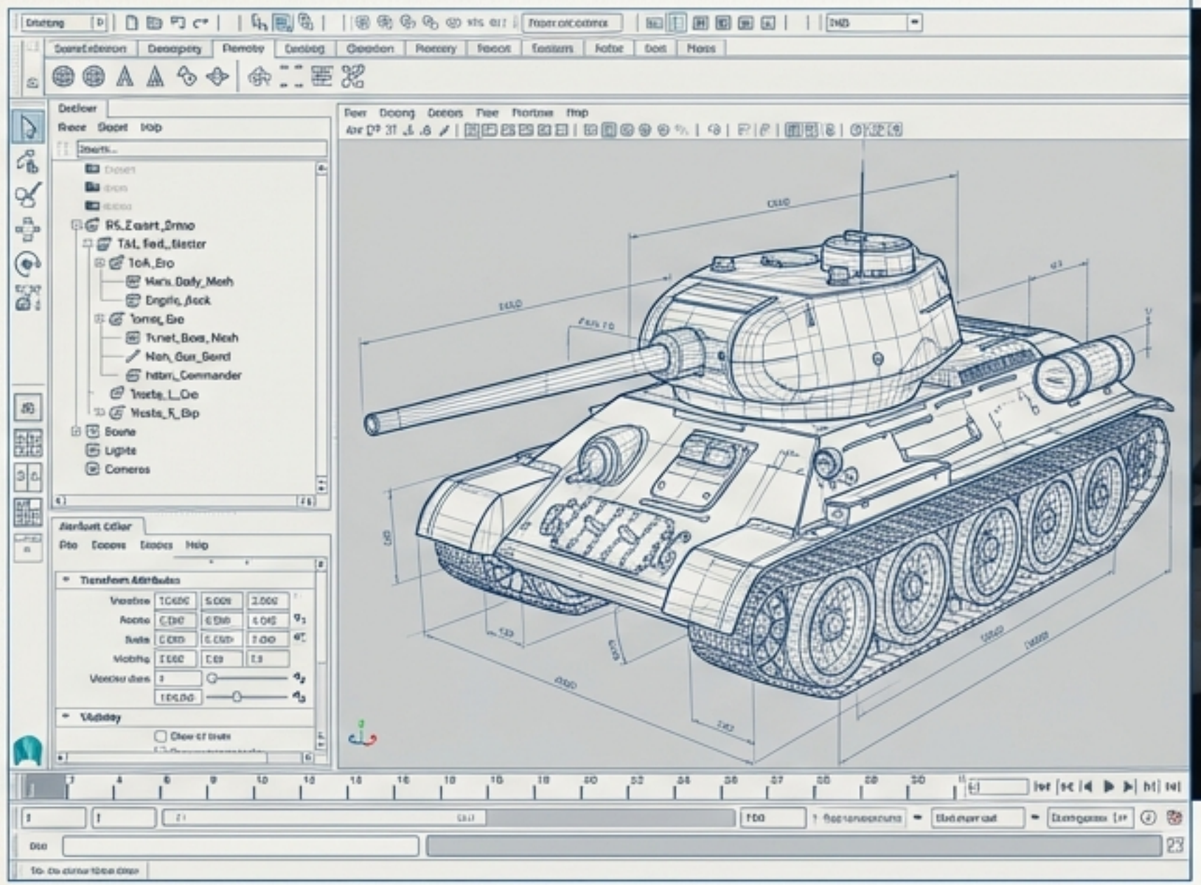
Mastering the G5 Pipeline: A Technical Artist's Guide to Exporting from Maya 5

From Maya Scene to Game-Ready Asset. A definitive guide to the T-34 vs. Tiger export workflow.



G5 PIPELINE VERIFICATION

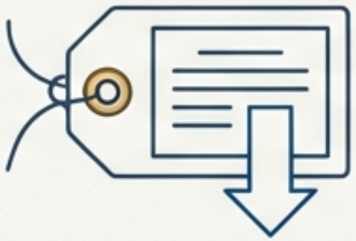
This deck provides the verified, step-by-step process for converting Maya 5.0 assets into the proprietary G5 engine format. It is a practical guide based on the official technical manual (v3.1).

