ENTERPRISE ARCHITECT

用户指南系列

动态图表

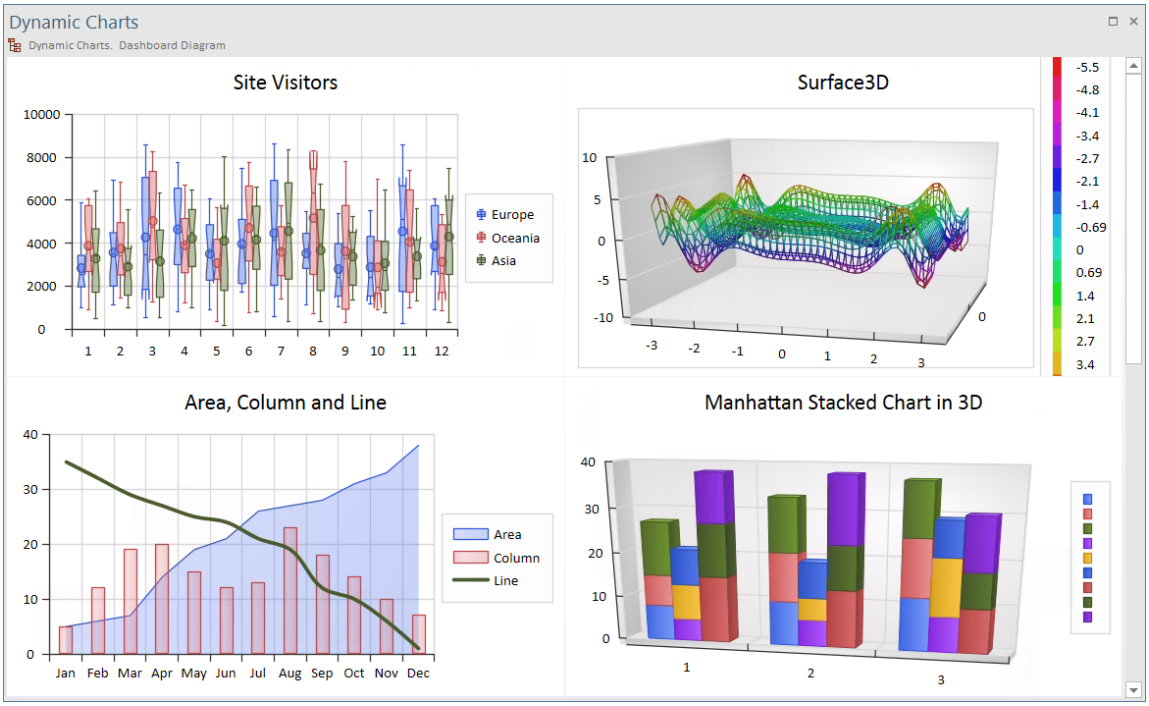
Author: Sparx Systems
Date: 13/11/2024
Version: 17.0

创建于 ENTERPRISE ARCHITECT

目录

动态图表

Enterprise Architect特征可以在图表打开时动态工件它，它是图表元素的样式和渲染。动态图表完全依赖代码来定义其系列、样式和内容，并且完全由客户端通过自动化接口进行管理 - 通常是插件和脚本。图表接口为客户提供了在查看图表时动态描述和填充图表的方法。该 API 范围广泛且灵活，允许您在运行时以图形方式说明许多不同的场景。特征允许您使用JavaScript作为代码或 JSON 定义您需要的任何类型的图表。



有关图表界面及其使用的A很好的参考是Enterprise Architect示例模型中的 报告 > 图表 > 动态图表”包。这个包包含许多图表的示例，每个图表都由JavaScript客户端动态描述。每个图表示例都展示了两种渲染方法 - JavaScript编码方法和 JSON 数据源方法。

动态图表对于表示模拟结果特别有用，允许您：

- 将仿真结果保存为可视图表元素
- 在您的报告中轻松包含由仿真结果填充的图表
- 无需任何额外的仿真工具即可与利益相关者分享用户友好的仿真结果

企业统一版和Enterprise Architect终极版中提供动态图表。

使用 JSON图表定义

与其自己编写动态图表，不如提供一个简单的图表描述。动态图表可以由单个数据源设计和完全定义； JSON 目前是首选的数据源格式，但将来会提供 XML 和其他格式。

您可以通过提供一个遵循 DynamicChart 模式的简单 JSON 数据结构来定义图表。该模式在Schema 编辑器中可用。它也可以在Enterprise Architect示例模型的 Dynamic Charts包中轻松查看。

（要查看 DynamicChart架构，请选择开发 >架构建模> Schema 编辑器> 打开Schema 编辑器，单击 配置文件”字段中的...按钮，然后选择 DynamicChartSchema。 ）

数据源 - JSON

要渲染图表数据结构，首先选择工件元素，然后打开内部代码编辑器对于浏览器中的选择，右键单击并选择 特征>编辑内部代码”，或者对于图表上的选择，右键单击并选择 脚本图表。这些将打开编辑器供您编辑图表脚本。创建一个 JSON 变量来定义要呈现的图表，然后编写您的 ConstructChart函数。

ConstructChart函数将显示在开始图表上的图表元素的标识（一个GUID string ）作为其单个参数。然后调用内置函数ConstructChartFromJSON，将GUID作为第一个参数，将 JSON 结构作为第二个参数传递，如下例所示：

