

CHART GURU 3D

Chart Guru 3D is the spatial module included with Chart Guru. It adds world-space chart meshes, 3D marks, Z-axis semantics, camera pose and projection, independent Canvas and UI Toolkit outputs, captures, and animated changes between 2D and 3D views.

This manual supplements the base Chart Guru guide. Use that guide for shared definitions, assets, data frames, layers, palettes, legends, annotations, selection, animation and the Chart Designer. The 3D module is installed with Chart Guru and needs no separate purchase or optional import.

> Scope: when behavior is identical to Chart Guru 2D, this guide only calls out the 3D-specific difference. Swift Charts 3D parity features and Chart Guru-only extensions are named separately.

Contents

Chart Guru 3D 1
What the 3D Module Adds 4
Visual Examples from the Engine 4
Setup and First Scene Chart 6
Requirements 6
Fastest Scene Chart 6
Authoring in the Inspector 6
Authoring with Chart3D 7
Chart3DBuilder 7
Code Path 7
Lifting 2D Marks into 3D 8
3D Marks and Chart Types 9
Swift Charts 3D Parity Marks 9
Surface Color, Clipping And Boundaries 9
Chart Guru 3D Extension Marks 9
When to Choose 3D 10
Z Axis Domains Ranges and Plot Size 11
Rendering Hosts Snapshots and UI Tiles 12
World Space Rendering 12
UI Outputs and Snapshot Ownership 13
UI Toolkit Output 13
Data Sources Selection and Morphing 14
Binding Data Sources 14
3D Selection and Morphing 15
Themes Materials Lighting and Pipelines 15
Theme Roles 15
Material Presets 16
Render Pipeline Support 16
Axes and Labels 16
Editor Tools 16

3D Gallery 16
Shared Chart Designer 17
Swift Charts 3D Import 17
Performance Guidance 17
Snapshot and UI Performance 18
Sample Tracks 18
3D API Quick Reference 18
Chart Types 18
3D Marks 18
Components and Utilities 19
Common Code Pattern 19
Troubleshooting 20
Troubleshooting Checklist 20
Support 21
Related Documentation 21
Useful Package References 21
Contact 21

WHAT THE 3D MODULE ADDS

Chart Guru 3D uses the shared ChartDefinition and Chart model, then renders spatial marks, Z-axis values and real 3D geometry. It keeps its own runtime assembly while the combined Chart Guru release supplies the module automatically.

Area	3D-specific delta
Chart model	ChartDimension.ThreeD, ZAxis, Pose, Projection, Z scale range, Z domains, and SelectedZValue.
Marks	IChart3DMark implementations contribute meshes, instances, ribbons, surfaces, sectors, or slabs.
Host	Chart3DComponent renders world-space meshes and labels.
Camera	Chart3DPose stores azimuth, inclination, and optional distance; projection can be Automatic, Orthographic, or Perspective.
Styling	One ChartTheme supplies palette, text and its Appearance3D block for materials, lighting, surfaces, axes, labels, shadows and environment.
Output	Chart3DImage, Chart3DTexture and Chart3DElement provide live UI images; Chart3DComponent.Capture and Chart3DSnapshot create textures and PNGs.
Editor	3D Gallery templates, the shared Chart Designer, tabbed inspectors, 2D/3D conversion, material presets, C# and Swift code, and PNG export.

Use 3D when depth carries data, a chart belongs inside a game world, or a surface needs spatial inspection. Keep dense comparison dashboards in 2D when that makes values easier to read.

VISUAL EXAMPLES FROM THE ENGINE

The figures in this guide are engine-rendered charts. Scene geometry, live UI outputs and image exports share the same spatial rendering system, with each output owning its view.

