INTRODUCTION

Chart Guru is a professional 2D charting package for Unity. It helps you build bar charts, line charts, area charts, scatter and bubble plots, pie and donut charts, radar charts, heatmaps, candlestick charts, sparklines, mini charts, and single-value indicators for runtime UI, editor tools, dashboards, HUDs, and data-heavy workflows.

The package is designed around a Swift Charts-inspired model: you describe data as marks, configure axes, titles, scales, legends, and themes, then render the chart through a Unity host such as ChartGraphic, ChartComponent, ChartGuruElement, or the Chart Designer preview.

Version 1.1.0 targets Unity 2022.2 or newer and uses Burst, Collections, Mathematics, Newtonsoft JSON, and TextMeshPro.

Contents

CHART LANGUAGE USED IN THIS GUIDE

A chart has a few named regions and building blocks. Knowing these terms makes the Inspector and API much easier to understand.

Term	Meaning
Chart area	The whole rectangle reserved for the chart, including title, subtitle, plot, axes, legend, labels, background, and border.
Plot area	The inner data region where bars, lines, points, areas, heatmap cells, and reference rules are drawn.
Mark	One visual data item or part of a series. Examples: BarMark, LineMark, AreaMark, PointMark, SectorMark, RectangleMark, RuleMark, RadarMark, and CandleStickMark.
Axis	A value or category guide. Cartesian charts normally use an X axis and a Y axis. Pie, donut, radar, and compact charts usually hide axes.
Scale and domain	The scale converts data values to positions. The domain is the range of data values shown, such as 0 to 100 or Jan to Dec.
Tick and grid line	A tick is a labeled value on an axis. A grid line extends a tick across the plot area to help compare values.
Series	A group of related marks, such as Sales and Expenses. Series keys drive grouping, stacking, colors, and legend entries.
Legend	The label list that explains series colors, symbols, or categories.
Title and subtitle	Text attached to the chart area, typically used for the main message and context such as period, units, or filter.
Data label and annotation	A data label shows a value near a mark. An annotation is richer explanatory text attached to a mark or reference line.

Beginner rule of thumb: choose bars for comparing categories, lines or areas for trends, points for relationships, pie or donut charts for simple part-to-whole stories, heatmaps for matrix intensity, and candlesticks for OHLC financial data.

INSTALLATION AND QUICK START

REQUIREMENTS

- Unity 2022.2 or newer.
- Burst 1.8.0 or newer, Collections 1.2.4 or newer, Mathematics 1.2.0 or newer, Newtonsoft JSON 3.2.2, and TextMeshPro 3.0.0 or newer. These are declared in the package manifest.
- A Canvas is recommended for the most common runtime UI path. UI Toolkit is also supported through *ChartGuruElement*.

INSTALL THE PACKAGE

1. Import Chart Guru from the Unity Asset Store.
2. Let Unity resolve the dependencies declared in package.json.
3. Open Tools > Chart Guru > Chart Gallery to confirm the package is installed and to browse live examples.
4. Open Tools > Chart Guru > Create Simple Chart when you want Chart Guru to create a sample Canvas and chart GameObject for you.

FASTEST FIRST CHART

5. Create a Canvas and add an empty child GameObject.
6. Add ChartGuru > Chart Graphic to the child GameObject.
7. Add ChartGuru > Data Sources > Mock Chart Data Source to the same GameObject.
8. Choose a Pattern such as TrendingUp, set Series Count and Points Per Series, and click Generate if needed.
9. Change Chart Type, theme, axes, labels, and legend settings on ChartGraphic. The chart updates in Edit Mode and Play Mode.

BUILD YOUR FIRST CHART

INSPECTOR PATH: NO CODE

Use the Inspector path when you want a fast dashboard, HUD, prototype, or designer-owned chart. ChartGraphic owns rendering and chart settings. A data source component feeds marks into it.

- Use *MockChartDataSource* for instant generated data.
- Use *ManualChartDataSource* when you want to type values directly into the Inspector.
- Use file or web data sources when a chart should follow CSV, JSON, Google Sheets, HTTP, Yahoo Finance, or high-throughput live samples.