Example Datasource in JSON

```
var barChart2DJSON =
{
 "Category" : "BarSmart",
 "Type" : "Simple",
 "Title" : "Vehicle Expenses",
 "Series" :
 [
   {
                "Label" : "Fuel",
                "Data":
                {
                        "Type" : "Column",
                        "Points" :
                        [
                                { "Category": "Jan", "Y": 1.0 },
                                { "Category": "Feb", "Y": 3.0 },
                                { "Category": "Mar", "Y": 7.0 },
                                { "Category": "Apr", "Y": 8.0 },
                                { "Category": "May", "Y": 10.0 },
                                { "Category": "Jun", "Y": 15.0 }
                        ]
                }
        },
```

```
{
        "Label" : "Taxes",
        "Data":
        {
                "Type" : "Normal",
                "Points" :
                [
                        { "Y":10.0 },
                        { "Y":12.0 },
                        { "Y":16.0 },
                        { "Y":17.0 },
                        { "Y":10.0 },
                        { "Y":12.0 }
                ]
        }
},
{
        "Label" : "Maintenance",
        "Data":
        {
                "Type" : "Normal",
                "Points" :
                [
                        { "Y":5.0 },
                        { "Y":2.0 },
                        { "Y":6.0 },
                        { "Y":7.0 },
                        { "Y":1.0 },
                        { "Y":2.0 }
                ]
        }
},
{
        "Label" : "Other",
        "Data":
        {
                "Type" : "Normal",
                "Points" :
                [
                        { "Y":2.5 },
                        { "Y":2.5 },
                        { "Y":2.5 },
```

```
                                    { "Y":2.5 },
                                    { "Y":2.5 },
                                    { "Y":2.5 }
                                ]
                        }
                }
        ]
};

function ConstructChart(chartGuid)
{
        ConstructChartFromJSON(chartGuid, barChart2DJSON);
}
```

进一步的例子

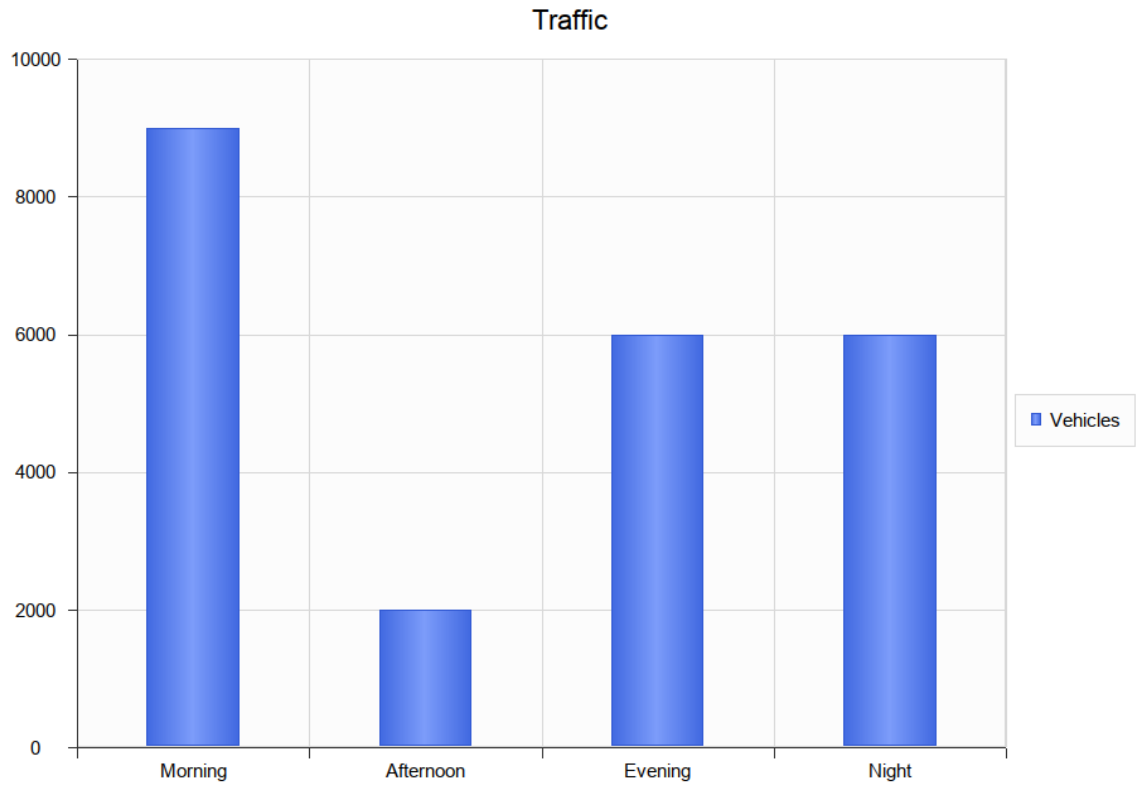
示例模型中提供了更多示例示例（参见包 报告 > 图表 > 动态图表”）。

每个图表示例都提供了一个仪表板图和 DynamicChart元素。从这些示例中选择一个元素，然后按 Alt+7”查看图表背后的行为。查看各种图表示例是了解如何使用 JSON 生成您可能感兴趣的图表类型的最佳方式。

使用仿真图表定义

模拟是观察行为的绝佳工具。在仿真中的任何时候，很容易分辨出我们在哪里以及我们所处的状态。当我们逐步进行仿真时，这些信息通常会被丢弃。例如，在向我们展示 24 小时内的交通流量的仿真中，我们可以很容易地观察到在早晚不同时间通过隧道的车辆数量。在仿真完成后保留此信息并使用它来提供有意义的东西可能会很有用。仿真中的动态图表特征使我们能够做到这一点。以上面的例子为例，我们可以记录仿真每一步的交通量，并用它制作一个图表，清楚地显示在仿真的 24 小时内通过隧道的交通量。图表实际上可以为我们显示仿真的时间线或仿真的总和效果。

