Geometric Ratio Model JSON Handbook

A Programmable Geometry Reference Based on the Geometric Ratio Model

Version: 1.0

Release Date: May 20, 2025

Author: M.C.M. van Kroonenburgh, MSc

Website: https://inratios.com

i-Depot Registration: 151927 (BOIP – Benelux Office for Intellectual Property)

Short Description

This handbook provides a structured, machine-readable implementation of the Geometric Ratio Model (GRM) in JSON format. It defines fixed ratios, tolerances, and validation logic for geometric shapes such as circles, spheres, and hexagons, enabling consistent use across AI models, CAD software, educational tools, and analytical workflows.

Inhoud

1. Introduction

1.1 Purpose of this handbook

This handbook defines the standardized data structure and implementation logic of the Geometric Ratio Model (GRM) in a programmable format. It provides a unified reference for using GRM-ratios, conditions, tolerances, and derived units in machine-readable systems such as AI models, CAD tools, educational platforms, and software validation engines.

The handbook is primarily intended for:

- AI developers working with shape recognition or geometric reasoning
- CAD and design tool creators seeking standardized proportional metrics
- Educators building tools or curricula around geometric concepts
- Engineers or analysts verifying shape-based models or estimations

1.2 Background

The GRM was originally introduced by M.C.M. van Kroonenburgh, MSc as an alternative to irrational constants such as π. By expressing fixed ratios for perfectly inscribed shapes, the model enables a consistent, rational approach to:

- Perimeter (SPU),
- Area (SAU),
- and Volume (SVU)

The GRM's logic is detailed in the official whitepaper series, and further expanded through application proposals (e.g., for pixel-based analysis and tolerance modeling). These principles form the theoretical foundation of the JSON schema presented in this handbook.

The GRM differs from traditional geometry by eliminating the need for irrational numbers such as π. Instead, it defines dimensionally consistent, fixed ratios based on inscribed shapes.

Two critical innovations strengthen the applicability of the GRM:

The Role of the Radius

In the GRM, the radius of an inscribed circle (or sphere) is treated not as a primary input, but as a derived geometric constant.

Specifically:

- For a circle inscribed in a square of perimeter 1.0 SPU, the radius is defined as r = 0.1250 SPU
- This fixed ratio emerges from the square-to-circle relationship, bypassing π-based calculations (e.g., no r × 2π)

This makes the model particularly suitable for use in digital systems, where discrete ratios are preferable to irrational approximations.

The Role of Tolerance and Deviation Modeling

Traditional geometric validation often lacks clarity on how close is close enough. The GRM integrates programmable tolerance bands and deviation classifications for each ratio:

- Tolerance ranges allow for numerical precision thresholds (e.g., ±0.005 SPU)
- Deviation classification supports outputs like: perfect, minor deviation, major deviation

This enables AI models or validation scripts to assess shapes not just mathematically, but also contextually, supporting:

- Shape verification
- Classification of imperfections
- Feedback in design and learning systems

Additional application domains (as developed in GRM Proposals)

In addition to theoretical modeling, the GRM has been extended into programmable applications. These include:

- **Pixel-Based Ratio Measurement**: measuring geometric properties in digital images and grid-based representations
- **Shape Recognition using GRM tolerances**: classifying observed forms into GRM-conforming or deviating categories
- **Proportional Design**: applying GRM ratios to modular layouts and interface design
- **Volume Estimation without Displacement**: inferring the volume of 3D shapes such as spheres and cubes using fixed GRM volumetric ratios (SVU), without

relying on physical displacement methods. This enables integration with CAD environments, 3D scanning, and simulation systems.

These applied domains are translated into JSON-compatible structures throughout this handbook and are

1.3 Scope and Structure

This handbook is not a mathematical theory document, but a technical implementation guide. It translates abstract GRM concepts into structured, verifiable, and extensible JSON entries.

The structure of the handbook includes:

- A schema specification (field-by-field JSON definition)
- A data dictionary for standard shapes
- Use cases and implementation examples
- Guidelines for integrating GRM into AI tools or software applications
- A glossary and reference tables for ratios and tolerances

2. GRM logic recap

This chapter summarizes the theoretical principles and dimensional logic of the Geometric Ratio Model (GRM). It forms the conceptual foundation on which the JSON structure is based.

2.1 Fixed ratios instead of π

The GRM replaces irrational constants such as π with fixed ratios that are dimensionally consistent and based on perfectly inscribed shapes.

These ratios are:

Quantity	Ratio (decimal)	Unit	Applies To
Perimeter	0.7854	SPU	Circle in square (2D)
Area	0.7854	SAU	Circle in square (2D)
Volume	0.5236	SVU	Sphere in cube (3D)

All values are defined within a reference frame of unit perimeter, area, or volume (i.e., the containing square or cube is 1 SPU, 1 SAU, or 1 SVU, respectively).

2.2 Dimensional consistency and units

The GRM introduces standardized, dimensionless units to express ratios:

- SPU (Standardized Perimeter Unit) – used in 1D/2D perimeter calculations
- SAU (Standardized Area Unit) – used for surface/area-based ratios
- SVU (Standardized Volume Unit) – used for 3D volumetric ratios

These units enable a consistent comparison across different shapes, dimensions, and systems. For example, any circle inscribed in a square will always have a perimeter of 0.7854 SPU, regardless of absolute size, as long as the perimeter of the square is 1 SPU.

2.3 Perfect inscription as a condition

All GRM ratios are only valid under strict geometric conditions:

- A shape must be perfectly inscribed in a square (2D) or cube (3D), meaning it touches all internal boundaries of the container without distortion.

This requirement ensures that the ratios remain fixed and meaningful. Any deviation from this condition (e.g. partial overlap, non-centered placement, irregular edges) may render the GRM ratio inapplicable or trigger a tolerance classification.

2.4 Application across dimensions

The GRM supports ratio-based modeling in:

- 1D: Linear structures and edge-based ratios
- 2D: Shapes such as circles, hexagons, and triangles within squares
- 3D: Volumes like spheres, cubes, and potentially more complex solids

The dimensional logic ensures that all ratios remain coherent within their scope, making the GRM suitable for layered modeling, nesting, and compositional geometry.

2.5 Derived metrics: Radius, tolerance, and pixel grid applications

1. **Radius**

   Beyond the primary GRM ratios, several derived constants and mechanisms have been introduced to enhance the model's practical use in digital systems:

   Radius (r = 0.1250 SPU)

   When a circle is perfectly inscribed in a square of 1.0000 SPU perimeter, the resulting radius is:

   r = 0.1250 SPU
   (equivalent to one-eighth the perimeter of the square)

   This fixed radius becomes a key internal reference in calculations and can be used in designs, simulations, and visual representations.