CODE PATH: FULL CONTROL

Use the code path when chart data is produced by gameplay, simulation, analytics, editor tooling, or custom runtime systems.

```
using ChartGuru;
using UnityEngine;

public sealed class MonthlySalesChart : MonoBehaviour
{
    private ChartGraphic _graphic;

    private void Start()
    {
        float[] values = { 65f, 89f, 45f, 78f, 92f, 55f };
        string[] labels = { "Jan", "Feb", "Mar", "Apr", "May", "Jun" };

        ChartBuilder builder = Chart.Create();
        for (int i = 0; i < values.Length; i++)
        {
            BarMark bar = new BarMark(i, values[i]).cornerRadius(8f);
            bar.CategoryLabel = labels[i];
            bar.foregroundStyle(PlottableValue.value("Month", labels[i]));
            builder.Add(bar);
        }

        Chart chart = builder
            .chartTitle("Monthly Sales")
            .chartSubtitle("Current fiscal year")
            .chartXAxis(x => x.Label("Month"))
            .chartYAxis(y => y.Label("Sales").TickFormat("F0"))
            .chartLegend(LegendPosition.Hidden)
            .Animate(AnimationStyle.Staggered)
            .Build();

        _graphic = ChartCanvasHelper.RenderToRectTransform(
            chart, GetComponent<RectTransform>(), true, "Monthly Sales");
    }

    private void OnDestroy()
    {
        ChartCanvasHelper.RemoveChart(_graphic);
    }
}
```

HORIZONTAL BARS

Horizontal bars are not a separate ChartType. They are regular BarMark data with horizontal orientation. In the Inspector, use Bar Orientation. In code, set the mark layout to BarLayout.Horizontal.

```
BarMark bar = new BarMark(value, categoryIndex)
    .layout(BarLayout.Horizontal)
    .cornerRadius(6f);
bar.CategoryLabel = "Long category name";
```

CORE CONCEPTS

MARKS

A mark is the smallest drawable unit in a chart. The chart type often follows the mark type, but Chart Guru can also compose compatible cartesian marks in one chart, such as bars plus rule marks or points plus a line.

Mark	Used for
BarMark	Vertical or horizontal bars, ranges, grouped bars, stacked bars, and mini bars.
LineMark	Trend lines and connected point series.
AreaMark	Filled trends, stacked areas, range bands, and confidence bands.
PointMark	Point plots, scatter plots, and bubble-like encodings when symbol size is mapped.
SectorMark	Pie and donut slices.
RectangleMark	Heatmap cells and rectangular ranges.
RuleMark	Horizontal or vertical reference lines and threshold markers.
RadarMark	Radar or spider chart spokes and filled shapes.
CandleStickMark	Open, high, low, close financial candles.

BUILDER

The builder is a fluent way to assemble a Chart model. You add marks, configure the chart, then call Build. The same Chart model can be rendered in Canvas, UI Toolkit, editor previews, or custom code.

```
Chart chart = Chart.Create()
    .Add(new LineMark(0f, 42f))
    .Add(new LineMark(1f, 48f))
    .chartTitle("Trend")
    .chartXAxis(x => x.Label("Time"))
    .chartYAxis(y => y.Label("Value"))
    .Build();
```

SEMANTIC ENCODING

Semantic encoding means you describe what a mark represents instead of hard-coding every color. Chart Guru can then assign consistent colors, create legend entries, and group marks into series.

```
bar.foregroundStyle(PlottableValue.value("Product", "Electronics"));

// Use explicit color only when you need direct control.
bar.foregroundStyle(Color.red);
```

SERIES, STACKING, AND LEGENDS

Marks with the same SeriesKey belong together. ForegroundStyleKey controls color assignment. For multi-series charts, use semantic foreground style values or set SeriesKey and ForegroundStyleKey yourself. Stacking is controlled with MarkStackingMethod: Unstacked, Standard, Normalized, or Center.

