, **Jigsaw**, or **Sabre Saw**, you can also draw up a true-scale shape in a popular SVG editor, then import that SVG file as a shape.

What we haven't mentioned yet, however, is the fact that you can also define complex, reusable shapes of any kind, adding them as templates to the ShopTools toolbox, and creating a set of practical jigs that can be reused any number of times and shared with other ShopTools users as well.

This article explores the creation and implementation of shape templates within ShopTools.

## The Basic Shape-Related Actions

First, there are the basic shapes supported by ShopTools, which are referred to as *Operation Actions* for reasons we will explore shortly. In the meantime, the full list of possible operation actions follows, separated into these categories.

* [**Curved Paths**](#_Curved_Paths). Unfilled arcs and curved lines.
* [**Circular Paths**](#_Circular_Paths). Unfilled circle patterns.
* [**Elliptical Paths**](#_Elliptical_Paths). Unfilled ellipse patterns.
* [**Linear Paths**](#_Linear_Paths). Single straight lines.
* [**Polygonal Paths**](#_Polygonal_Paths). Unfilled freeform polygons.
* [**Rectangular Paths**](#_Rectangular_Paths). Unfilled uniform rectangles.
* [**Circular Fills**](#_Circular_Fills). Filled circles.
* [**Elliptical Fills**](#_Elliptical_Fills). Filled ellipses.
* [**Polygonal Fills**](#_Polygonal_Fills). Filled freeform polygons.
* [**Rectangular Fills**](#_Rectangular_Fills). Filled uniform rectangles.
* [**Non-Drawing Movements**](#_Non-Drawing_Movements). Movements that allow the tool to change places.
* [**Pin-Point Cuts**](#_Pin-Point_Cuts). Drilling motions.

### Curved Paths

Unfilled arcs and curved lines.

* **DrawArcCenterOffsetXY**. Draw an arc from the specified start coordinate to the point nearest the specified end point, given the radius from the center to the starting offset.
* **DrawArcCenterOffsetXYAngle**. Draw an arc using a center coordinate, an offset starting coordinate, and a sweep angle.
* **DrawArcCenterRadiusStartSweepAngle**. Draw an arc using a center coordinate, a radius, a start angle, and a sweep angle.

### Circular Paths

Unfilled circle patterns.

* **DrawCircleCenterDiameter**. Draw a circle using a center reference point and diameter.
* **DrawCircleCenterRadius**. Draw a circle using a center reference point and radius.
* **DrawCircleDiameter**. Draw a circle using corner X, Y references and diameter.
* **DrawCircleRadius**. Draw a circle using corner X, Y references and radius.

### Elliptical Paths

Unfilled ellipse patterns.

* **DrawEllipseCenterDiameterXY**. Draw an ellipse using a center reference point and independent diameters.
* **DrawEllipseCenterRadiusXY**. Draw an ellipse using a center reference point and independent radii.
* **DrawEllipseDiameterXY**. Draw an ellipse using corner X, Y references and independent diameters.
* **DrawEllipseLengthWidth**. Draw an ellipse using corner X, Y starting references, length, and width.
* **DrawEllipseRadiusXY**. Draw an ellipse using corner X, Y references and independent radii.
* **DrawEllipseXY**. Draw an ellipse using starting and ending X, Y coordinates.

### Linear Paths

Single straight lines.

* **DrawLineAngleLength**. Draw a line using a point, an angle, and a length.
* **DrawLineLengthWidth**. Draw a line using a point, a length, and a width.
* **DrawLineXY**. Draw a line using two points.

### Polygonal Paths

Unfilled freeform polygons.

* **DrawPath**. Draw the path specified in the PathData property.

### Rectangular Paths

Unfilled uniform rectangles.

* **DrawRectangleCenterLengthWidth**. Draw a rectangle using a center coordinate, length, and width.
* **DrawRectangleLengthWidth**. Draw a rectangle using a corner, length, and width.
* **DrawRectangleXY**. Draw a rectangle using two corner points.

### Circular Fills

Filled circles.

* **FillCircleCenterDiameter**. Fill a circle using the center reference and a diameter.
* **FillCircleCenterRadius**. Fill a circle using its center reference and a radius.
* **FillCircleDiameter**. Fill a circle using its corner reference and a diameter.
* **FillCircleRadius**. Fill a circle using its corner reference and a radius.