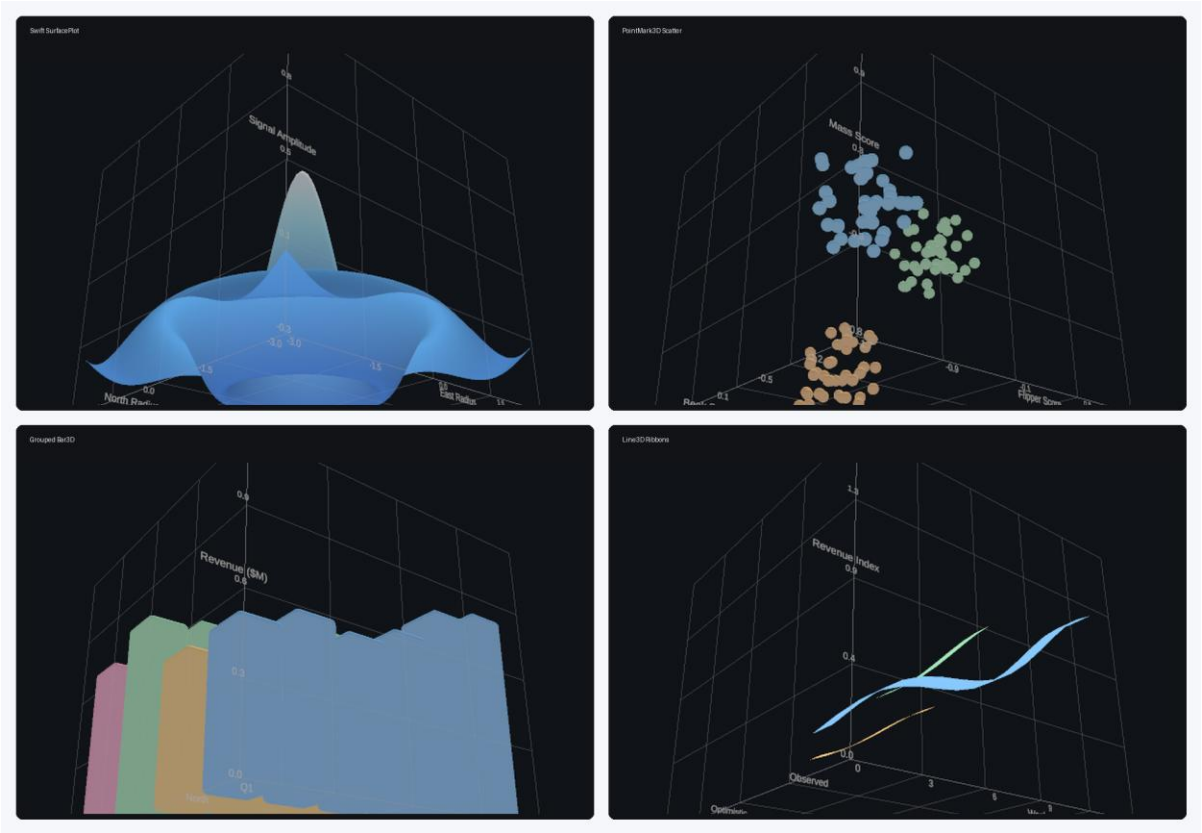


Figure 1. Engine-rendered 3D families: function surfaces, point clouds, grouped bars with depth lanes and connected spatial series.

SETUP AND FIRST SCENE CHART

REQUIREMENTS

- Import the combined Chart Guru package. Chart Guru 3D is included under the package's Reuse folder; no separate module import is required.
- Unity 2022.3 or newer is supported. The manifest declares Burst, Collections, Input System, Mathematics and TextMeshPro. The bundled samples use Active Input Handling set to Input System Package (New).
- Assemblies that create 3D charts in code must reference the ChartGuru.3D runtime assembly. Its namespace is ChartGuru.

FASTEST SCENE CHART

1. Open GameObject > Chart Guru > 3D > Point Chart, Bar Chart or Surface Chart. GameObject > UI > Chart Guru > 3D Bar Chart creates a Canvas output.
2. Unity creates the chart with a serialized inline definition and editor preview data. Add a source, a data asset or real table rows before relying on it in Play Mode or a build.
3. If the scene has no light, the menu action creates a directional light for the chart.
4. Use Plot Size to fit the chart into the scene. The pose turns the chart content about its center; a Chart3DCameraRig can instead frame and orbit a dedicated camera.

AUTHORING IN THE INSPECTOR

- Check the Data status for missing columns, empty layers or compile warnings. Editor-only preview data helps authoring but is not runtime data.
- In Data, use Add Data Source > Manual or Mock when the chart needs a source. Select Source opens the resolved source; a source on the same object or an inherited parent can feed several charts.
- Choose a 3D chart type such as Bar3D, Point3D, Surface3D, Scatter3D, Pie3D, Donut3D, Radar3D, Line3D, Area3D, or Rectangle3D.
- Use the Data, Marks, Axes, Labels, Style, Behavior and Output tabs. Choose Surface Material in Style; configure pose, projection, plot size, axes, shadows, environment and host rendering controls where they apply.
- Use Output > Export PNG to save an image. Rebuild() recompiles a changed definition; RebuildMeshes() refreshes geometry without recompiling its data.