CHART TYPES AND WHEN TO USE THEM

Chart type	Best use	Notes
Bar	Compare values across categories.	Use BarLayout.Horizontal for long category names. Supports rounded corners, ranges, grouping, and stacking.
Line	Show change over time or ordered X values.	Supports Linear, Monotone, CatmullRom, Cardinal, Natural, and Step interpolation modes.
Area	Show trend plus magnitude.	Can fill below a line or between ranges. Supports stacking and centered streamgraph style.
Point / Scatter / Bubble	Show individual observations or relationships.	Use symbols and symbol size scale for additional dimensions.
Pie / Donut	Show simple part-to-whole composition.	Use sparingly. Works best with a small number of slices and clear labels.
Radar	Compare several attributes per entity.	Useful for profiles, skill charts, and multi-axis comparison.
HeatMap	Show intensity across a two-dimensional grid.	Built from RectangleMark and color scales.
Candlestick	Show financial OHLC data.	Uses open, high, low, close columns or CandleStickMark values.
Sparkline / MiniBar / MiniPie / MiniDonut / Indicator	Embed compact values in HUDs, dashboards, tables, or inspector panels.	Compact types hide axes and reduce padding by default.

CHOOSING A CHART TYPE

- Use bar charts for ranking and category comparison.
- Use line charts for a trend where order matters.
- Use area charts when accumulated magnitude or range matters.
- Use scatter or bubble charts when both X and Y are measured values.
- Use heatmaps when a matrix pattern matters more than individual numbers.
- Use compact charts when the chart is a supporting signal, not the whole focus.

DATA SOURCES

Data sources are components that build marks and apply them to a target chart. They work in Edit Mode and Play Mode where appropriate, and most data source inspectors expose Refresh or Generate actions for quick iteration.

Source	Use it for
MockChartDataSource	Instant generated data for prototypes and demos. Patterns include Random, Linear, Exponential, Logarithmic, Sine, Cosine, TrendingUp, TrendingDown, Seasonal, Stepped, Sparse, RandomWalk, BellCurve, Sawtooth, and GaussianNoise.
ManualChartDataSource	Hand-entered labels, values, series, colors, and reference rules directly in the Inspector.
CsvChartDataSource	CSV, TSV, semicolon-separated files, TextAssets, StreamingAssets, inline text, or absolute paths.
JsonChartDataSource	JSON files or inline JSON, including nested data via root selectors.
GoogleSheetsChartDataSource	A shared Google Sheets URL converted to chart-ready tabular data.
WebChartDataSource	HTTP CSV or JSON endpoints with headers, polling, retry, and backoff.
YahooFinanceChartDataSource	Stock and financial time-series data for line or candlestick charts.
RingBufferChartDataSource	High-throughput live samples, appended efficiently and flushed once per frame.

MOCK DATA SOURCE

MockChartDataSource is the recommended starting point. Choose a pattern, label preset, point count, series count, value range, and noise factor. It can generate once or stream continuously with Constant, Random, Burst, or Drip rhythm.

```
MockChartDataSource mock = gameObject.AddComponent<MockChartDataSource>();
mock.Pattern = SeriesPattern.TrendingUp;
mock.SeriesCount = 3;
mock.PointsPerSeries = 12;
mock.LabelPreset = LabelPreset.MonthsShort;
mock.GenerateOnce();
mock.StartStreaming();
```

FILE AND WEB MAPPING

CSV, JSON, Sheets, Web, and Yahoo sources parse data into rows and columns, then use *ChartDataMapping* to decide which columns become X values, Y values, labels, series, and OHLC candles.

- XColumn becomes the X axis. If empty, row index is used.
- LabelColumn becomes the category or data label when present.
- YColumns are the plotted numeric values. WideFormat treats each Y column as its own series.
- SeriesByColumn pivots long-format data into multiple series.
- OpenColumn, HighColumn, LowColumn, and CloseColumn create candlestick marks.

- MaxRows limits how many parsed rows are consumed.