The Three Pillars of the G5 Pipeline

Attribute-Driven

The engine is controlled by custom attributes added to Maya objects

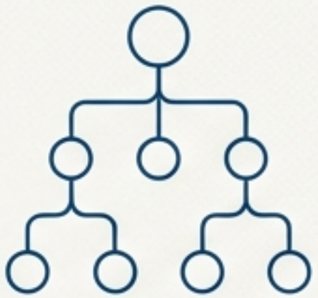
The exporter doesn't guess intent. All engine-specific behaviours—from collision to rendering—are **explicitly defined** by attributes you set within Maya's Attribute Editor. Mastering these attributes is mastering the pipeline.



Convention over Configuration

Naming conventions are mandatory and trigger specific processes

The pipeline uses specific naming suffixes and prefixes (e.g., `_1m`, `_col`, `_LOD#`) to automatically identify and process geometry. Correct naming is not a suggestion; it is a requirement.



Hierarchy is Law

The Maya Outliner structure directly translates to the in-game object hierarchy.

How you parent and group objects How you parent and group objects in your Maya scene defines their relationship and structure within the G5 engine. A clean Outliner is **essential for a functional asset.**

Section 1.0: Environment Configuration.

System Requirements

The pipeline requires **Maya 5.0** specifically. The `MayaExp.mll` plugin and MEL scripts are not compatible with later versions.

Using **Maya 6.0** or later will result in script errors or unpredictable behaviour.

Script Installation

Copy the entire 'Tools' folder content into your Maya scripts directory.

```
C:/Documents and Settings/[Username]/My Documents/maya/5.0/scripts/
```

Plugin Installation

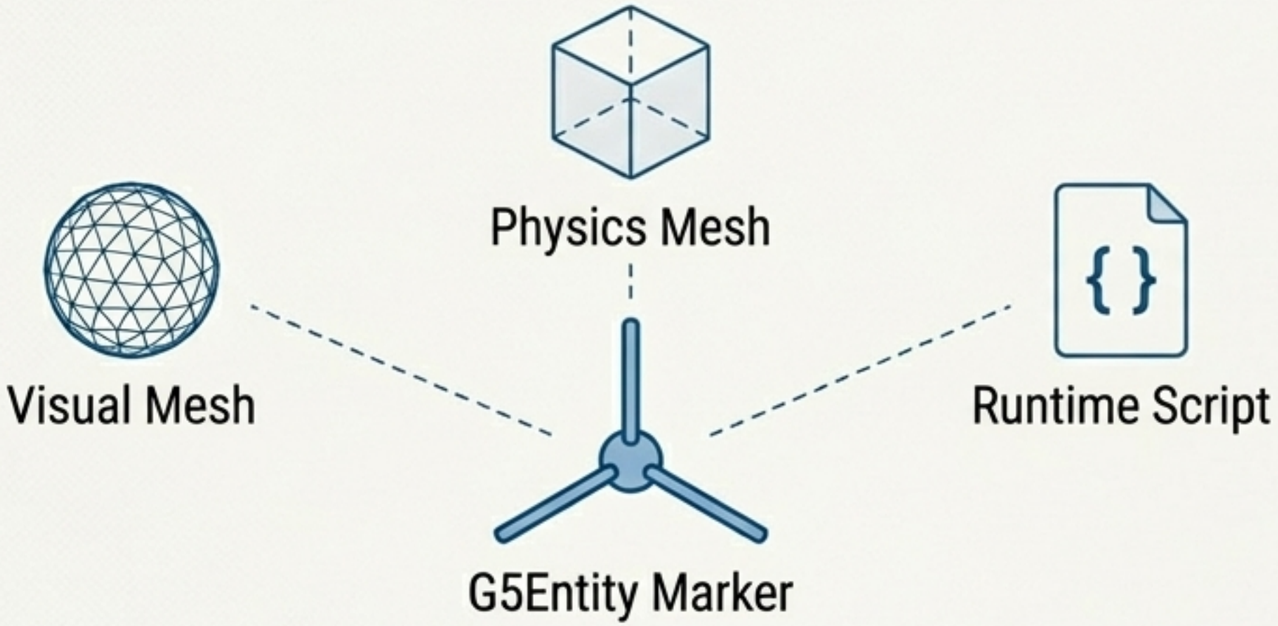
Copy `MayaExp.mll` to the Maya plugins folder and run `InitMayaExp.bat` to register the required `libguide40.dll` library.