AUTHORING WITH CHART3D

CHART3DBUILDER

Use Chart3D.Create when the chart is authored in code. The builder creates a normal Chart instance, sets it to ChartDimension.ThreeD, accepts 3D marks, and applies camera, Z-axis, domain, range, stacking, and 3D theme settings before Build returns the Chart.

- Theme(theme) still applies the shared ChartTheme for palette and text choices.
- The shared theme's Appearance3D controls the spatial look. Choose a chart-level Surface Material in Style, or a host Material Override when one chart requires a custom material.
- ChartZAxis, ChartZScaleDomain, ChartZScaleRange, Chart3DPose, Chart3DCameraProjection, and ChartZSelection are the most important 3D-only builder calls.

CODE PATH

This example combines a function surface with a point on that surface. The X, Y and Z values all fit within the explicitly configured domains. Assign the chart to a world-space host and keep it alive while that host renders it.

```
using ChartGuru;
using UnityEngine;

public static class SurfaceSceneExample
{
    public static Chart Create(GameObject owner)
    {
        float x = 0.3f, z = 0.5f;
        Chart chart = Chart3D.Create()
            .Add(new SurfacePlotMark("X", "Y", "Z",
                (px, pz) => Mathf.Sin(px) * Mathf.Cos(pz))
                .ForegroundStyle(SurfaceStyle.HeightBased)
                .YRange(-1f, 1f).Resolution(64, 64))
            .Add(new PointMark3D(x, Mathf.Sin(x) * Mathf.Cos(z), z)
                .Symbol(Symbol3D.Sphere).SymbolSize(0.07f)
                .ForegroundStyle(Color.cyan))
            .ChartXAxis(axis => axis.Label("X"))
            .ChartYAxis(axis => axis.Label("Y"))
            .ChartZAxis(axis => axis.Label("Z"))
            .ChartXScaleDomain(-1f, 1f)
            .ChartYScaleDomain(-1f, 1f)
            .ChartZScaleDomain(-1f, 1f)
            .Chart3DPose(Chart3DPose.Default)
            .Chart3DCameraProjection(Chart3DCameraProjection.Perspective)
            .Build();
        Chart3DComponent host = owner.AddComponent<Chart3DComponent>();
        host.SetChart(chart);
        host.PlotSize = new Vector3(2f, 1.5f, 2f);
        return chart;
    }
}
```

Chart3DBuilder.Add accepts 3D marks. Use the Z-lifting helpers when starting from a 2D mark, rather than passing the unconverted mark directly. Rule overlays do not replace an already inferred primary 3D mark type. Detach or release the host before disposing the chart returned by this example.

LIFTING 2D MARKS INTO 3D

Use the Z extension methods when existing 2D authoring code already creates PointMark, BarMark, LineMark, or AreaMark objects and you only need to add a depth lane.

```
PointMark3D point = new PointMark(0f, 12f)
    .ForegroundStyle(Color.green)
    .Z(2f);

Bar3DMark bar = new BarMark(1f, 42f)
    .ForegroundStyle(Color.magenta)
    .Z(0f)
    .Depth(0.65f);
```

3D MARKS AND CHART TYPES

SWIFT CHARTS 3D PARITY MARKS

PointMark3D, RectangleMark3D, RuleMark3D, and SurfacePlotMark are the closest matches to Swift Charts 3D. Prefer them when a chart is meant to mirror Swift Charts 3D concepts.

Swift parity mark	What it represents
PointMark3D	A point in X/Y/Z space rendered as a sphere, cylinder, cone, cube, billboard or custom mesh. SymbolSize is a diameter in plot units; Swift sphere radii are converted on import and export.
SurfacePlotMark	A bivariate surface y = f(x, z), with height-based, normal-based, gradient, or solid styling.
RectangleMark3D	A ranged rectangle in a plane. Swift Charts 3D rectangles span two axes; full three-axis boxes are Chart Guru extensions.
RuleMark3D	A 3D reference or drop line between two 3D points.