STYLING, LAYOUT, AND LABELS

THEMES

ChartTheme is a ScriptableObject that controls palette, text sizes, axis colors, grid colors, background, plot styling, shadows, glow, animation defaults, and text scaling. Assign a theme in the builder, ChartGraphic, ChartComponent, or ChartGuruElement.

```
Chart chart = Chart.Create()
    .theme(myTheme)
    .chartTitle("Revenue")
    .chartLegend(LegendPosition.Bottom)
    .Build();
```

CHART AREA AND PLOT AREA

Use the chart area settings for the outer background and border. Use plot area settings for the data region background, border, corner radius, and padding. This distinction matters when you want a card-like chart frame but a clean data plot inside it.

AXES AND SCALES

Axes can be automatic or explicitly configured. Use domains when the chart should keep a stable range. Use categorical or band axes for named categories, time axes for dates, logarithmic or symmetric-log scales for broad numeric ranges, and power scales when a non-linear transform is needed.

```
Chart chart = builder
    .chartXAxis(x => x.Label("Month").TickCount(6))
    .chartYAxis(y => y.Label("Revenue").Domain(0, 100).TickFormat("F0"))
    .chartYScale(0, 100)
    .Build();
```

TITLES, LABELS, AND ANNOTATIONS

Use the chart title for the main message and subtitle for context. Data labels show values on marks. Annotations attach explanatory text to a mark or reference line. RuleMark is the preferred way to add targets, thresholds, or baselines.

```
RuleMark target = RuleMark.Horizontal(100f).lineStyle(new[] { 8f, 4f });
target.foregroundStyle(Color.green);
target.annotation(AnnotationPosition.TopLeading, "Target");
builder.Add(target);
```

INTERACTION, ANIMATION, AND LIVE UPDATES

SELECTION AND GESTURES

ChartGraphic implements pointer, hover, drag, and scroll interfaces. The Chart model exposes selected point, selected X or Y value, selected angle, selected ranges, hovered index, and selection indicator styles such as Highlight, Lollipop, and RuleMark.

- Use point selection for dashboards and drilldown.
- Use range selection for time windows and brushing.
- Use angle selection for pie and donut slices.
- Use scrollable axes and visible domains when only part of a large data set should be visible.

ANIMATION AND MORPHING

Animations can run on initial display or data changes. Built-in styles include None, Fade, Grow, Staggered, StaggeredReverse, Simultaneous, FromCenter, Random, Spring, Slide, and Scale. Morphing can transition between compatible chart types at runtime.

```
chart.MorphTo(ChartType.Line, new MorphOptions { Duration = 0.6f });
chart.SetValues(newValues);      // fastest Y-only update when shape is unchanged
chart.SetData(newMarks);         // full replace when mark shape changes
```

RENDERING PATHS

All 2D rendering paths share the same Chart model and renderer. Choose the host that matches where the chart appears.

Host	Use it when
ChartGraphic	The chart belongs in a uGUI Canvas, including Screen Space Overlay, Screen Space Camera, or World Space Canvas.
ChartComponent	You want a quick scene component for prototyping without building a full Canvas workflow.
ChartGuruElement	The UI is built with UI Toolkit, UXML, USS, or Unity data binding.
Chart Designer / Chart Gallery	You need an editor preview or a visual authoring surface.

UI TOOLKIT

ChartGuruElement is a VisualElement with UXML attributes for chart type, title, labels, axes, legend, marks, data source, and interaction. Use UIToolkitChartBinder when a scene component should feed a named element inside a UIDocument.

```
<ui:UXML xmlns:cg="ChartGuru.UIToolkit">
    <cg:ChartGuruElement
        chart-type="Line"
        chart-title="Sales Trend"
        legend-position="Bottom"
        show-data-labels="true"
        animate-on-start="true" />
</ui:UXML>
```

EDITOR TOOLS

CHART GALLERY

Open Tools > Chart Guru > Chart Gallery. The gallery shows live preview cards for chart configurations and is the fastest way to discover suitable chart types.