2. **Tolerance Ranges and Classifications**

   To support real-world variability, the GRM incorporates tolerance bands that define acceptable deviations from the ideal ratio values.

   Each ratio may include:

   - A tolerance range (e.g. ±0.005)
   - A classification system (e.g. perfect, minor deviation, major deviation)

   These elements are especially useful in:

   - AI shape recognition
   - Measurement validation
   - CAD model checking

3. **Pixel-based radius and grid approximations**

   In pixel-based environments (e.g. digital image analysis), the radius and perimeter must often be approximated over discrete grids. GRM-based tools use the fixed radius and ratio values to evaluate whether a digital shape conforms to an ideal geometric model.

   These approximations are addressed more fully in the GRM Application Proposal: "*Pixel-Based Ratio Measurement*".

2.6 Application-driven extensions of GRM logic

Several application proposals have extended the core GRM ratios into practical implementation domains. These applications preserve the original geometric logic, but translate it into programmable or domain-specific formats.

1. **Pixel-based ratio measurement**

   This approach approximates GRM ratios in discrete grids, enabling shape analysis within rasterized images. Each shape's perimeter, area, or radius is mapped to a pixel count, which is then compared to the GRM reference ratios. Use cases include:

   - Image analysis
   - Medical segmentation
   - AI visual recognition

2. **Shape recognition with deviation classification**

   By defining tolerance ranges and confidence bands, the GRM supports a classification system for identifying:

   - Perfectly GRM-conforming shapes
   - Minor deviations (within tolerance)
   - Major deviations (outliers)

   Applicable in:

   - CAD validation
   - AI-detected geometry
   - Physical object measurement

3. **Proportional design principles**

   The GRM also defines layout logic where elements are arranged or sized using GRM ratios. This supports:

   - Modular interfaces
   - Proportional screen design
   - Structurally coherent blueprints

   This logic is defined in the Application Proposal "*Proportional Design with GRM*" and supports GUI design, mechanical planning, and educational tools.

4. **Volume Estimation without Displacement**

   Using fixed volumetric ratios (SVU), the GRM enables accurate estimation of object volume without requiring water displacement or empirical approximation. This is particularly relevant for:

   - Simulation environments
   - Digital twins
   - CAD model validation
   - 3D scanning

   This method is detailed in *Whitepaper IV – Volume Estimation Without Displacement*.

3. JSON schema specification

This chapter defines the data model and field structure used to represent GRM-based geometry in JSON format. The schema allows for programmable access to geometric ratios, shape validation logic, tolerance classification, and derived values such as radius.

The schema is designed to be:

- Machine-readable and interoperable across platforms
- Dimensionally consistent and scalable
- Strictly aligned with the principles of the Geometric Ratio Model (GRM)

3.1 Object structure overview

Each geometric shape is represented as a JSON object with the following hierarchical structure:

```
json

{
  "shape_id": "circle",
  "dimension": "2D",
  "geometry": "inscribed_in_square",
  "grm_units": {
    "perimeter": {
      "value": 0.7854,
      "unit": "SPU",
      "tolerance": {
        "range": [0.7800, 0.7900],
        "classification": {
          "perfect": [0.7845, 0.7865],
          "minor_deviation": [0.7800, 0.7845],
          "major_deviation": [0.7865, 0.7900]
        }
      }
    },
    "area": {
      "value": 0.7854,
      "unit": "SAU"
    },
    "radius": {
      "value": 0.1250,
      "unit": "SPU"
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "square_container_required"
  ]
}
```

3.2 Field definitions

Field	Type	Required	Description
shape_id	string	✅	Unique identifier for the shape (e.g. "circle", "sphere")
dimension	string	✅	Dimensional scope: "1D", "2D", or "3D"
geometry	string	✅	Shape's spatial condition, e.g. "inscribed_in_square"
grm_units	object	✅	Contains ratio values (e.g. perimeter, area, volume, radius)

Field	Type	Required	Description
value	number	✅	Fixed GRM value for a given metric (e.g. 0.7854)
unit	string	✅	Must be "SPU", "SAU", or "SVU"
tolerance.range	array	❌	[min, max] deviation range from ideal ratio
classification	object	❌	Deviation band tags: perfect, minor_deviation, major_deviation
conditions	array	✅	Set of validity conditions (e.g. "must_be_perfectly_inscribed")

3.3 Optional extensions (by Proposal)

Proposal	JSON Extension(s)	Notes
Pixel-Based Ratio Measurement	pixel_radius, pixel_area	Approximate values in discrete environments
Shape Recognition with Tolerance	classification, confidence_score	Based on deviation from GRM ratio
Proportional Design	design_ratio, layout_unit	Ratio rules for modular or hierarchical design
Volume Estimation without Displacement	volume.value (SVU)	Replaces empirical volume measurements

3.4 Schema flexibility and validation rules

- All ratio values must correspond to shapes that are perfectly inscribed in their container (square or cube).
- Tolerances are optional but recommended when validation or recognition is involved.
- Classification ranges must not overlap and must lie within the defined tolerance range.
- Derived fields like radius or pixel_radius must be consistent with the base ratio values.

3.5 Example entry: Sphere (3D)

```
json

{
  "shape_id": "sphere",
  "dimension": "3D",
  "geometry": "inscribed_in_cube",
  "grm_units": {
    "volume": {
      "value": 0.5236,
      "unit": "SVU",
      "tolerance": {
        "range": [0.5180, 0.5280],
        "classification": {
          "perfect": [0.5220, 0.5250],
          "minor_deviation": [0.5180, 0.5220],
          "major_deviation": [0.5250, 0.5280]
        }
      }
    },
    "radius": {
      "value": 0.1250,
      "unit": "SPU"
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "cube_container_required"
  ]
}
```

4. GRM Shape Definitions and JSON Records

This chapter provides detailed definitions for each GRM-compliant shape, including their dimensional scope, geometric configuration, fixed ratios, and programmable JSON representation.

Each section describes:

- The mathematical properties of the shape
- The applicable GRM ratios (e.g., SPU, SAU, SVU)
- Any conditions for ratio validity (e.g., perfect inscription, symmetry)
- The tolerance structure (if relevant)
- A complete JSON shape record following the schema in Chapter 3

Shapes are grouped by dimension (2D or 3D) and include both:

- Primitive reference forms (e.g., circle, sphere, cube)

- Composite or derived configurations (e.g., subtractive constructions)

The goal of this chapter is to define a standardized data representation for GRM-compatible shapes that can be used across software, AI models, CAD systems, and educational environments.

Each JSON entry is designed to be:

- Self-contained and verifiable
- Human-readable and machine-compatible
- Extensible for future implementations

4.1 Circle (2D)

Definition

A circle is defined as the set of all points in a plane that are equidistant from a central point. In the GRM framework, a valid circle must be perfectly inscribed in a square, such that it touches all four inner edges of the square without overlapping or distortion.