SURFACE COLOR, CLIPPING AND BOUNDARIES

Use *ForegroundStyle(SurfaceStyle.HeightBased)* to color a surface by height, or *NormalBased* to color it by orientation. The gradient overload supplies a custom palette. *YRange(min, max)* sets the height-to-color mapping and does not clip geometry. Use the Chart Guru extension *ClampY(min, max)* only when sampled heights should be clamped before domain inference and projection.

Undefined function samples preserve empty interiors. The default *Boundary(SurfaceBoundaryMode.ClipToDomain)* closes nonfinite edges against the appropriate Y-domain boundary. *Boundary(SurfaceBoundaryMode.Open)* is a Chart Guru extension that leaves those edges open. Surface resolution changes the sampling density; it does not turn undefined interior data into valid samples. Keep the Swift parity controls and Unity-specific extensions distinct when documenting a chart.

CHART GURU 3D EXTENSION MARKS

Chart Guru also adds Unity-specific 3D visualizations that Swift Charts 3D does not provide directly, including extruded bars, line ribbons, area prisms, 3D pies and donuts, radar meshes, sized scatter symbols, and serialized surface grids.

```
Chart3DBuilder builder = Chart3D.Create()
    .ChartType(ChartType.Bar3D)
    .Grouped()
    .Add(new Bar3DMark(0f, 10f, 0f).Width(0.6f).Depth(0.6f))
    .Add(new Bar3DMark(0f, 14f, 1f).Width(0.6f).Depth(0.6f));
```

WHEN TO CHOOSE 3D

Use 3D for surfaces, spatial comparisons, depth lanes, scene-embedded charts, or premium preview imagery. Avoid 3D when the reader needs maximum label density or precise 2D comparison.

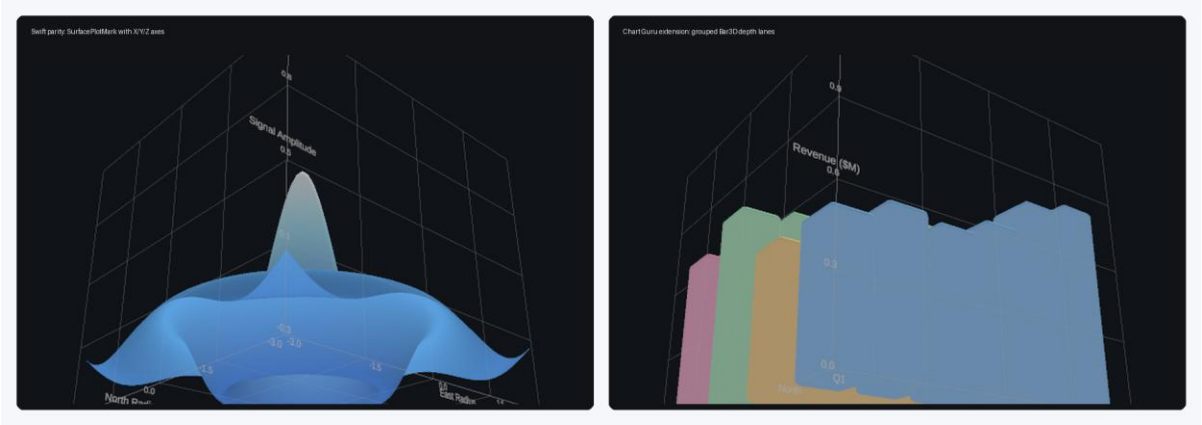


Figure 2. 3D mark deltas: a Swift-parity SurfacePlotMark with X/Y/Z axes and a Chart Guru grouped Bar3D extension showing depth lanes and extrusion.

Use Swift-parity marks when matching native Chart3D concepts. Use Chart Guru extension marks when the Unity scene needs extruded bars, ribbons, sectors, radar meshes, or other 3D-only visual forms.

Z AXIS DOMAINS RANGES AND PLOT SIZE

The 3D module separates data domains from plot ranges. Domain methods describe data values. Range methods describe normalized plot extents, then PlotSize scales the result for the host.

- Prefer ChartZScaleDomain for data semantics. Explicit scale ranges use plot units where the automatic extent is -0.5 to 0.5: a range of -0.65 to 0.65 spans 1.3 plot widths at any host size.

ChartXScaleDomain, *ChartYScaleDomain* and *ChartZScaleDomain* describe the data. Their range counterparts choose plot-space output extents. Parameterized *ChartXScale*, *ChartYScale* and *ChartZScale* overloads also support the configured nonlinear scale type and its parameter. Geometry, axes, labels and selection use the same configured scales, so do not pre-transform only the rendered marks.