Initialisation

On first launch, execute the `G5Exp.mel` script to create the **G5Engine** shelf and load all custom commands. A successful installation is confirmed by the appearance of the shelf.

The G5Entity: An Asset's Core Identity.

The `G5Entity` is a locator node that serves as the organizational backbone for every game object. It is the root of the asset hierarchy.



The `ClassName` Attribute

The `ClassName` attribute on the G5Entity links the asset to a runtime script file (a `.txt` file). This script defines the object's behaviour in the game.
A mismatch or typo in this name is the most common reason for an object appearing in-game with no functionality.

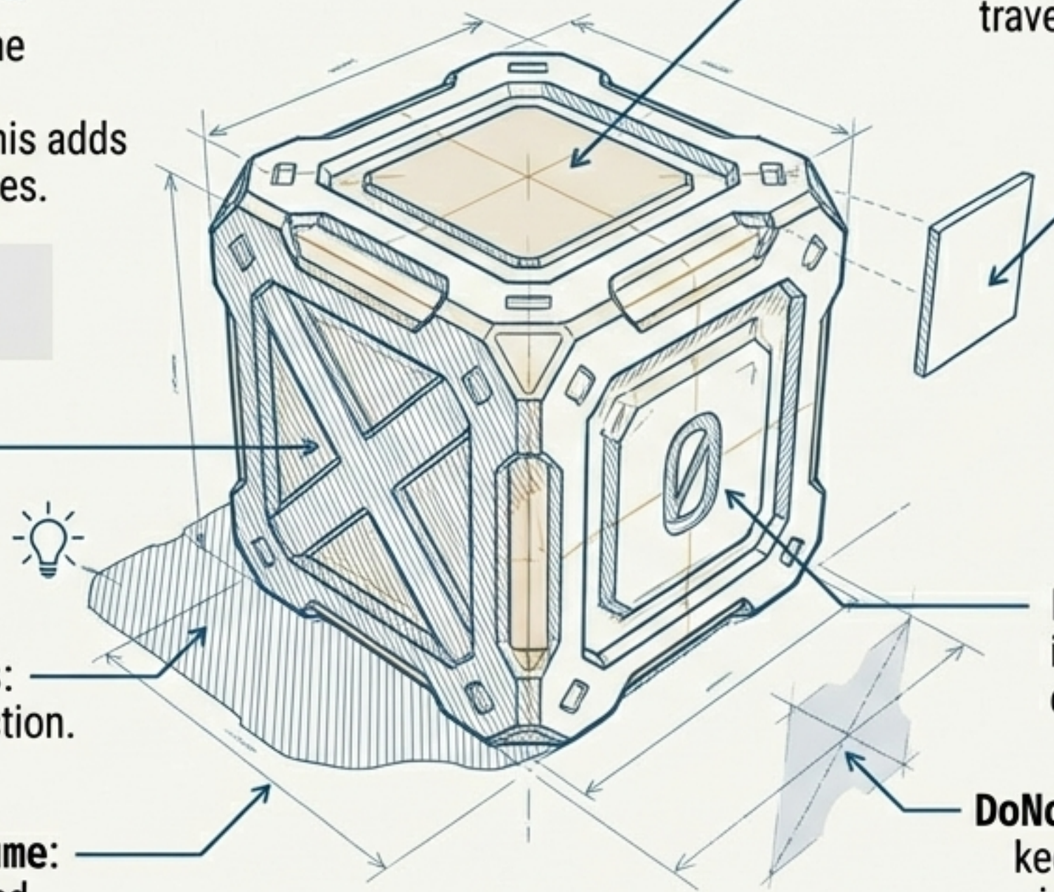
The `createG5Entity` command mentioned in project documentation was not found in the provided MEL scripts. Please consult your project lead for the current, correct method of creating G5Entity markers.