CHART DESIGNER

Open Tools > Chart Guru > Chart Designer. The designer is a WYSIWYG authoring window with Data, Style, Axes, Legend, and Actions tabs. It can preview, theme, configure, and export C# code.

SWIFTUI CHART IMPORTER

The importer translates common Swift Charts snippets into Chart Guru builder code. It recognizes BarMark, LineMark, AreaMark, PointMark, SectorMark, RuleMark, semantic foreground style, symbols, symbol sizes, interpolation, scales, legends, and common ForEach patterns.

CREATE SIMPLE CHART AND UXML TEMPLATE

Use Tools > Chart Guru > Create Simple Chart to create a ready-to-run scene chart. Use Assets > Create > ChartGuru > UXML Chart Template to create a starter UI Toolkit chart template.

PERFORMANCE GUIDANCE

- Prefer SetValues when only Y values change and the mark count stays the same.
- Hide dense data labels on large charts. Text is pooled, but thousands of visible labels are still harder to read and render.
- Use compact charts for small dashboard signals instead of full axes and legends.
- Use RingBufferChartDataSource for frequent live samples instead of rebuilding a chart every event.
- Keep RenderTexture size close to the displayed chart size. Oversized charts cost memory and fill rate.
- Let LOD and decimation handle very dense line or candlestick data where possible.
- Reuse ChartGraphic and Chart instances when updating dashboards instead of destroying and recreating UI objects.

TROUBLESHOOTING

Problem	Check
Chart is blank	Confirm the GameObject has ChartGraphic or another host, has a valid size, and has marks from code or a data source.
No data appears from a source	Confirm the data source targets the correct chart, then click Generate or Refresh. For file/web sources, inspect ChartDataMapping.
Axes look wrong	Check chart type, bar orientation, domain, scale type, category labels, and whether the chart type hides axes by design.
Legend is missing	A legend needs series or foreground style keys. Use semantic foregroundStyle values or set LegendPosition to a visible value.
Labels overlap	Reduce visible labels, rotate axis labels, shorten category names, use a wider chart, or use label collision settings.
Live chart allocates or stutters	Prefer SetValues or RingBufferChartDataSource, reduce label count, disable overlapping animations, and avoid recreating chart objects every frame.
UI Toolkit chart does not update	Confirm ChartGuruElement is bound to the intended data source or binder, and that the data source raises DataChanged after refreshing.

API QUICK REFERENCE

CHART TYPES

2D ChartType values: Bar, Line, Point, Area, Pie, Donut, Radar, Bubble, Scatter, HeatMap, Candlestick, Sparkline, MiniBar, MiniPie, MiniDonut, Indicator.

COMMON BUILDER CALLS

```
Chart.Create()
Chart.Create<T>(data, item => new BarMark(...))
builder.Add(mark)
builder.chartType(ChartType.Bar)
builder.theme(theme)
builder.chartTitle("Title").chartSubtitle("Subtitle")
builder.chartXAxis(axis => axis.Label("X"))
builder.chartYAxis(axis => axis.Label("Y"))
builder.chartXScale(min, max).chartYScale(min, max)
builder.chartLegend(LegendPosition.Bottom)
builder.chartScrollableAxes(ScrollableAxes.Horizontal)
builder.chartXVisibleDomain(length)
builder.chartForegroundStyleScale(dictionaryOrColorScale)
builder.chartSymbolScale(dictionary)
builder.chartSymbolSizeScale(new ClosedRange<float>(4f, 24f))
builder.showDataLabels()
builder.Animate(AnimationStyle.Staggered)
builder.Build()
```

COMMON RUNTIME CALLS

```
chart.SetValues(values);
chart.SetData(marks);
chart.AddMark(mark);
chart.ClearMarks();
chart.MorphTo(ChartType.Pie);
chart.ClearSelection();

ChartGraphic graphic = ChartCanvasHelper.RenderToRectTransform(chart, rect, true, "Chart");
ChartCanvasHelper.RemoveChart(graphic);
```

SUPPORT

Web: https://www.wetzold.com/tools

Discord: https://discord.gg/uzeHzEMM4B