Enable *Chart3DComponent.PreserveDataProportions* when equal distances in comparable X/Y/Z units should occupy equal visual distances; snapshots provide the same choice in *Chart3DSnapshotOptions*. Choose explicit plot size and ranges deliberately when axes have different units or when a dashboard needs a particular composition. A camera pose changes the view, while a scale domain changes the data mapping.

RENDERING HOSTS SNAPSHOTS AND UI TILES

Host or helper	Use it when	Key controls
Chart3DComponent	The chart belongs in a 3D scene as real world-space geometry.	Definition or ChartAsset, data asset or source, PlotSize, axes, shadows, environment, Rebuild and RebuildMeshes.
Chart3DCameraRig	The chart needs a dedicated orbit camera or reproducible preview pose.	Chart3DPose, projection mode, distance, reset, optional child camera.
ChartRaycaster and shared interaction	A scene chart should receive pointer input without a collider.	Camera raycaster, EventSystem, selection, hover, orbit, wheel or pinch zoom.
Chart3DSnapshot	You need static textures, PNGs, preview tiles, gallery cards, or store imagery.	AutoFrame, supersample, transparent background, RenderTo, EncodeToPng.
Chart3DTexture	A Canvas RawImage should show a live 3D chart.	Own or Source content, view pose and projection, theme, output size, quality, Invalidate and timing.
Chart3DImage	A MaskableGraphic should show a live 3D chart with masking or custom UI material behavior.	Own or Source content, view, masking, material, preserve aspect, Invalidate and timing.
Chart3DSnapshotOptions	Output must be consistent across previews, documentation, and marketing captures.	Theme, Appearance, pose, projection, plot size, framing, resolution and supersampling.

WORLD SPACE RENDERING

- Chart3DComponent renders a Chart as world-space 3D geometry.
- It owns the surface mesh, a line mesh for axes/grid/rules, TextMeshPro label objects, optional shadow/environment handling, and instanced symbol batches.
- Chart3DCameraRig applies Chart3DPose and Chart3DCameraProjection through an optional child camera.
- Add ChartRaycaster to the camera and use an EventSystem for scene pointer events. The shared input path supports hover and tooltips, mark selection, drag orbit, double-tap reset, and wheel or pinch zoom; chart colliders are not required.

- Chart3DTexture renders a live 3D chart into a RawImage for Canvas dashboards.
- Chart3DImage is a MaskableGraphic variant for masks, rounded-corner materials, and custom Graphic workflows.

UI OUTPUTS AND SNAPSHOT OWNERSHIP

Chart3DTexture renders into a Canvas *RawImage* and *Chart3DImage* is a maskable Graphic. In Own mode each hosts its own inline definition or ChartAsset, data and transitions. In Source mode it mirrors a *Chart3DComponent's content and plot size. Each output owns a private view,* so orbiting one output does not turn the scene chart or another output. Use Chart3DElement for UI Toolkit with the same definition, assets and data-source model.

*Outputs redraw when chart data, theme, options, pixel size or pose changes. Refresh Hz adds periodic render*s for stateful content that changes without notifying the chart; zero is the normal change-driven mode. Invalidate() requests a redraw. Manual timing r*ender*s only when Refresh *i*s called. Auto Rotate turns the view in Play Mode and leaves Edit M*ode* at its initial pose.

```
using System.IO;
using ChartGuru;

public static class SnapshotPngExample
{
    public static void Save(
        Chart chart,
        Chart3DSnapshotOptions options,
        string writablePath)
    {
        byte[] png = Chart3DSnapshot.EncodeToPng(chart, options);
        File.WriteAllBytes(writablePath, png);
    }
}
```

Chart3DComponent.Capture(Chart3DCaptureOptions) returns a caller-owned RGBA texture with the host's plot size, appearance and framing, independently of the scene camera. OverridePose, Pose and Projection choose a different capture view. A running transition is captured at its destination. Destroy the returned texture when finished.

Chart3DSnapshot.RenderTo renders into a caller-provided RenderTexture without CPU readback. Render returns a caller-owned Texture2D; RenderOrbit returns caller-owned frame textures. Destroy those textures after use. EncodeToPng releases its intermediate texture. Chart3DSnapshotOptions uses the shared Theme and optional Appearance overrides.