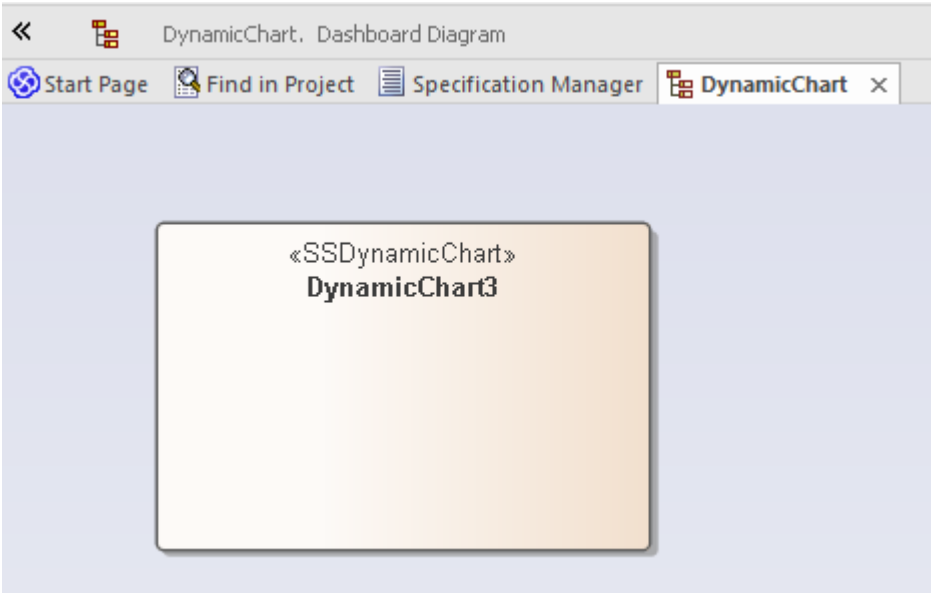
在仿真中制作自定义图表



You can fashion all sorts of Charts from any Simulation. Each time a Simulation is run, any DynamicChart elements referenced (by name) by the Simulation are updated. The Simulation will search for any named Chart in the same Package as the model.

Follow this simple process:

1. Create a DynamicChart element for the Simulation.



2. In the initial step of the Simulation, use JavaScript to define a variable to hold the vehicle numbers.

```
//
//  the traffic variable will hold the traffic numbers as Simulation proceeds and is initially zero
//  each element of the array represents a period of the day, morning, afternoon, evening and night.
//
var traffic = [0,0,0,0];
```

3. Next write the JavaScript that describes, in JSON format, the Chart to produce.

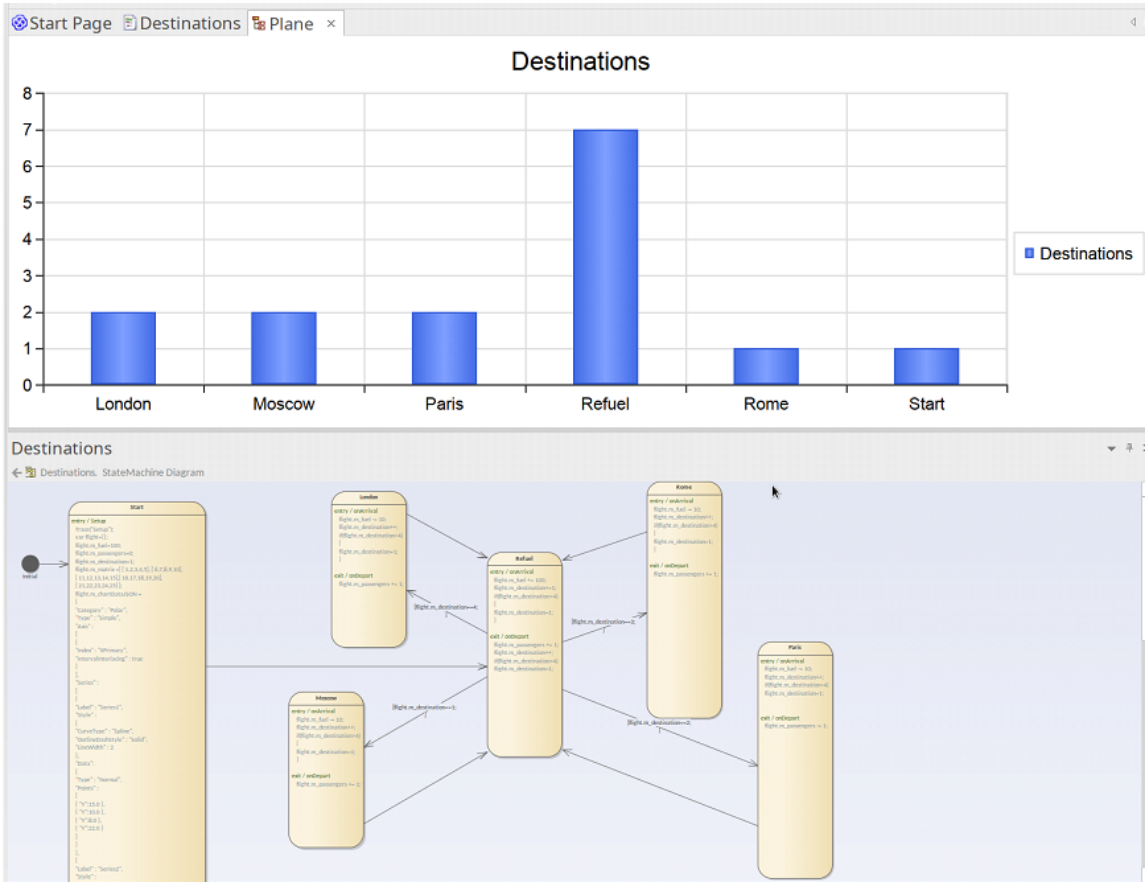
```
//
// The JSON instance describing the chart to produce. (complies with the EA DynamicChart Schema)
//
    var chartData =
    {
     "Category" : "Column",
      "Type" : "Simple",
      "Title" : "Traffic",
      "Series" :
      [
            {       "Label" : "Vehicles",
                    "Data" :
                    {
                          "Type" : "Column",
                          "Points" :
                          [
                                { "Category": "Morning", "Y" : 0 },//  The Y values of the axis are initially
zero
                                { "Category": "Afternoon", "Y" : 0 },        //  they will be filled in at end
of Simulation
                                { "Category": "Evening", "Y" : 0 },
                                { "Category": "Night", "Y" : 0 }
                          ]
                    }
            }
      ]
    };
```

4. At various transitions in the Simulation update the traffic numbers.

```
//
// 2000 vehicles went through the tunnel in the afternoon (element 1)
//
traffic[1] += 2000;
```

5. At the end of the Simulation, use the data captured during the run to fill the series.
```
// fill points in series with the number of vehicles for each part of the day
var dataPoints = chartData.Series[0].Data.Points;
for(var dp = 0; dp < traffic.length; dp++)
{
    dataPoints[dp].Y = traffic[dp];
}
```
6. Update the model.
```
// Call the EA function to populate the DynamicChart element named 'Vehicles' with this data.
sim.GenerateChart( "Vehicles", JSON.stringify(chartData));
```