This configuration enables the GRM to assign fixed and dimensionally consistent values for both perimeter and area, using standardized units.

GRM Ratios for Circle

Quantity	Value	GRM Unit	Condition
Perimeter	0.7854	SPU	Inscribed in square (2D)
Area	0.7854	SAU	Inscribed in square (2D)
Radius	0.1250	SPU	Derived from square perimeter = 1 SPU

- 0.7854 SPU: ratio of the circle's perimeter within a square of 1.0 SPU perimeter
- 0.7854 SAU: ratio of the circle's area within a square of 1.0 SAU area
- 0.1250 SPU: derived fixed radius assuming a square of 1.0 SPU

Validation Conditions

GRM ratios for the circle are valid only if:

- The circle is perfectly inscribed in a square
- The square's perimeter is standardized as 1.0 SPU
- The circle touches all four sides symmetrically

If tolerance validation is used, deviation from the ideal ratio may be classified.

GRM JSON Entry

```
json
{
  "shape_id": "circle",
  "dimension": "2D",
  "geometry": "inscribed_in_square",
  "grm_units": {
    "perimeter": {
      "value": 0.7854,
      "unit": "SPU",
      "tolerance": {
        "range": [0.7800, 0.7900],
        "classification": {
          "perfect": [0.7845, 0.7865],
          "minor_deviation": [0.7800, 0.7845],
          "major_deviation": [0.7865, 0.7900]
        }
      }
    },
    "area": {
      "value": 0.7854,
      "unit": "SAU"
    },
    "radius": {
      "value": 0.1250,
      "unit": "SPU"
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "square_container_required"
  ]
}
```

Applications

This shape record is foundational to many GRM-based implementations:

- **Pixel-Based Ratio Measurement** – determining digital circle conformity in raster grids
- **Shape Recognition with Tolerances** – identifying valid circle-like forms within classification bounds
- **Proportional Design** – using the circle's ratios to structure graphic elements and layouts
- **Mathematics and geometry education** – explaining proportional reasoning without π
- **Model input for AI and visual tools** – standard ratio check for detected or generated forms

4.2 Sphere (3D)

Definition

A sphere is defined as a perfectly symmetrical three-dimensional object in which every point on the surface is equidistant from the center. In the context of GRM, the sphere must be perfectly inscribed in a cube such that it touches all six inner faces of the cube without any deformation.

This configuration enables the GRM to assign a fixed volumetric ratio to the sphere using the Standardized Volume Unit (SVU).

GRM Volumetric Ratio

Quantity	Value	GRM Unit	Condition
Volume	0.5236	SVU	Inscribed in cube
Radius	0.1250	SPU	Derived from cube perimeter = 1

- 0.5236 SVU is the GRM-fixed ratio for the volume of a sphere in a cube with volume 1.0 SVU.
- The radius is derived as one eighth of the cube's perimeter: r = 0.1250 SPU.

Validation Conditions

The GRM ratio for the sphere is only valid if:

- The sphere is perfectly inscribed in a cube
- The cube has a reference perimeter of 1.0 SPU
- The sphere touches all internal surfaces symmetrically

Optional deviation classification (if applied) may follow GRM tolerance protocols for volumetric metrics.

GRM JSON Entry

```
json
{
  "shape_id": "sphere",
  "dimension": "3D",
  "geometry": "inscribed_in_cube",
  "grm_units": {
    "volume": {
      "value": 0.5236,
      "unit": "SVU",
      "tolerance": {
        "range": [0.5180, 0.5280],
        "classification": {
          "perfect": [0.5220, 0.5250],
          "minor_deviation": [0.5180, 0.5220],
          "major_deviation": [0.5250, 0.5280]
        }
      }
    },
    "radius": {
      "value": 0.1250,
      "unit": "SPU"
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "cube_container_required"
  ]
}
```

Applications

This shape record is particularly relevant for:

- Volume Estimation Without Displacement (GRM Whitepaper IV)
- 3D model validation in CAD and simulation tools
- Digital twins and metrology where physical measurement is impractical
- AI-based shape classification based on volume properties

4.3 Hexagon (2D)

Definition

A regular hexagon is a six-sided polygon with equal edge lengths and internal angles of 120 degrees. Within the GRM framework, a valid hexagon must be perfectly inscribed in a square, such that its outermost points symmetrically touch all four sides of the container.

The GRM ratio for the area of an inscribed hexagon is defined in relation to the Standardized Area Unit (SAU) of the square.

GRM Area Ratio for Hexagon

Quantity	Value	GRM Unit	Condition
Area	0.9014	SAU	Inscribed in square (2D)

- This value is derived geometrically and validated through simulation.
- The perimeter and radius are not fixed under GRM unless a secondary constraint (e.g. symmetry axis alignment) is introduced. Therefore, only area is considered stable and standardizable.

Validation Conditions

This GRM ratio applies only if:

- The hexagon is regular (all sides and angles equal)
- The shape is perfectly inscribed in a square of 1.0 SAU
- The outermost points of the hexagon symmetrically touch all four square edges

Due to the higher complexity of inscribing a polygon in a square, visual or pixel-based validation methods may be required in practical applications.

GRM JSON Entry

```
json
{
  "shape_id": "hexagon",
  "dimension": "2D",
  "geometry": "inscribed_in_square",
  "grm_units": {
    "area": {
      "value": 0.9014,
      "unit": "SAU",
      "tolerance": {
        "range": [0.8950, 0.9075],
        "classification": {
          "perfect": [0.8990, 0.9035],
          "minor_deviation": [0.8950, 0.8990],
          "major_deviation": [0.9035, 0.9075]
        }
      }
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "square_container_required",
    "must_be_regular_hexagon"
  ]
}
```

Applications

The hexagon's GRM ratio is particularly relevant for:

- Shape Recognition – distinguishing regular hexagons from near-circular forms
- Tolerance Modeling – validating hexagonal patterns in designs (e.g., honeycomb layouts)
- Visual classification – identifying polygonal structures in image processing
- Modular proportional design – using the hexagon as a GRM-compliant building block

The shape also plays a central role in Appendix A, where its area ratio is compared to that of the circle and triangle.

4.4 Equilateral triangle (2D)

Definition

An equilateral triangle is a polygon with three sides of equal length and internal angles of 60 degrees. Within the GRM framework, the triangle must be perfectly inscribed in a square, meaning that the triangle's geometry is centrally aligned and symmetrically bounded by the square's edges.

Unlike circles or hexagons, inscribing a triangle in a square introduces directional asymmetry — the triangle touches only three sides of the square. Therefore, only area is considered a stable GRM ratio in this configuration.