UI TOOLKIT OUTPUT

Use Chart3DElement in UXML or code for a live spatial chart. It supports SetChart, SetData, MorphTo, Pose, Projection, AutoRotate, Zoom, Lift and PlayLift, plus shared tooltips. Subscribe to DataPointClicked, DataPointSelected, DataPointHovered, SelectionCleared or TooltipChanged for

application UI. UIToolkitChartBinder can bind a named 3D element in a UIDocument or compatible PanelRenderer.

DATA SOURCES SELECTION AND MORPHING

Chart Guru 3D uses the shared ChartDataSource pipeline. A source publishes a frame; the host's definition and layers compile it into spatial marks. Put ManualChartDataSource, MockChartDataSource or a file or web source on the host or an inherited parent.

Input situation	Definition-driven 3D mapping
Single-series points	Bind X, Y and Z channels to numeric, date or category columns. An automatic layer uses row index and depth lanes when the data has no explicit spatial positions.
Multi-series data	Bind several value columns for wide-series data, or a Series column for long data. Without explicit Z, supported marks use series depth lanes.
Surface3D grid	Bind grid X, Z and height columns for a sampled surface, or use a Surface function layer. Missing grid values remain holes.
Pie3D / Donut3D	Values form extruded sectors. Pie and Donut fill their host footprint while preserving the intended data proportions.
Radar3D	Categories define the spokes; values set the radial distances and series occupy depth levels. The radial domain controls the rings and marks consistently.
Range entries	Rectangle3D and Rule3D can use X/Y/Z start and end values.

BINDING DATA SOURCES

The host chooses runtime SetData first, then its enabled resolved source, a ChartDataAsset, and the definition's inline table. A source on a parent is inherited only by charts without their own data. Configure mapping in the chart's Marks layers; there is no Target Chart3DComponent field.

```
Chart3DComponent host = gameObject.AddComponent<Chart3DComponent>();
host.Definition.Type = ChartType.Bar3D;
MockChartDataSource mock = gameObject.AddComponent<MockChartDataSource>();
mock.Regenerate();
host.Rebuild();
// The host resolves the source on its GameObject and compiles its frame.
```

3D SELECTION AND MORPHING

ChartRaycaster routes scene pointer events through Chart3DHitTester. The hit includes the selected mark and X/Y/Z values; Canvas and UI Toolkit outputs use their own view for the same interaction. Enable selection and hover in Behavior, and use Orbit for drag rotation.

- MorphTo changes a definition-driven 3D host to another 3D chart family and retargets compatible channels.
- Value-only SetData(frame, shapeChanged: false) updates compatible marks while columns, row count and categories stay fixed; unsupported shapes rebuild safely.
- A custom ChartDataSource can overwrite the values of its published columns and call PublishValueChange once for all changed columns.
- PlayLift(0f) flattens a spatial chart and PlayLift(1f) lifts it into depth without changing its data; the shared Designer uses this for dimension transitions.
- Chart3DImage, Chart3DTexture and Chart3DElement expose the same lift and view controls while preserving their independent output views.
- Function surfaces and sampled grids keep their supported update paths; do not label a changed row count, columns or categories as a value-only change.

Shared chart workflows also apply to 3D data: semantic selection links can propagate point and X/Y/Z range state, and accessibility descriptors, audio graphs and semantic CSV/JSON exports can consume 3D values through the shared chart model. See the base Chart Guru manual for setup and ownership. Configure the 3D host's screen-space frames when bridging accessibility, and choose a semantic point match when linked charts use different mark orders.

THEMES MATERIALS LIGHTING AND PIPELINES

THEME ROLES

Use one shared ChartTheme. Its palette and typography serve 2D and 3D charts, and Appearance3D adds spatial material, lighting, label, shadow, environment and surface settings. The theme Inspector's 3D tab previews these changes.

```
ChartTheme theme = ChartTheme.Dark;
Chart chart = Chart3D.Create()
    .Theme(theme)
    .Add(new Bar3DMark(0f, 10f, 0f))
    .Build();
// Edit a project ChartTheme asset's 3D appearance for a reusable custom look.
```

MATERIAL PRESETS

Set the chart's Surface Material under Style > Material. Vertex-lit matte, glossy, dark polished and glass presets, plus vertex-unlit rendering, supply common looks. A host's Material Override takes precedence over the chart preset and theme material.