默认图表由仿真制作



除了您专门生成的图表外，还可以通过模拟自动生成摘要图表。所有这一切都是在包中添加一个工件所需的，并为其状态机与包相同的名称。默认的图表总结了模拟执行过程中的状态转换。如果在模拟完成时找到默认的图表，该图表的数据将被更新并自动显示。

要将默认图表添加到您的状态机模拟，请按照下列步骤操作：

1. 找到包含要在其上执行模拟的状态机的包。
2. 创建一个仪表板图作为那个包的孩子。
3. 在仪表板图中添加一个工件状态机图，并赋予它与机器相同的名称。

模拟结束后，只需打开仪表板图即可显示摘要。

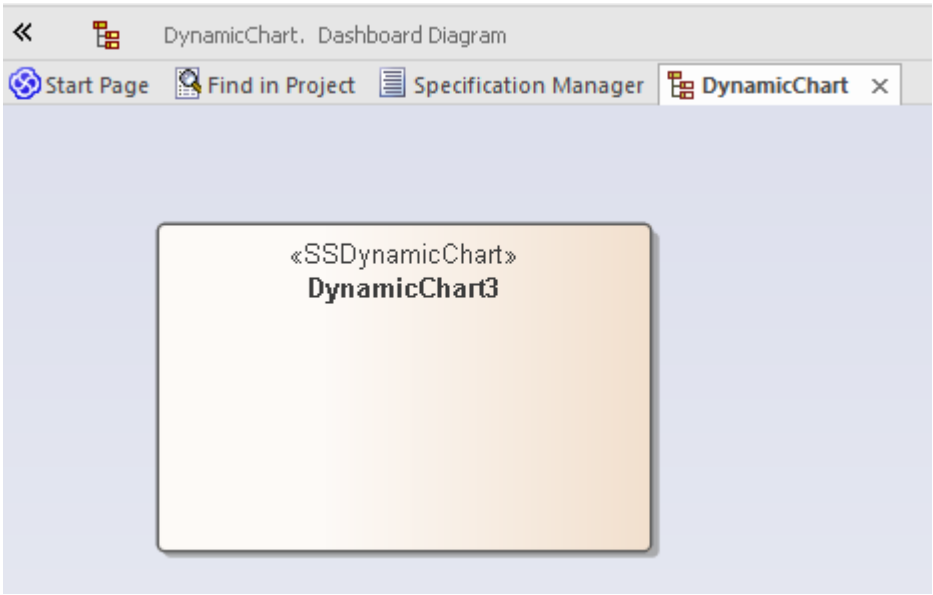
使用图表定义JavaScript

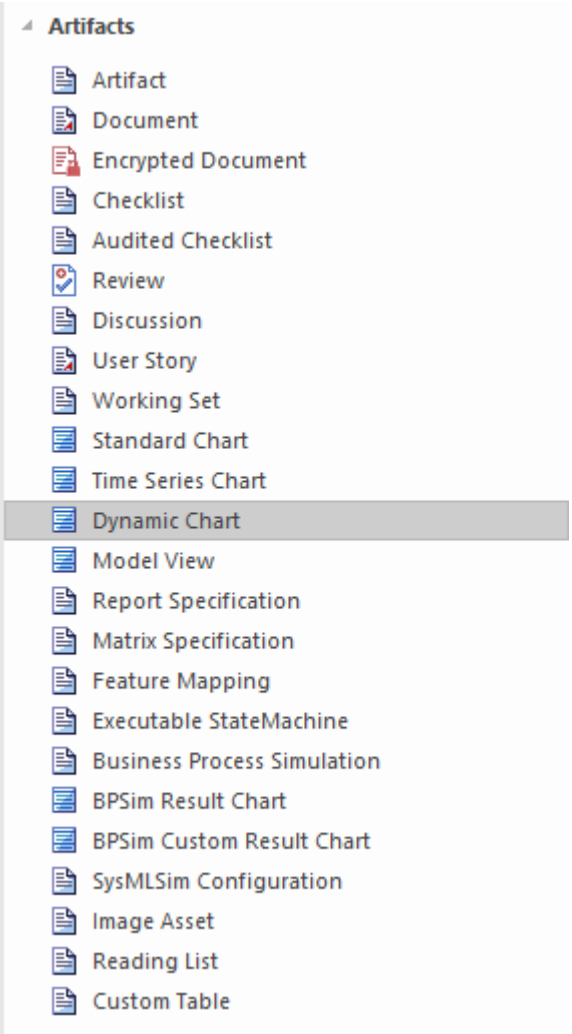
在这个主题中，我们只讨论使用JavaScript和图表界面的DynamicChar工件。

通过JavaScript定义图表

您要做的第一件事是在适当的包中创建一个仪表板图。右键单击包并选择 添加图表”选项。

在 新图表”对话框中选择 构造>图表和图表”类型，当显示空图表时，将 劭态图表”图标从工具箱的 图表”页面拖到它上面。





现在编写JavaScript来设置样式，从包含图表的工件函数开始调用，每当打开图形进行查看时，它就会自动呈现图表。元素的GUID作为参数传递给 ConstructChart。在这个函数中，显示什么类型的图表、图表的图表、它包含的系列数量以及组成系列的数据点完全取决于您。使用自动化接口中的图表包，几乎可以显示任何您需要的图表。

在此示例中，您将创建一个分组列图表，显示几个月的车辆费用。每个组将代表一个月，并将分解为该月发生的不同费用。

首先，点击工件并按 Alt+7，或点击 编辑图表脚本”上下文菜单选项；每个方法都显示代码编辑器窗口。此处提供了要使用的代码，随后是打开图表时将生成的图形。

重要的是，注记：

- **!INC 当地的 Scripts.ChartAutomation** 陈述;所有图表都必须包含这个声明
- ConstructChart函数（第 7 行）