GRM Area Ratio for Triangle

Quantity	Value	GRM Unit	Condition
Area	0.4330	SAU	Equilateral triangle inscribed in square

- The GRM area value 0.4330 SAU corresponds to an equilateral triangle whose base spans the full width of the square, with its apex centered on the top edge.
- This ratio is derived geometrically and assumes vertical orientation.

Validation conditions

This GRM ratio is valid only if:

- The triangle is equilateral (all sides equal)
- The triangle is perfectly inscribed in a square of 1.0 SAU
- The base is aligned with the square's bottom edge, and the apex touches the top edge

Other configurations (e.g. rotated or differently inscribed triangles) will yield different area ratios and are not GRM-standardized.

GRM JSON Entry

```
json
{
  "shape_id": "triangle_equilateral",
  "dimension": "2D",
  "geometry": "inscribed_in_square",
  "grm_units": {
    "area": {
      "value": 0.4330,
      "unit": "SAU",
      "tolerance": {
        "range": [0.4280, 0.4380],
        "classification": {
          "perfect": [0.4310, 0.4350],
          "minor_deviation": [0.4280, 0.4310],
          "major_deviation": [0.4350, 0.4380]
        }
      }
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "square_container_required",
    "must_be_equilateral_triangle",
    "base_must_align_with_square_edge"
  ]
}
```

Applications

The GRM triangle is particularly useful in:

- Proportional design systems – using triangle-based layout units
- Deviation classification – testing shape symmetry or distortion
- Pixel-based grid analysis – estimating geometric ratios from triangle renderings
- Educational tools – explaining proportions, ratios, and 2D symmetry in a bounded system

This triangle is also included in comparative ratio charts with the circle and hexagon in *Appendix A*.

4.5 Ellipse (2D)

Definition

An ellipse is a two-dimensional curve defined as the set of points for which the sum of the distances to two fixed foci is constant. In the GRM framework, an ellipse may be inscribed within a square, but unlike the circle, it does not touch all four sides equally unless the radii are equal.

The ellipse is therefore treated as a semi-GRM-compliant shape with ratio approximations that depend on axis symmetry and aspect ratio.

GRM Ratio (Area Approximation)

Quantity	Value (approx.)	GRM Unit	Condition
Area	0.7000–0.7854	SAU	Ellipse inscribed in square (2D), depending on axis ratio

- A perfect circle (equal axes) gives the GRM reference: 0.7854 SAU
- A flattened ellipse (major axis > minor axis) will yield lower values
- Exact ratio depends on the aspect ratio (a/b) of the ellipse

For practical purposes, tolerance zones may be used to classify ellipses as:

- Near-circular (close to 0.7854)
- Elliptical (intermediate ratio)
- Deviating (outside acceptable band)

Validation conditions

To be represented in the GRM dataset:

- The ellipse must be axis-aligned (not rotated)
- It must be fully contained within the square
- The bounding box must be GRM-normalized (1.0 SAU square)

Aspect ratio (a:b) may be included as metadata for classification.

GRM JSON Entry (Example for Ellipse with a:b = 2:1)

```
json
{
  "shape_id": "ellipse_2_1",
  "dimension": "2D",
  "geometry": "inscribed_in_square",
  "grm_units": {
    "area": {
      "value": 0.7000,
      "unit": "SAU",
      "tolerance": {
        "range": [0.6850, 0.7150],
        "classification": {
          "near_circular": [0.7700, 0.7854],
          "elliptical": [0.7000, 0.7700],
          "deviating": [0.6850, 0.7000]
        }
      }
    }
  },
  "conditions": [
    "must_be_fully_contained",
    "must_be_axis_aligned",
    "square_container_required",
    "aspect_ratio_required"
  ],
  "metadata": {
    "aspect_ratio": "2:1"
  }
}
```

Applications

Ellipses are valuable in:

- Shape Recognition – distinguishing between circles and elongated forms
- Deviation analysis – detecting distortion in pixel-based systems
- Proportional UI design – for aesthetics, balance, and layout dynamics
- Medical imaging and segmentation – e.g., approximating anatomical structures

Because ellipses do not fully comply with the core GRM constraints, they are handled as controlled approximations in the JSON structure.

4.6 Cube (3D)

Definition

A cube is a three-dimensional object with six square faces of equal size, twelve equal edges, and right angles at all vertices. In the GRM framework, the cube is treated as a reference shape, functioning as the volumetric container for inscribed 3D forms like the sphere.

Unlike other shapes in this chapter, the cube's GRM ratios are always 1.0000 by definition, as it forms the basis for all Standardized Volume Unit (SVU) comparisons.

GRM Ratios for Cube

Quantity	Value	GRM Unit	Condition
Volume	1.0000	SVU	Reference volume
Surface Area	1.0000	SAU	Reference area (optionally)
Edge Length	0.2500	SPU	One edge, assuming cube perimeter = 1 SPU

- The volume of the cube is defined as 1.0 SVU
- Its perimeter (interpreted as total edge length) is 1.0 SPU, which implies 12 edges of length 1/12 SPU
- The edge length is derived for use in radius-related logic and proportional calculations

---

Validation conditions

The cube as a shape record:

- Serves as a standardized reference, not as a measurable object
- Must be axis-aligned and dimensionally perfect
- Can be used to infer radius, contain inscribed spheres, or serve as a bounding model for 3D validation

---

GRM JSON Entry

```
json
{
  "shape_id": "cube",
  "dimension": "3D",
  "geometry": "reference_shape",
  "grm_units": {
    "volume": {
      "value": 1.0000,
      "unit": "SVU"
    },
    "surface_area": {
      "value": 1.0000,
      "unit": "SAU"
    },
    "edge_length": {
      "value": 0.0833,
      "unit": "SPU"
    }
  },
  "conditions": [
    "must_be_perfect_cube",
    "axis_aligned",
    "used_as_reference"
  ]
}
```

Note: The value 0.0833 SPU for edge length assumes a cube with total edge length (perimeter) of 1.0 SPU, i.e. 1 / 12.

Applications

The cube provides the volumetric foundation for:

- Volume Estimation Without Displacement (serves as container for inscribed sphere)
- Bounding box modeling for 3D recognition
- CAD validation and digital twin construction
- GRM normalization logic in AI and software tools

It is the 3D analogue of the square used in all 2D GRM configurations.

4.7 Cylinder (3D)

Definition

A cylinder is a three-dimensional solid with two parallel circular bases connected by a curved surface. In the GRM framework, a valid cylinder is perfectly inscribed in a cube, such that both circular faces are centered on the top and bottom square faces of the cube, and the curved surface is tangent to the four vertical cube walls.

This configuration allows the GRM to define a fixed volume ratio relative to the cube container, similar to the sphere.