Appearance3D supplies roughness and *metalness* defaults. Marks can override them with Roughness(value) and *Metalness(value)*, from zero to one. The material preset and these per-mark overrides remain independent. Use scientific height or normal colormaps, or custom gradients, for surfaces; check transparency and background together.

RENDER PIPELINE SUPPORT

The vertex-lit and vertex-unlit chart shaders include Built-in, URP and HDRP programs. They render forward and do not contribute baked GI. Pipeline packages select their matching shader programs; HDRP exposure, tonemapping and fog can still affect the final appearance.

- Check surfaces, instanced symbols, transparent marks, rule widths, labels and the final output in the target render pipeline. Use supersampling for offscreen edge quality.

AXES AND LABELS

Axes use the chart's X/Y/Z AxisConfig settings for grid, ticks, labels and titles. Hide an axis title with Show Title while retaining its name for tooltips. Dates work on every axis and log ticks keep small fractional values readable. Scene titles, subtitles and legends are placed outside the plot and included in framing.

EDITOR TOOLS

3D GALLERY

Open Tools > Chart Guru > 3D Gallery. It browses chart templates with the same family filters, search, badges and card actions as the 2D Gallery. A Swift Charts badge marks export-compatible templates; Chart Guru badges identify extension features.

- Use families and search to filter chart types.
- Rotate on Hover turns the tile under the pointer. Switch it off for stationary previews.
- Double-click a card or press Enter to open a copy in the shared Chart Designer. Card actions and the context menu create the chart in the scene, export PNG, copy code or ID, and open the 3D manual.

- Open the large preview to orbit the chart and compare pose, projection or theme without changing the template. Use the 2D Gallery button to carry the current search into the companion gallery. Browsing does not add scene objects.

SHARED CHART DESIGNER

Open Tools > Chart Guru > Chart Designer and choose 3D, or open a 3D Gallery template. Edit data, mark layers, style, axes, legend and actions in the same document as 2D charts. The dimension control converts compatible families while retaining their data and settings; a 2D Canvas inspector conversion requires a world-space Canvas.

```
Chart chart = Chart3D.Create()
    .Add(new PointMark3D(0f, 1f, 0f))
    .Chart3DPose(Chart3DPose.Default)
    .Chart3DCameraProjection(Chart3DCameraProjection.Perspective)
    .Build();
```

SWIFT CHARTS 3D IMPORT

Use the Designer's C# and Swift panels to inspect generated code. Paste supported Swift Chart3D source to replace the document, then review the chart and diagnostics. PointMark, RuleMark, planar RectangleMark and SurfacePlot are the native parity forms. Full boxes, extruded bars, pies, radar and other extensions are identified separately.

The importer reads extractable literal data and supported surface closures, and reports unsupported application data, functions and modifiers by name. The first scale modifier on each axis takes precedence, so combine domain, range and type in that call. The Swift panel lists Chart Guru settings that cannot be exported; C# generation keeps supported spatial styling and material presets.

PERFORMANCE GUIDANCE

Tune surfaces, previews, and scene chrome based on how the chart is used. A large interactive scene chart has different constraints than a small dashboard thumbnail.

Area	Guidance
Surface resolution	Tune SurfacePlotMark.Resolution or the theme's 3D surface resolution. Higher values add vertices and triangles; use lower resolution for small UI outputs.
Point clouds	Prefer instanced PointMark3D or Scatter3DMark symbols. Keep symbol sizes and custom meshes reasonable.
Scene chrome	Disable axes, labels, shadow plane, or environment for small tiles and high-frequency previews.

Area	Guidance
Snapshots	Use RenderTo for an existing RenderTexture without CPU readback, Capture for a host-faithful still, and EncodeToPng when PNG bytes are required.
UI refresh	Keep Refresh Hz at zero for change-driven output. Use periodic refresh only for unannounced time-dependent content, and avoid oversized textures or high supersampling on small tiles.

SNAPSHOT AND UI PERFORMANCE

Each UI output owns a private offscreen view and redraws on relevant changes. Reuse the output instead of constructing a snapshot every frame. Reserve CPU readback and PNG encoding for deliberate exports.

- Choose render resolution near the displayed pixel size and keep supersampling within the needed quality. Invalidate after external changes that do not notify the chart; manual timing also needs an explicit Refresh call.

SAMPLE TRACKS

Start with Examples/00_Entry/Chart Guru 3D Showcase. The tracks 01_ChartTypes, 02_DataAndInteraction, 03_StyleAndMotion and 04_Workflows cover spatial marks, data sources, interaction, materials, motion and outputs. Charts are saved definitions that can be inspected before Play Mode.