代码

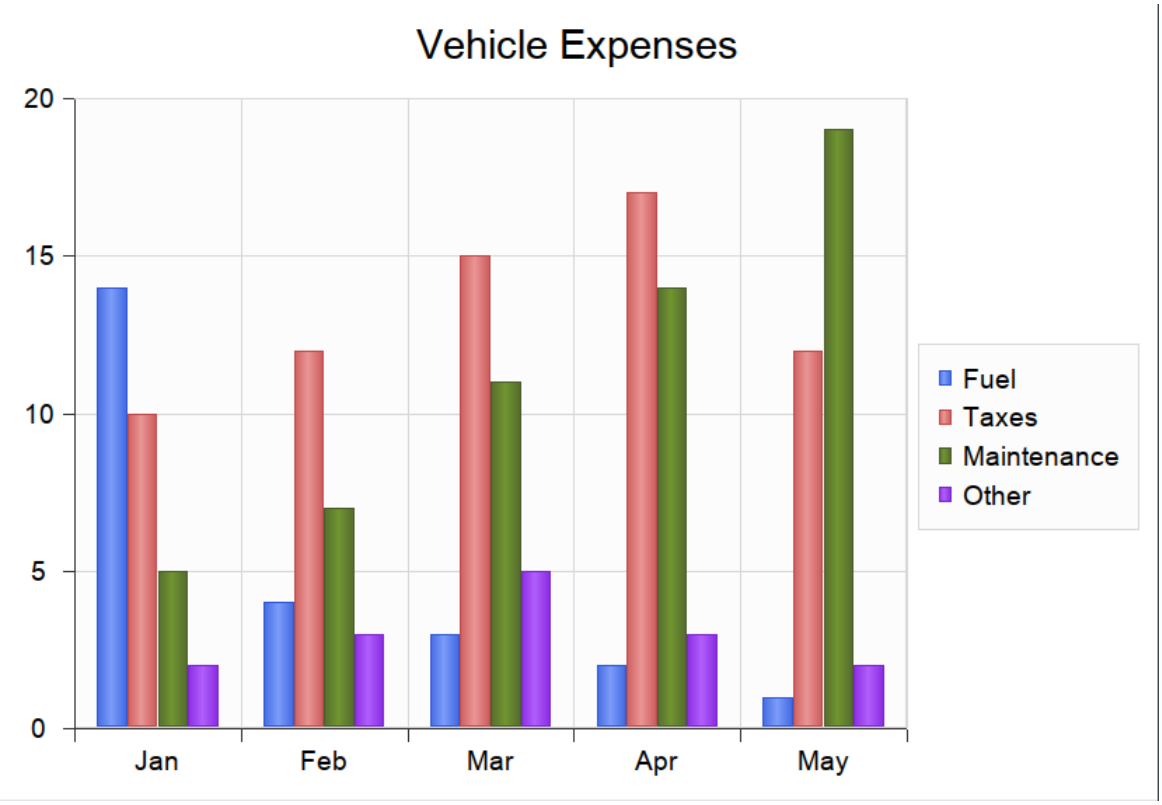
```
!INC Local Scripts.ChartAutomation

var monthNames = [ "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ];
```

```
function Rand(min, max) {
  min = cephes.ceil(min);
  max = cephes.floor(max);
  return cephes.floor(cephes.drand() * (max - min)) + min; }

function ConstructChart( guid )
{
        var chart as EA.Chart;                          // The script first of all
        var element = GetElementByGuid(guid);           // declares the automation
        var series1 as EA.ChartSeries;                  // objects it will use
        var series2 as EA.ChartSeries;
        var series3 as EA.ChartSeries;
        var series4 as EA.ChartSeries;

        chart = element.GetChart();

        var chartCategory = ChartCategory.Column();
        var chartType = ChartType.SIMPLE();
        chart.SetChartType( chartCategory, chartType, false, true);

        chart.Title = "Vehicle Expenses";
        series1 = chart.CreateSeries("Fuel");          // The script then obtains the Chart object and creates the
        series2 = chart.CreateSeries("Taxes");         // series. A chart is composed of a number of series, and
        series3 = chart.CreateSeries("Maintenance");   // in this example each series will represent a type of expense.
        series4 = chart.CreateSeries("Other");

        series1.AddDataPoint3( monthNames[0], 14);  // A series is composed of a number of datapoints and, here, the

        series1.AddDataPoint3( monthNames[1], 4);   // script adds the values for each of the points to each series.
        series1.AddDataPoint3( monthNames[2], 3);
        series1.AddDataPoint3( monthNames[3], 2);
        series1.AddDataPoint3( monthNames[4], 1);

        series2.AddDataPoint(10);
        series2.AddDataPoint(12);
        series2.AddDataPoint(15);
        series2.AddDataPoint(17);
        series2.AddDataPoint(12);

        series3.AddDataPoint(5);
        series3.AddDataPoint(7);
        series3.AddDataPoint(11);
```

```
        series3.AddDataPoint(14);
        series3.AddDataPoint(19);

        series4.AddDataPoint(2);
        series4.AddDataPoint(3);
        series4.AddDataPoint(5);
        series4.AddDataPoint(3);
        series4.AddDataPoint(2);

        series1.SetGroupID(0);
        series1.SetGroupID(0);
        series3.SetGroupID(1);
        series4.SetGroupID(1);

        chart.Redraw();
}
```

输出

这是代码生成的图表。



调试动态图表

创建 Dynamic图表JavaScript ，您可以像调试任何其他代码一样对其进行调试。右键单击图表中的动态图形并选择 调试图表脚本”选项。该脚本显示在调试视图中。

进一步的例子

示例模型中提供了进一步的编码示例（参见包 报告 > 图表 > 动态图表”）。每个图表示例都提供了一个仪表板图和一个 DynamicChart元素。选择这些元素中的任何一个并按 Alt+7 以查看图表背后的行为代码。参考这些示例是了解如何编写每种类型的图表的最佳方式。

JavaScript是使用自动化编码动态图表的主要语言。但是，第三方自动化客户端参与该过程当然是可行的，JavaScript主机将任务委派给C#和C++等语言的自动化客户端，可能能够从模型外部获取数据不适用于脚本。

动态图表资源

如资源表中所述，您可以A许多资源来帮助在Enterprise Architect中动态构建图表。

资源

您可以使用的资源包括此表中描述的功能。

资源	描述
JavaScript库	A JavaScript函数库，包括使用自动化接口或使用 JSON 作为 Charts 的数据源对 Charts 进行编码的示例。这个库可以在Enterprise Architect的脚本控件的“Local脚本”组中找到。
示例模型	Enterprise Architect示例模型包含许多图表示例，展示了可以在显示时动态创建和设置样式的各种类型的图表。这些在包'Reporting > Charts > Dynamic Charts'中。 示例包括 3D 曲面和线框图表、显示价格和数量波动的系列股票图表、显示访客数量的箱线图、曼哈顿堆栈图表等。 每个示例提供两个版本。第一个版本使用JavaScript代码来设计和填充图表。第二个使用 JSON 数据源来描述和填充图表。
图表模式	示例模型参考模型，包括动态图表， Schema 编辑器使用配置文件生成的 JSON 和 XML 模式，作为工件文件的类，您可以通过双击它们快速查看。

图表API

图表接口是提供动态创建图表的方法的 API object 。它可用于构造任何受支持的图表类型。

Figure 接口是使用 DynamicChart元素上A图表方法获得的。图表元素可以从图表工具箱A 图表"页面创建，通常用于仪表板图。

图表类

The Chart Class is the primary interface for Chart elements; it is used to create a series, add datapoints to a series and configure the chart appearance.