GRM Ratio for Cylinder Volume

Quantity	Value (approx.)	GRM Unit	Condition
Volume	0.7854	SVU	Cylinder inscribed in cube
Radius	0.1250	SPU	Derived from cube perimeter
Height	0.2500	SPU	Cube height = 1/4 of perimeter

- The volume ratio is equal to that of a circle:

  $V = \pi r^2 h$ → GRM approximation: $V = 0.7854 \times \text{height}$

- With height = edge length of the cube, the final volume becomes 0.7854 SVU
- Radius is same as for the circle: 0.1250 SPU

Validation Conditions

The GRM ratio applies if:

- The cylinder is perfectly inscribed in a cube of volume 1.0 SVU
- The circular faces are aligned with the top and bottom cube faces
- The curved surface is tangent to all four vertical cube sides
- The base circle follows the same radius rule as in 2D GRM

GRM JSON Entry

```
json
{
  "shape_id": "cylinder",
  "dimension": "3D",
  "geometry": "inscribed_in_cube",
  "grm_units": {
    "volume": {
      "value": 0.7854,
      "unit": "SVU",
      "tolerance": {
        "range": [0.7800, 0.7900],
        "classification": {
          "perfect": [0.7845, 0.7865],
          "minor_deviation": [0.7800, 0.7845],
          "major_deviation": [0.7865, 0.7900]
        }
      }
    },
    "radius": {
      "value": 0.1250,
      "unit": "SPU"
    },
    "height": {
      "value": 0.2500,
      "unit": "SPU"
    }
  },
  "conditions": [
    "must_be_perfectly_inscribed",
    "axis_aligned",
    "circular_faces_touch_cube_top_bottom"
  ]
}
```

Applications

Cylinders are relevant for:

- Volume Estimation Without Displacement – offering an intermediate between sphere and cube
- Engineering models – pipes, containers, and structural elements
- Shape Recognition in 3D scanning – validating cylindrical objects
- Parametric design – consistent ratio modeling in CAD

4.8 Composite shapes (Multi-Element Forms)

Definition

A composite shape refers to any geometric configuration consisting of two or more GRM-compliant base shapes combined through operations such as stacking, subtracting, adjoining, or nesting.

These shapes do not conform to a single fixed GRM ratio themselves, but they can be analyzed by decomposing them into their constituent parts and applying GRM ratios individually. The result is a derived or inferred GRM value based on logical combination.

GRM Ratio Handling in Composite Shapes

Strategy	Approach
Additive model	Sum of the GRM volumes/areas of multiple parts
Subtractive model	Subtract one or more internal shapes from a container
Nested model	Contained shapes within a GRM-standard envelope
Tiled model	Repeating GRM base units across a defined area or volume

Each part must retain **its own GRM compliance,** and the overall result is computed from those individual contributions.

Example: Square with Circular Cutout

Component	Ratio Value	GRM Unit	Role
Square (base)	1.0000	SAU	Container
Circle (cutout)	0.7854	SAU	Subtracted

Effective area = 1.0000 – 0.7854 = 0.2146 SAU

GRM JSON Entry (Composite: Square minus Circle)

```
json
{
  "shape_id": "square_minus_circle",
  "dimension": "2D",
  "geometry": "composite",
  "composite_structure": {
    "type": "subtractive",
    "base_shape": {
      "shape_id": "square",
      "grm_units": {
        "area": {
          "value": 1.0000,
          "unit": "SAU"
        }
      }
    },
    "subtracted_shape": {
      "shape_id": "circle",
      "grm_units": {
        "area": {
          "value": 0.7854,
          "unit": "SAU"
        }
      }
    }
  },
  "result": {
    "area": {
      "value": 0.2146,
      "unit": "SAU"
    }
  },
  "conditions": [
    "all_parts_must_be_grm_compliant",
    "axis_alignment_required"
  ]
}
```

Applications

Composite shapes are especially relevant for:

- Design and architecture – subtractive and modular composition
- CAD and simulation models – modeling voids, holes, intersections
- Shape recognition via decomposition – AI-based classification of compound geometry
- Mathematics and education – illustrating area/volume through logical breakdown

They also enable approximate GRM validation for non-standard or organic shapes, using GRM-defined primitives.

***Note**: In GRM-based systems, composite structures are considered derived constructions, not primary GRM shapes. They are supported in JSON via the composite structure field and can be evaluated recursively.*

5 . Classification and Tolerance Modeling

Introduction

This chapter defines how the GRM framework handles imperfections, deviations, and classification of measured or observed shapes relative to ideal GRM values. While the geometric ratios in GRM are fixed and dimensionally absolute, real-world applications, such as AI recognition, CAD validation, or scanned input, often yield approximate results.

To accommodate this, GRM introduces a programmable structure for:

- Tolerance ranges around each ideal ratio
- Deviation classification labels (e.g. perfect, minor, major)
- Confidence scoring (optional)
- Validation logic within the JSON schema

These mechanisms ensure that GRM-based systems can distinguish between:

- Shapes that are fully compliant
- Shapes that are acceptable but imperfect
- Shapes that are invalid within the GRM scope

5.1 Tolerance range structure

Each GRM ratio (e.g. perimeter, area, volume) may be accompanied by a tolerance range, which defines the lower and upper bounds within which a measurement is still considered acceptable.

Format in JSON:

```
json
"tolerance": {
  "range": [min_value, max_value]
}
```

Example for a circle perimeter: [0.7800, 0.7900]

This defines a validation envelope around the ideal GRM ratio.

5.2 Deviation classification labels

To further refine interpretation, GRM tolerance ranges can be subdivided into qualitative deviation bands:

Label	Description
perfect	Within a very narrow band of the ideal value
minor_deviation	Slight deviation, still acceptable
major_deviation	Close to the edge of tolerance or questionable

Format in JSON:

```
json
"classification": {
  "perfect": [0.7845, 0.7865],
  "minor_deviation": [0.7800, 0.7845],
  "major_deviation": [0.7865, 0.7900]
}
```

This system supports conditional logic, tagging, or feedback in applications.

5.3 Confidence Scoring (Optional)

GRM allows implementers to assign a confidence score **(**0.0–1.0) based on the proximity of a measured value to the ideal.

This can be derived by:

- Normalizing the deviation relative to the full tolerance range
- Inverting the deviation to score closeness

Format:

```
json
"confidence_score": 0.92
```

Use of confidence scoring is optional and application-specific.

5.4 Validation logic

To validate a shape against GRM criteria:

1. Compare the measured value to the value in the GRM record.
2. If outside the tolerance.range, the shape is non-compliant.
3. If within, classify using the classification bands.
4. Optionally, calculate a confidence score for reporting or decision-making.