### Elliptical Fills

Filled ellipses.

* **FillEllipseCenterDiameterXY**. Fill an ellipse using a center reference point and independent diameter values.
* **FillEllipseCenterRadiusXY**. Fill an ellipse using a center reference point and independent radii.
* **FillEllipseDiameterXY**. Fill an ellipse using a corner point and independent diameter values.
* **FillEllipseLengthWidth**. Fill an ellipse using corner X, Y starting references, length, and width.
* **FillEllipseRadiusXY**. Fill an ellipse using a corner point and independent radii.
* FillEllipseXY. Fill an ellipse using starting and ending X, Y coordinates.

### Polygonal Fills

Filled freeform polygons.

* **FillPath**. Fill the path specified in the PathData property.

### Rectangular Fills

Filled uniform rectangles.

* **FillRectangleCenterLengthWidth**. Fill a rectangle using a center coordinate, length, and width.
* **FillRectangleLengthWidth**. Fill a rectangle using one corner, width, and height.
* **FillRectangleXY**. Fill a rectangle using two corners.

### Non-Drawing Movements

Movements that allow the tool to change places.

* **MoveAngleLength**. Move the bit, without cutting, at an angle, by a specified length.
* **MoveXY**. Move the tool, without cutting, to the specified coordinate.

### Pin-Point Cuts

Drilling motions.

* **PointXY**. Drill at a point represented by the X and Y coordinates.

## The Basics of ShopTools Objects

// TODO: !1 - Stopped here...

// TODO: Describe how objects are defined as being multiple operations in a single template, allowing for complex parameterized shapes being handled as single objects.

## Creating A Pattern Template File

The easiest way to define one or more of the shapes from the previous section as something you are able to select in ShopTools is to create a Pattern Template file, save it with a **.patterns.json** extension then import it for use with the menu option **File / Import / Patterns**.

In this section, we'll look at the structure of a pattern template file. First, as you might have noticed in the filename extension above, this is going to be a JSON file, which means that although it is structured like data, it should also be somewhat easy to read to the human eye.

First, we'll go through the list of possible properties, then finish by looking at a few examples.

### Pattern Template Properties

A pattern template contains several properties, some of which are optional, and some of which are required. Each of the following property descriptions contains a notice of whether it is optional or required.

#### Property - Display Format

**Usage**: Optional

{Line}

**Description**

The custom display format to use when preparing the string information for this item to be displayed in general areas like the status bar or descriptions of the property's current value.

{Line}

**Syntax, BNF**

The following loose definitions of the DisplayFormat string, in Backus-Naur Form, illustrate the general function of the DisplayFormat string.

<interpolation\_content> ::= <text\_segment> | <interpolation\_expression>

<interpolation\_delimiter> ::= <interpolation\_start\_delimiter> | <interpolation\_end\_delimiter>

<interpolation\_start\_delimiter> ::= '{'

<interpolation\_end\_delimiter> ::= '}'

<format\_type> ::= '+-' | 'Abbreviation'

<text\_segment> ::= <any\_character\_except\_interpolation\_delimiter\_or \_quote>

<interpolation\_expression> ::= <interpolation\_start\_delimiter> <variable\_name> [':' <format\_type>] <interpolation\_end\_delimiter>

{Line}

**Syntax Examples**

Here are some examples of how the the DisplayFormat property uses interpolated strings to format the textual output of the values related to an object created from a template.

* {TemplateName} - {OffsetXOrigin:Abbreviation}{OffsetX:+-}, {OffsetYOrigin:Abbreviation}{OffsetY:+-}. Displays the template name followed by a hyphen and the abbreviation of the OffsetX Origin setting then OffsetX value, using a leading polarity sign.
* {TemplateName} - {Angle}, {Length}. Displays the template name followed by a hyphen, then the angle and the length of the shape.
* {TemplateName} - {EndOffsetXOrigin:Abbreviation}{EndOffsetX:+-}, {EndOffsetYOrigin:Abbreviation}{EndOffsetY:+-}. Displays the template name followed by a hyphen, then the abbreviation of the EndOffsetX Origin setting and the EndOffsetX value, a comma, then the EndOffsetY Origin setting and finally, the EndOffsetY value.