Chart Attributes

Attribute	Description
Title	String Notes: Read/Write The title of the chart.
Category	ChartCategory Notes: Read only The chart category; provided in the SetChartType method.
Type	ChartType Notes: Read only The chart type; provided in the SetChartType method.

Chart Methods

Method	Description
AddChartDataYXZ(double Y, double X, double Z, long seriesIndex)	long Adds a datapoint to an existing series. Parameters: • Y: double, the primary Y axis value • X: double, the primary X axis value • Z: double, the primary Z axis value • seriesIndex: long, the index of the series (returned by the CreateSeries methods)
AddChartDataYY1(string category, double Y, double Y1, long seriesIndex)	long Adds a datapoint to an existing series. Parameters: • category: string - the x axis group, column or label • Y: double, the primary Y axis value • Y1: double, the secondary Y axis value • seriesIndex: long, the index of the series (returned by the CreateSeries and CreateSeriesEx methods)

CreateSeries(string name)	LDISPATCH Adds a new series to the chart. Returns an interface that can be used to add data to the series and configure its appearance. Parameters: • name: string, the displayed name of the series
CreateSeriesEx(string name, long color, ChartType type, ChartCategory category)	LDISPATCH Creates a series of a particular chart category and type and returns an IChartSeries interface. This allows charts to form multiple series in various ways. The CombinedCharts in the EAExample Model are an example of this, displaying three series for the Area, Column and Line categories respectively. Parameters: • name: string, the name of the series • color: long, RGB color value,-1 for default • type: ChartType, one of the ChartType enumerations • category: ChartCategory, one of the ChartCategory enumerations
EnableResizeAxes(boolean bEnable)	void Grants or denies the ability to resize axes. Parameters: • bEnable: boolean
GetChartAxis(ChartAxisType type)	LDISPATCH Returns an IChartAxis interface for the specified axis. Parameters: • type: ChartAxisType, one of the ChartAxisType enumerations
GetDiagram3D()	LDISPATCH Returns an IChartDiagram interface that can be used to specify the rendering engine; Software or OpenGL.
GetSeries(long index)	LDISPATCH Returns an IChartSeries interface for the given index. Indices for series begin at zero. Parameters: • index: long
GetSeriesCount()	long Returns the number of series represented by the chart.
Redraw()	void Redraws the chart.
SetChartType(ChartType type, ChartCategory category, boolean bRedraw, boolean bResizeAxis)	void This is typically the first call made on the Chart interface. It defines the style and appearance of the Chart when it is rendered. Parameters: • type: ChartType

	• category: ChartCategory • bRedraw: Boolean • bResizeAxis: Boolean
SetCurveType(ChartCurveType type)	void Sets the interpolation method of drawing the curve. Parameters: • type: ChartCurveType, one of the ChartCurveType enumerations, such as Line, Spline or SplineHermite.
SetSeriesShadow(boolean bShow)	void Displays or hides shadows on series. Parameters: • bShow: Boolean
SetThemeOpacity(long percentage)	void Sets the opacity of the chart. Parameters: • percentage: long, chart transparency as a percentage
ShowAxis(long index)	void Shows the axis for the given index. Parameters: • index: long, one of the ChartAxisType enumerations
ShowDataLabels(boolean show, boolean border, boolean dropLineTomarker)	void Shows or hides data labels on the chart. Parameters: • show: Boolean, show or hide labels • border: Boolean, show or hide border on labels • dropLineTomarker: Boolean, changes position of label with respect to line
ShowDataMarkers(boolean show, long size, ChartmarkerShape shape)	void Shows or hides data markers on the chart. Also allows setting the appearance of markers. Parameters: • show: Boolean, show or hide markers • size: long, size of markers in pixels • shape: ChartmarkerShape, one of the ChartmarkerShape enumerations

图表枚举

These enumerations, used specifically by methods in the Chart interface, are described in the topics of this section. Click on the enumeration name in the list to the left of this text.

ChartAxisCrossType

Enum Values

Enum	Value
Auto	value: 0
MaximumAxisValue	value: 1
MinimumAxisValue	value: 2
AxisValue	value : 3
Ignore	value: 4
FixedDefaultPos	value: 5

ChartAxisIndex

Enum Values

Enum	Value
Unknown	value: -1
X	value: 0
Y	value: 1
Z	value: 2

ChartAxisLabelType

Enum Values

Enum	Value
NoLabels	value: 0
NextToAxis	value: 1
High	value: 2
Low	value: 3

ChartAxisTickMarkType

Enum Values

Enum	Value
NoTicks	value: 0
Inside	value: 1
Outside	value: 2
Cross	value: 3

ChartAxisType

A set of constants that refer to the various axes used in charts.