5.5 Use cases

These validation mechanisms are essential in:

- AI shape recognition systems
- CAD model verification
- Quality control and inspection software
- Education and feedback tools
- GRM-powered assistant prompts (e.g. GPTs)

6. Application Mapping of GRM Proposals

Introduction

This chapter connects the original GRM application proposals to their corresponding JSON data representations. Each application translates specific aspects of GRM theory into programmable logic, enabling use in AI, CAD, education, image analysis, and simulation environments.

The mapping is structured around five core proposals:

1. Pixel-Based Ratio Measurement
2. Shape Recognition with Tolerance Classification
3. Proportional Design with GRM

4. Volume Estimation Without Displacement
5. Radius Modeling and Derivation

For each, we describe:

- The purpose and use case
- The relevant GRM units and conditions
- How it is encoded in JSON
- Where it integrates with the schema from Chapter 3

6.1 Pixel-based ratio measurement

Purpose:
Approximate perimeter, area, or radius of digital shapes using pixel counts and compare them against GRM references.

JSON Extensions:

```
json
"pixel_metrics": {
  "pixel_area": 183,
  "pixel_perimeter": 52,
  "pixel_radius": 8.0
}
```

Logic:

- Convert pixel-based values to GRM ratios by normalizing over pixel grid size
- Compare results to GRM constants (e.g., 0.7854 SPU)
- Validate using tolerance structure from Chapter 5

6.2 Shape recognition with tolerance classification

Purpose:
Classify a scanned or detected shape as *GRM-conformant* based on how closely its measured ratios match expected GRM values.

JSON Extensions:

```
json
"recognition": {
  "measured_perimeter": 0.7831,
  "deviation": -0.0023,
  "classification": "minor_deviation",
  "confidence_score": 0.91
}
```

Logic:

- Measure value → compare with grm_units.value
- Classify using tolerance.classification
- Score optionally using confidence_score

6.3 Proportional design with GRM

Purpose:
Use GRM ratios as design anchors for spacing, sizing, and layout in GUIs, mechanical systems, or learning materials.

JSON Extensions:

```
json
"design_guides": {
  "module_unit": "circle",
  "ratio_reference": "SAU",
  "scaling_pattern": "0.7854 × n",
  "layout_alignment": "centered"
}
```

Logic:

- Derive layout proportions from GRM units (e.g., circles = 0.7854 SAU)
- Use repeating or nested patterns (e.g., tiled hexagons or concentric rings)
- Validate design consistency using base units

6.4 Volume estimation without displacement

Purpose:
Estimate the volume of a 3D object using GRM ratios and standardized cube reference, without needing physical measurement.

JSON Extensions:

```
json
"volume_estimation": {
  "method": "GRM_fixed_ratio",
  "shape": "sphere",
  "volume_ratio": 0.5236,
  "estimated_volume": 0.5236
}
```

Logic:

- Use known GRM volume (e.g., sphere = 0.5236 SVU)
- Infer real-world volume by scaling relative to container
- Validate or approximate deviations using tolerance ranges

6.5 Radius Modeling and Derivation

Purpose:
Use the derived radius (r = 0.1250 SPU) as a programmable unit in calculations, layout generation, or pixel conversions.

JSON Extensions:

```
json
"radius_model": {
  "derived_radius": 0.1250,
  "unit": "SPU",
  "used_in": ["circle", "sphere", "cylinder"]
}
```

Logic:

- Apply fixed ratio for radius in all inscribed circle/sphere calculations
- Use for graphical representation or scaling
- May be validated or estimated indirectly via perimeter or area

7. Sample Use Cases

Introduction

This chapter demonstrates how the GRM JSON structure can be applied in practical scenarios. Each use case shows how shape data, ratios, and classifications are used in real-world systems such as AI models, CAD software, educational platforms, and verification tools.

The examples include:

1. Shape validation from raw input
2. AI prompt with embedded GRM logic
3. CAD-based layout validation
4. Pixel-based shape recognition
5. Educational geometry module

7.1 Shape validation from raw input (e.g. via sensors)

Scenario:
A system receives a scanned 2D shape and computes its perimeter as 0.7831 SPU.

JSON Input:

```
json
{
  "input_shape": "circle",
  "measured_perimeter": 0.7831,
  "reference_value": 0.7854,
  "tolerance_range": [0.7800, 0.7900]
}
```

System Output:

```
json

{
  "shape_id": "circle",
  "classification": "minor_deviation",
  "deviation": -0.0023,
  "confidence_score": 0.91
}
```

7.2 AI prompt-based shape checker (e.g. GPT agent)

Prompt:

“Can you check if this 2D shape with a perimeter of 0.7850 is a GRM-compliant circle?”

Expected Agent Logic:

- Compare 0.7850 to GRM baseline 0.7854 SPU
- Deviation = –0.0004
- Output: perfect match within tolerance

7.3 CAD Design Validation (Proportional Layouts)

Scenario:
A CAD module checks if components follow GRM-based proportions.

JSON Snippet:

```
json
{
  "layout_unit": "hexagon",
  "expected_area": 0.9014,
  "measured_area": 0.8990,
  "classification": "perfect"
}
```

Interpretation:
Design matches GRM hexagon within perfect tolerance range.

7.4 Pixel-based shape recognition

Scenario:
A camera system detects an object with:

- 183 pixels in area
- Grid width = 192 pixels (bounding box)

Calculation:

- Normalized area ≈ 0.783
- Matches circle within minor_deviation range

Output:

```
json
{
  "shape_id": "circle",
  "pixel_area": 183,
  "normalized_ratio": 0.783,
  "classification": "minor_deviation"
}
```

7.5 Educational geometry tool

Scenario:
A teaching module asks students to calculate the area of an inscribed triangle.

Expected Result:

- shape_id = triangle_equilateral
- area = 0.4330 SAU

System Response:
If student enters 0.432:

```
json
{
  "feedback": "Correct within perfect range. Well
done!",
  "deviation": -0.0010,
  "confidence_score": 0.96
}
```

Chapter 8 – Integration guidelines

Introduction

This chapter provides best practices and structural recommendations for integrating the GRM JSON framework into software systems, APIs, AI pipelines, CAD tools, and educational environments.

The goal is to ensure:

- Correct use of GRM ratios and conditions
- Traceable and auditable logic
- Future-proof extensibility
- Cross-system compatibility

8.1 Data access and parsing

Recommendation:
Use standard JSON parsers (available in all major programming languages) to load GRM shape definitions into memory. Each shape record is self-contained and can be identified by its shape_id.

Example (Python):

```
python

import json
with open("grm_shapes.json") as f:
    data = json.load(f)
circle = data["circle"]
```

8.2 Validation flow for input shapes

For any measured or detected shape:

1. Select GRM reference by shape_id
2. Compare measured value (e.g. area or perimeter) to grm_units.value
3. Use tolerance.range to verify compliance
4. Use classification for semantic feedback
5. Optionally assign confidence_score

8.3 Modular integration in AI pipelines

Use cases:

- Prompt-injected GRM validation
- Auto-classification during visual inference
- Ratio-based geometric reasoning

Tip:
Embed the relevant GRM JSON record inside the model's prompt or knowledge base, or call it dynamically at runtime.