Open 04_Workflows/Swift Charts 3D Import to compare imported live charts with their native reference renders and inspect the Swift and generated C#. The production examples Terrain Survey, Sound Spectrogram and Charts in a Game World combine spatial charts with real datasets or stated simulation inputs.

3D API QUICK REFERENCE

CHART TYPES

3D ChartType values: Point3D, Rule3D, Rectangle3D, Surface3D, Bar3D, Line3D, Area3D, StackedArea3D, Pie3D, Donut3D, Radar3D, Scatter3D.

3D MARKS

Core 3D marks: PointMark3D, RuleMark3D, RectangleMark3D, SurfacePlotMark, SurfaceGrid3DMark, Bar3DMark, Line3DMark, Area3DMark, StackedArea3DMark, Sector3DMark, Donut3DMark, Radar3DMark, Scatter3DMark.

COMPONENTS AND UTILITIES

Runtime components and utilities include Chart3DComponent, Chart3DCameraRig, ChartRaycaster, Chart3DTexture, Chart3DImage, Chart3DElement, Chart3DViewSettings, Chart3DSnapshot, Chart3DSnapshotOptions, Chart3DCaptureOptions and the shared ChartTheme with Appearance3D.

COMMON CODE PATTERN

```
Chart chart = Chart3D.Create()
    .ChartType(ChartType.Bar3D)
    .Add(new Bar3DMark(0f, 10f, 0f).Width(0.6f).Depth(0.6f))
    .ChartZAxis(axis => axis.Label("Series"))
    .Chart3DPose(Chart3DPose.Default)
    .Chart3DCameraProjection(Chart3DCameraProjection.Orthographic)
    .Build();
```

TROUBLESHOOTING

- If the chart is blank, check Data diagnostics, real marks or rows, 3D Chart Type, non-zero Plot Size and the active camera. Editor preview data is not included in Play Mode or builds.
- If materials are pink, select a valid Surface Material or Material Override and confirm the shader program supports the installed pipeline version.
- If depth looks flat, check Z mapping, automatic series lanes, Z domain and plot range, PlotSize.z and Preserve Data Proportions.
- If axes or labels are missing, enable Draw Axes, inspect Axis Labels and the shared theme's 3D label settings, and check contrast against the background.
- If scene selection does nothing, add ChartRaycaster to the rendering camera, keep an EventSystem and input module in the scene, and enable the applicable Behavior settings.
- If a UI tile redraws too often, keep Refresh Hz at zero, disable unnecessary Auto Rotate or animations, and check for repeated data or definition changes.
- If a snapshot is black or cropped, start with Hero options, enable AutoFrame, raise FramingPadding, and verify theme/materials.

TROUBLESHOOTING CHECKLIST

Problem	Check
Chart is blank	Data diagnostics, mark count, 3D chart type, PlotSize, and active camera.
Pink materials	Surface preset, material override, shader fallback, and active render pipeline.
Data source ignored	Enabled source on the chart or eligible parent, published frame, runtime data overrides, and Marks channel mappings.
Depth looks flat	Explicit Z values, series lanes, ChartZScaleDomain, ChartZScaleRange, and PlotSize.z.
Selection missing	Camera ChartRaycaster, EventSystem and input module, Behavior selection settings, and the correct view for UI outputs.
UI tile too expensive	Displayed pixel size, supersampling, Refresh Hz, Auto Rotate, scene chrome and repeated data changes.
Snapshot cropped	Hero options, AutoFrame, FramingPadding, PlotSizeOverride, and theme/material validity.
HDRP mismatch	Installed HDRP version, chart shader program, material overrides, transparency, exposure, tonemapping and fog.

SUPPORT

RELATED DOCUMENTATION

Use the base Chart Guru manual for shared concepts and the Examples README for the current scene tracks, datasets and interactions. The included 3D AI skills explain setup, outputs, surfaces, Swift import, input, live data and themes.

USEFUL PACKAGE REFERENCES

```
Package references:
Examples/README.md
Examples/Shared/Data/README.md
AI/README.md and AI/Skills
PNG exports use the writable path chosen by the application.
```

CONTACT

Web: https://www.wetzold.com/tools
Discord: https://discord.gg/uzeHzEMM4B


IMPOSSIBLE
ROBERT