Enum Values

Enum	Value
CHART_Y_PRIMARY_AXIS	value: 0
CHART_Y_SECONDARY_AXIS	value: 1
CHART_X_PRIMARY_AXIS	value: 2
CHART_X_SECONDARY_AXIS	value: 3
CHART_Z_PRIMARY_AXIS	value: 4
CHART_Z_SECONDARY_AXIS	value: 5
CHART_Y_POLAR_AXIS	value: 6
CHART_X_POLAR_AXIS	value: 7
CHART_A_TERNARY_AXIS	value: 8
CHART_B_TERNARY_AXIS	value: 9
CHART_C_TERNARY_AXIS	value: 10

ChartBarShape

Enum Values

Enum	Value
Box	value: 0
Pyramid	value: 1
PyramidPartial	value: 2

ChartCategory

Enum Values

Enum	Value
chartDefault	value: 0
chartLine	value: 1
chartPie	value: 2
chartPie3D	value: 3
chartPyramid	value: 4
chartPyramid3D	value: 5
chartFunnel	value: 6
chartFunnel3D	value: 7
chartColumn	value: 8
chartBar	value: 9
chartHistogram	value: 10
chartArea	value: 11
chartStock	value: 12
chartBubble	value: 13
chartLongData	value: 14
chartHistoricalLine	value: 15
chartPolar	value: 16
chartDoughnut	value: 17
chartDoughnut3D	value: 18
chartTorus3D	value: 19
chartTernary	value: 20
chartColumn3D	value: 21

chartBar3D	value: 22
chartLine3D	value: 23
chartArea3D	value: 24
chartSurface3D	value: 25
chartDoughnutNested	value: 26
chartBoxPlot	value: 27
chartBarSmart	value: 28
chartBar3DSmart	value: 29

ChartColorMode

Enum Values

Enum	Values
Single	value: 0
Multiple	value: 1
Palette	value: 2
Custom	value: 3
Series	value: 4

ChartCurveType

Enum Values

Enum	Value
NoLine	value: 0
Line	value: 1
Spline	value: 2
SplineHermite	value: 3
Step	value: 4
ReversedStep	value: 5

ChartDashStyle

Enum Values

Enum	Value
Solid	value: 0
Dash	value: 1
Dot	value: 2
DashDot	value: 3
DashDotDot	value: 4
Custom	value: 5

ChartFrameStyle

Enum Values

Enum	Value
None	value: 0
Mesh	value: 1
Contour	value: 2
ContourMesh	value: 3

ChartGradientType

Enum Values

Enum	Value
None	value: 0
Horizontal	value: 1
Vertical	value: 2
DiagonalLeft	value: 3
DiagonalRight	value: 4
CenterHorizontal	value: 5
CenterVertical	value: 6
RadialTop	value: 7
RadialCenter	value: 8
RadialBottom	value: 9
RadialLeft	value: 10
RadialRight	value: 11
RadialTopLeft	value: 12
RadialTopRight	value: 13
RadialBottomLeft	value: 14
RadialBottomRight	value: 15
Bevel	value: 16
PipeVertical	value: 17
PipeHorizontal	value : 18

ChartMarkerShape

Enum Values

Enum	Value
Circle	value: 0
Triangle	value: 1
Rectangle	value: 2
Rhombus	value: 3

ChartStockSeriesType

Enum Values

Enum	Value
Bar	value: 0
Candle	value: 1
LineOpen	value: 2
LineHigh	value: 3
LineLow	value: 4
LineClose	value: 5
LineCustom	value: 6

ChartType

Enum Values

Enum	Value
chartTypeDEFAULT	value: 0
chartTypeSIMPLE	value: 1
chartTypeSTACKED	value: 2
chartType100STACKED	value: 3
chartTypeRANGE	value: 4

ChartWallOptions

Enum Values

Enum	Value
None	value: 0, 0x0000
FillLeftWall	value: 1, 0x0001
OutlineLeftWall	value: 2, 0x0002
FillRightWall	value: 4, 0x0004
OutlineRightWall	value: 8, 0x0008
FillFloor	value: 16, 0x0010
OutlineFloor	value: 32, 0x0020
DrawAll	value: 65535, 0xFFFF
DrawLeftWall	FillLeftWall | OutlineLeftWall
DrawRightWall	FillRightWall | OutlineRightWall
DrawFloor	FillFloor | OutlineFloor
DrawAllWalls	DrawLeftWall | DrawRightWall
OutlineAllWalls	OutlineLeftWall | OutlineRightWall
OutlineAll	OutlineAllWalls | OutlineFloor
FillAllWalls	FillLeftWall | FillRightWall
FillAll	FillAllWalls | FillFloor
Default	OutlineAll

ChartAxisIndex类

ChartAxisIndex Attributes

Attribute	Description
Visible	Boolean Shows or hides the axis.

ChartAxisIndex Methods

Method	Description
EnableMajorUnitIntervalInterlacing(boolean binterlace)	void Turns interlacing on or off.
GetGuid()	string Returns the guid of the axis. Uniquely identifies an axis.
GetLabel()	string Returns the value of the label of the axis.
SetAxisName(string label, boolean showonaxis)	void Sets the label for the axis and whether it should be displayed on the chart. Parameters: • label: string, the text for the label • showonaxis: Boolean, a true value indicates that the label is displayed
SetCrossType(long type)	void Provides a directive or hint for use when calculating the position of labels on an axis. Parameters: • type: long, one of the ChartAxisCrossType enumerations
SetDataFormat(string format, boolean formatAsDate)	void Sets the format string for the conversion of values to strings (e.g. "%.4f"). If the datapoints represent datetime values, the formatAsDate argument should be true, and the format string set appropriately (e.g. "%H:%M") Parameters: • format: string, the format to use when converting datapoint values to string • formatAsDate: Boolean, a true value indicates the datapoint represent a datetime
SetDisplayUnits(double	

units)	void Sets the display units on the axis. Basically, the datapoint values are divided by this figure to give a major unit value. For example, if the datapoint contains meter values, a value of 1000 would result in kilometers being used as the major unit on the axis. Parameters: • units: double, the value of a single unit on the axis
SetFixedDisplayRange(double fmin, double fmax)	void Sets a fixed range for the axis. Parameters: • fmin: double, the minimum value • fmax: double, the maximum value
SetLabelType(long labelpos)	void Sets the position of labels on the axis. Parameters: • labelpos: long, one of the ChartAxisLabelType enumerations
SetTickMark(long tickmarkpos)	void Sets the position of tick marks on the axis. Parameters: • tickmarkpos: long, one of the ChartAxisTickMarkType enumerations
ShowMajorGridLines(boolean show)	void Shows or hides grid lines.

ChartDataValue类

The ChartDataValue class provides an interface that allows values to be obtained from points in a series.