8.4 Embedding in CAD and layout systems

- Use GRM units as parametric design constraints
- Define grid snapping or modular units based on SPU, SAU, SVU
- Apply ratio validation on shape libraries (e.g. checking if all holes are 0.7854 SAU)
- Annotate model components with shape_id, grm_units, and conditions

8.5 Educational tools and dynamic interfaces

- Use GRM ratios as fixed anchors for visual learning
- Allow toggling of tolerance bands for real-time feedback
- Show deviations or confidence visually using colors or bars
- Combine multiple shapes using composite_structure logic

8.6 Error handling and edge cases

Scenario	Recommendation
Shape falls outside tolerance	Tag as non_compliant or invalid
Missing classification	Fallback to tolerance range only
Incomplete composite shape	Require all subcomponents to be GRM-valid
Invalid units or structure	Validate against JSON schema or fallback default values

8.7 Version control and referencing

Each GRM dataset or shape record should include:

- version: the GRM version used (e.g. "1.0")
- source: reference to handbook or whitepaper
- Optional: license, author, date_generated

This ensures consistency and attribution across applications.

9. Licensing and Attribution

9.1 Intellectual property statement

The Geometric Ratio Model (GRM), including all associated constants, standardized units (SPU, SAU, SVU), JSON schema structures, and derived models (e.g. radius, tolerance classification), is an original theoretical model developed by:

M.C.M. van Kroonenburgh, MSc

The model is officially registered under i-Depot number 151927 at the Benelux Office for Intellectual Property (BOIP) and is protected under intellectual property law.

9.2 Licensing conditions

The use of the GRM JSON structure, shape definitions, and derived logic is subject to the conditions outlined in the GRM Open Usage Framework v1.0GRM Open Usage Framewor....

License Types:

Use Case Type	Permission Required?	Conditions
Educational use	❌ No	Free use with attribution
Scientific exploration	❌ No	Open use with citation
Non-commercial AI tools	❌ No	Allowed with source reference
Commercial software/tools	✅ Yes	License + permission required
API/SaaS integration	✅ Yes	Attribution + license required
LLM training or embedding	✅ Yes	Prior written approval required
Derivative works	Case-by-case	Must preserve core logic and cite source

9.3 Attribution requirements

Any use of GRM in published materials, software, user interfaces, or technical documentation must include the following attribution:

"This tool/method uses the Geometric Ratio Model (GRM), developed by M.C.M. van Kroonenburgh, MSc – https://inratios.com"

Attribution must be visible in:

- Documentation and metadata
- Software About/Info pages or tooltips
- Academic publications or whitepapers

9.4 Disclaimer of applicability

GRM is a theoretical geometric framework. It is not a substitute for:

- Certified physical measurement standards
- Legal metrology
- Medical or engineering compliance systems

All applications using GRM must ensure proper contextual use, particularly when used in professional domains.

9.5 Contact for licensing or collaboration

For licensing inquiries, commercial partnerships, or derivative application approval, please contact:

📧 info@inratios.com
🌐 https://inratios.com

9.6 Open License (Creative Commons)

Unless otherwise stated, this handbook and the GRM JSON schema are licensed under:

Creative Commons Attribution–NonCommercial–ShareAlike 4.0 International
(CC BY-NC-SA 4.0)
🔗 https://creativecommons.org/licenses/by-nc-sa/4.0/

You are free to:

- Share – copy and redistribute the material in any medium or format
- Adapt – remix, transform, and build upon the material

Under the following terms:

- Attribution – You must give appropriate credit to the author.
- NonCommercial – You may not use the material for commercial purposes.
- ShareAlike – If you remix or transform the material, you must distribute your contributions under the same license.

Appendix A – GRM Ratio Table

Overview

This appendix presents a consolidated table of all fixed geometric ratios defined within the Geometric Ratio Model (GRM), categorized by shape and dimension. These ratios represent the core measurable quantities (perimeter, area, and volume) expressed in standardized units (SPU, SAU, SVU) under specific geometric conditions (e.g., perfect inscription).

Each entry includes:

- The dimensional scope of the shape (2D or 3D)
- The relevant GRM value(s) for each quantity
- The corresponding standardized unit
- Any associated radius (if applicable)
- A summary of the geometric condition
- Notes on tolerances, approximations, or constraints

This table serves as a reference for:

- JSON construction (Chapter 3)
- Shape validation (Chapter 5)
- Application mapping (Chapter 6)
- Educational or computational tools

All values are normalized relative to a reference square or cube with unit perimeter, area, or volume (i.e., 1.0 SPU, SAU, or SVU).

Table A.1 – Fixed GRM Ratios for Standard and Composite Shapes

Shape	Dim.	Quantity	Value	Unit	Radius	Condition	Notes
Circle	2D	Perimeter	0.7854	SPU	0.1250	Inscribed in square	Core shape
		Area	0.7854	SAU	—		
Sphere	3D	Volume	0.5236	SVU	0.1250	Inscribed in cube	See WP IV
Hexagon	2D	Area	0.9014	SAU	—	Inscribed in square	Regular only
Triangle (Equilateral)	2D	Area	0.4330	SAU	—	Inscribed in square	Aligned base required
Ellipse (2:1)	2D	Area (est.)	0.7000	SAU	—	Inscribed, axis-aligned	Approximate
Cylinder	3D	Volume	0.7854	SVU	0.1250	Axis-aligned in cube	Same r as circle
		Height	0.2500	SPU	—		Cube edge assumed
Cube	3D	Volume	1.0000	SVU	—	Reference shape	
		Surface Area	1.0000	SAU	—		
		Edge Length	0.0833	SPU	—		1 / 12 perimeter
Composite (Sq–Ci)	2D	Area	0.2146	SAU	—	Subtractive composition	1.0 – 0.7854

Appendix B – Reference Formulas and GRM Derivations

Purpose

This appendix documents the reference formulas from classical geometry that underlie the GRM fixed ratios. It also shows how these formulas are converted to GRM units (SPU, SAU, SVU), often by removing π or replacing it with GRM-compliant constants. This provides a traceable link between traditional measurement and GRM logic.

Tabel B.1 – Classical vs. GRM-Adjusted Formulas

Shape	Quantity	Classical Formula	GRM Derivation (SPU/SAU/SVU)	Notes
Circle	Perimeter	$2\pi r$	$8r = 0.7854 \cdot s$	$For\ r = 0.1250s$
Circle	Area	πr^2	$64r^2 = 0.7854 \cdot s^2$	Assuming square perimeter $s = 1$
Sphere	Volume	$\frac{4}{3}\pi r^3$	$512r^3 = 0.5236 \cdot s^3$	r = 0.125s, cube edge as base
Cylinder	Volume	$\pi r^2 h$	$64r^2h = 0.7854 \cdot s^3$	For full-height cube
Hexagon	Area	$\frac{3\sqrt{3}}{2}a^2$	~0.9014 (numerical)	Regular, inscribed in square
Triangle	Area	$\frac{1}{2}bh$	$0.4330 \cdot s^2$	For equilateral triangle in square
Cube	Volume	a^3	1.0000 SVU	Reference shape
Ellipse	Area (approx.)	πab	Varies with axis ratio	GRM range: 0.7000–0.7854

- **Circle:**
  The substitution $r = \frac{1}{8} \cdot s$ simplifies the perimeter to $8r = s \cdot 0.7854$, making π obsolete in this formulation. Same logic applies for area.
- **Sphere:**
  Given a cube of perimeter 1.0 SPU, the derived $r = 0.1250 \cdot s$ results in the GRM volume ratio $0.5236 \cdot s^3$
- **Cylinder:**
  If the height equals the cube's edge (i.e. $h = 0.25 \cdot s$), and radius is $0.125s$, the GRM-compliant volume becomes $0.7854 \cdot s^3$.

- **Hexagon, Triangle, Ellipse:**
  These use standard area formulas, numerically evaluated under square constraints. For the hexagon and triangle, their shapes are symmetric and regular; for the ellipse, the ratio depends on the major/minor axes and is not exact.

B.2 Notes on Ratio Derivations and Assumptions

- **Circle**

  The circle's classical formulas for perimeter and area depend on π and radius. In GRM, π is eliminated by fixing the radius as a proportion of the square's perimeter:

  - $r = 0.125 \cdot s$
  - Perimeter becomes $2\pi r \rightarrow 8r = 0.7854 \cdot s$
  - Area becomes $\pi r^2 \rightarrow 64r^2 = 0.7854 \cdot s^2$

- **Sphere**

  The volume of a sphere is classically defined as $\frac{4}{3}\pi r^3$.By using a fixed radius inside a unit cube (SPU = 1), GRM derives:

  - $r = 0.1250 \cdot s$
  - Volume becomes $\frac{4}{3}\pi r^3 \rightarrow 512r^3 = 0.5236 \cdot s^3$

- **Cylinder**

  A GRM-aligned cylinder fits inside the cube, with:

  - Radius: $r = 0.1250 \cdot s$
  - Height: $h = 0.25 \cdot s$
  - Volume becomes $\pi r^2 h \rightarrow 64r^2 h = 0.7854 \cdot s^3$

- **Hexagon**

  The area of a regular hexagon is classically $\frac{3\sqrt{3}}{2}a^2$. When numerically evaluated as a hexagon inscribed in a square, this yields:

  - GRM area ≈ 0.9014 SAU
  - Ratio is accepted as fixed for regular inscribed hexagons.

- **Triangle (Equilateral)**

  The area of an equilateral triangle is derived from:

  - $A = \frac{1}{2}$bh
  - In GRM, inscribed in a square, this gives $A \approx 0.4330 \cdot s^2$

- **Ellipse**

  The classical area is πab, with a and b the semi-axes. GRM treats the ellipse as an approximate shape:

  - Area ≈ 0.7000–0.7854 SAU depending on aspect ratio
  - Not a fixed GRM ratio but tolerated in structured classification

Appendix C – GRM JSON Schema Tree

Purpose

This appendix presents the structural overview of the GRM JSON schema as a hierarchical tree. It serves as a technical reference for developers and toolbuilders to validate, parse, or extend GRM-compliant JSON datasets.

It complements the schema field definitions from Chapter 3 and shows the logical nesting, data types, and required/optional fields.

Notes

- Fields are modular: not all records include all blocks (e.g. pixel data is optional)
- Schema supports composite and derived shapes (via composite_structure)
- Use in combination with tolerance rules from Chapter 5
- Matches handbook structure for chapters 3 (schema), 6 (applications), and 4 (shapes)

(See next page)


```
GRM Shape Record (root object)
├── shape_id (string) *                # Unique ID, e.g. "circle"
├── dimension (string) *               # "2D", "3D"
├── geometry (string) *                # e.g. "inscribed_in_square"
├── grm_units (object) *               # Holds core ratios
│   ├── perimeter (object) [optional]
│   │   ├── value (number) *
│   │   ├── unit (string) *
│   │   ├── tolerance (object) [optional]
│   │   │   ├── range [min, max] (array)
│   │   │   ├── classification (object)
│   │   │   │   ├── perfect (array)
│   │   │   │   ├── minor_deviation (array)
│   │   │   │   ├── major_deviation (array)
│   ├── area (object) [optional]
│   │   ├── value (number)
│   │   ├── unit (string)
│   │   ├── tolerance (object) [optional] {same structure}
│   ├── volume (object) [optional]
│   │   ├── value (number)
│   │   ├── unit (string)
│   │   ├── tolerance (object) [optional] {same structure}
│   ├── radius (object) [optional]
│   │   ├── value (number)
│   │   ├── unit (string)
│   ├── surface_area (object) [optional]
│   │   ├── value (number)
│   │   ├── unit (string)
│   ├── edge_length (object) [optional]
│   │   ├── value (number)
│   │   ├── unit (string)
│
├── conditions (array of strings) *    # e.g. ["must_be_perfectly_inscribed"]
├── recognition (object) [optional]    # For shape classification
│   ├── measured_perimeter (number)
│   ├── deviation (number)
│   ├── classification (string)
│   ├── confidence_score (number)
├── pixel_metrics (object) [optional]  # For pixel-based ratio estimation
│   ├── pixel_area (number)
│   ├── pixel_perimeter (number)
│   ├── pixel_radius (number)
├── design_guides (object) [optional]  # For proportional design use
│   ├── module_unit (string)
│   ├── ratio_reference (string)
│   ├── scaling_pattern (string)
│   ├── layout_alignment (string)
├── volume_estimation (object) [optional] # For estimating object volumes
│   ├── method (string)
│   ├── shape (string)
│   ├── volume_ratio (number)
│   ├── estimated_volume (number)
├── radius_model (object) [optional]
│   ├── derived_radius (number)
│   ├── unit (string)
│   ├── used_in (array of strings)
├── composite_structure (object) [optional] # For additive or subtractive models
│   ├── type (string)                  # "additive", "subtractive", etc.
│   ├── base_shape (object)
│   ├── subtracted_shape (object)
│   ├── result (object)
│   │   ├── area (number) or volume (number)
├── metadata (object) [optional]       # Traceability and versioning
│   ├── version (string)
│   ├── author (string)
│   ├── source (string)
│   ├── license (string)
```