ChartDataValue Methods

Method	Description
GetValue()	double Returns the value associated with the datapoint.
IsEmpty()	Boolean True if no value exists for the datapoint.
SetEmpty(boolean empty)	void Sets a datapoint on a series to be empty. Parameters: • empty: Boolean, true if the datapoint is to be considered as empty, having no value
SetValue(double value)	void Sets the value of a datapoint. Parameters: • value: double, the value of the datapoint; setting a value makes a datapoint non-empty

ChartDiagram3D类

ChartDiagram3D Methods

Method	Description
SetDrawWallOptions(long options, boolean redraw)	void Sets the option for how walls and floors - if any - are displayed on the 3D chart. The options parameter is a bitmask of one or more values from the ChartWallOptions enum. Parameters: • options: Long, bitmask of wall and floor display options • redraw: Boolean, redraws the chart after the function completes
SetRenderingType(long engine)	void Parameters: • engine: long, 0 for software,1 for openGL

ChartFormatSeries类

A helper class for the ChartSeries class that allows setting appearance options.

ChartFormatSeries Methods

Method	Description
SetCurveType(ChartCurveType type)	void Sets the graphic option for rendering lines. Parameters: • type: long, one of the ChartCurveType enumerations
SetSeriesLineWidth(long width)	void Sets the line width in pixels. Parameters: • width: long, a pixel value
SetSeriesOutlineDashStyle(ChartDashStyle dashstyle)	void Sets the dash style of the line on the chart/graph. Parameters: • dashstyle: ChartDashStyle, one of the ChartDashStyle enumerations

ChartSeries类

ChartSeries Methods

Method	Description
AddBoxPlotData(double ave, double min, double q1, double q2, double q3, double max, double notched)	long For a chart having the BoxPlot category, adds a single datapoint to the series. Parameters: • ave: double, the mean value at this point • min: double, the minimum value at this point • q1: double, the first quartile value • q2: double, the second quartile value • q3: double, the third quartile value • max: double the maximum value at this point • notched: double, for a series with notched style, the notched value to express at this point
AddDataPoint(double Y)	long Adds a datapoint to the series. Returns the index of the point, which is the number of points -1. Parameters: • Y: double, the Y axis value
AddDataPoint2(double Y, double X)	long Adds a datapoint to the series. Returns the index of the point, which is the number of points -1. Parameters: • Y: double, the Y axis value • X: double, the X axis value
AddDataPoint3(string category, double Y)	long Adds a Y axis value for a given category on the X axis. Parameters: • category: string, the category or column name • Y: double, the value
AddStockData(double open, double high, double low, double closing, VARIANT timestamp)	void Adds data to a series for a chart of the Stock category. Parameters: • open: double, opening value • high: double, high value • low: double, low value • closing: double, closing value • timestamp: {datetime, double utcsecs} either VARIANT date value or double, in which case the value is interpreted as the number of seconds since midnight

	on January 1st, 1970, UTC time
AddSurfaceColors（变种颜色）	void Adds one or more colors to the series. Parameters: • colors: long, or long[], a single RGB color or an array of RGB color values
CloseShape(boolean close, boolean fill)	void Connects the first and last datapoints and fills the shape if 'fill' is true. Parameters • close: Boolean, if true closes the series • fill: Boolean, fills the shape
GetDataPointCount()	long Returns the number of datapoints in the series.
GetDataPointValue(long index)	LDISPATCH Returns a ChartDataValue interface for the datapoint with the given index. Parameters: • index: long, the index of the datapoint (typically returned by AddDataPoint functions; a value in the range 0 to n-1, where n is the number of points returned by the *GetDataPointCount* function)
GetSeriesFormat()	LDISPATCH Returns a ChartFormatSeries interface that allows the chart appearance to be changed.
SetBarShape(long barshape)	void Sets the shape for Bar charts, 0 for Box, 1 for Pyramid, 2 for PyramidPartial. Parameters: • barshape: ChartBarShape, one of the ChartBarShape enumerations
SetColorMapCount(long count)	void Sets the number of colors used when rendering the series. Typical values are 4, 8, 16 and 32 Parameters: • count: long, the number of colors to use
SetColorMode(ChartColorMode mode)	void For 3D charts, sets the interpolation method for filling shapes. Single, for example, would result in the 3D object being filled by varying the color slightly. The level of variation will depend on the number of colors used by the chart (see *SetColorMapCount*). Parameters: • mode: ChartColorMode
SetDrawFlat(boolean flat)	void Draws the shape flattened when set to true. Parameters:

	• flat: Boolean, draw flat
SetFrameColor(long color)	void Sets the color of the frame for 3D objects. Parameters: • color: long, the RGB color value for coloring the frame
SetFrameStyle(ChartFrameStyle style)	void Sets the frame style for the chart - none, mesh, contour or both. Parameters: • style: ChartFrameStyle, one of the ChartFrameStyle enumerations
SetGradientType(long type)	void Sets the gradient type to use. Parameters: • type: long, one of the ChartGradientType enumerations
SetGroupID(long id)	void Groups series on a stacked chart having the same id. Must be a non-negative number. Parameters: • id: long, a non-negative number used to group the series on a chart
SetLevelRangeMode(long mode)	void Sets the mode for ranges in series. • 0 - Minimum and maximum for Series • 1 - Minimum and maximum for Y axis • 2 - Custom Parameters: • mode: long, either 0 or 1 supported
SetRelatedAxis(string axis, long index)	void Sets the related axis for a series. The related axis is created using the Split function of the ChartAxis interface. The axis is first created using Split, then a new series is created, and this function called on it to one of its axes. The axis is specified by the index parameter; the value is one of the ChartAxisIndex enumerations (0 for X, 1 for Y or 2 for Z) Parameters: • axis: string, the guid of the axis returned by a ChartAxis.Split method call; returned by the ChartAxis.GetGuid method • index: long, one of the ChartAxisIndex enumerations
SetStockSeriesType(ChartStockSeriesType type)	void For Stock charts, sets the graphic used to render the series. Parameters: • type: ChartStockSeriesType, one of the ChartStockSeriesType enumerations
SetWireFrame(boolean	void

wired)	Sets the wireframe option on or off. When set to true, the chart is no longer rendered as a solid object but is instead rendered as a frame composed of wires. Parameters: • wired: Boolean, displays as a wireframe object if true