Anatomy of a G5 Mesh

The Foundational Command

Every mesh destined for the G5 engine must first be processed with the **addMeshProperties** command. This adds all necessary engine-specific attributes.

```
addMeshProperties;
```



IsWalkMesh: Marks this surface as traversable for character navigation.

IsBillboard: Renders the mesh as a 2D quad that always faces the camera.

IsCollisionMesh: Enables physics collision. Can be combined with **IsWalkMesh**.

LodNumber: The Level of Detail index for this mesh (0 is highest detail).

CastShadows / ReceiveShadows: Standard controls for shadow interaction.

DoNotExport: The essential flag to keep helper or reference geometry in the scene without exporting it.

IsShadowMesh / HasShadowVolume: Designates meshes for the specialised shadow volume rendering technique.

The Blueprint for Collision: Creating Physics Shapes.

Concept: For performance, the G5 engine uses simplified, non-rendered geometry for physics calculations. You must generate these shapes from your visual meshes.

The Generation Command: Use the `createPhysicsShape` command with a numeric parameter to generate the desired collider type. The new shape will be parented to the original mesh.

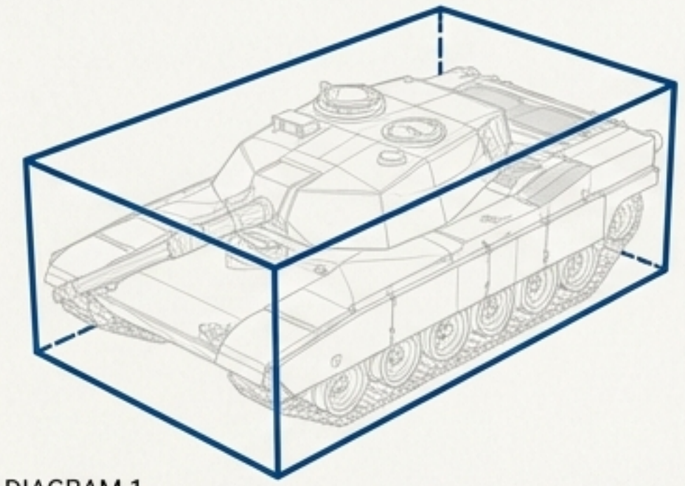


DIAGRAM 1

```
createPhysicsShape(1); // Box
```

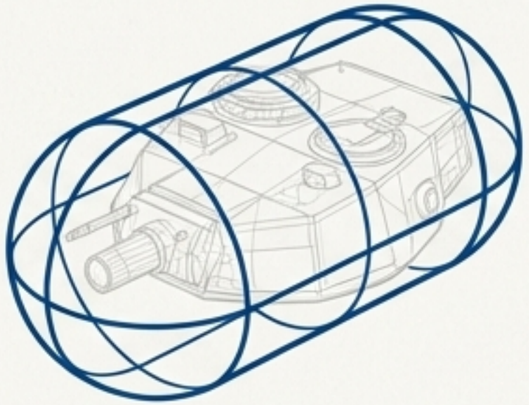


DIAGRAM 2

```
createPhysicsShape(2); // Capsule
```

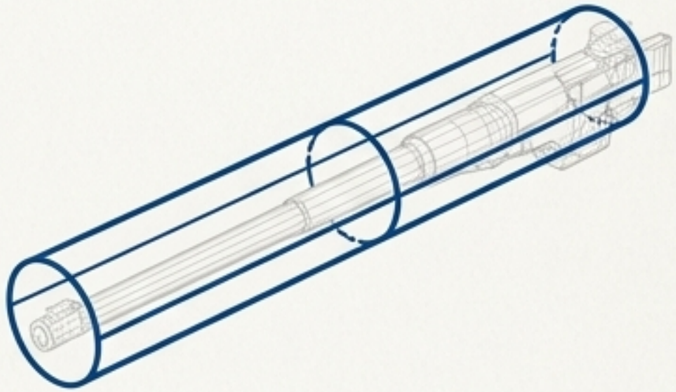
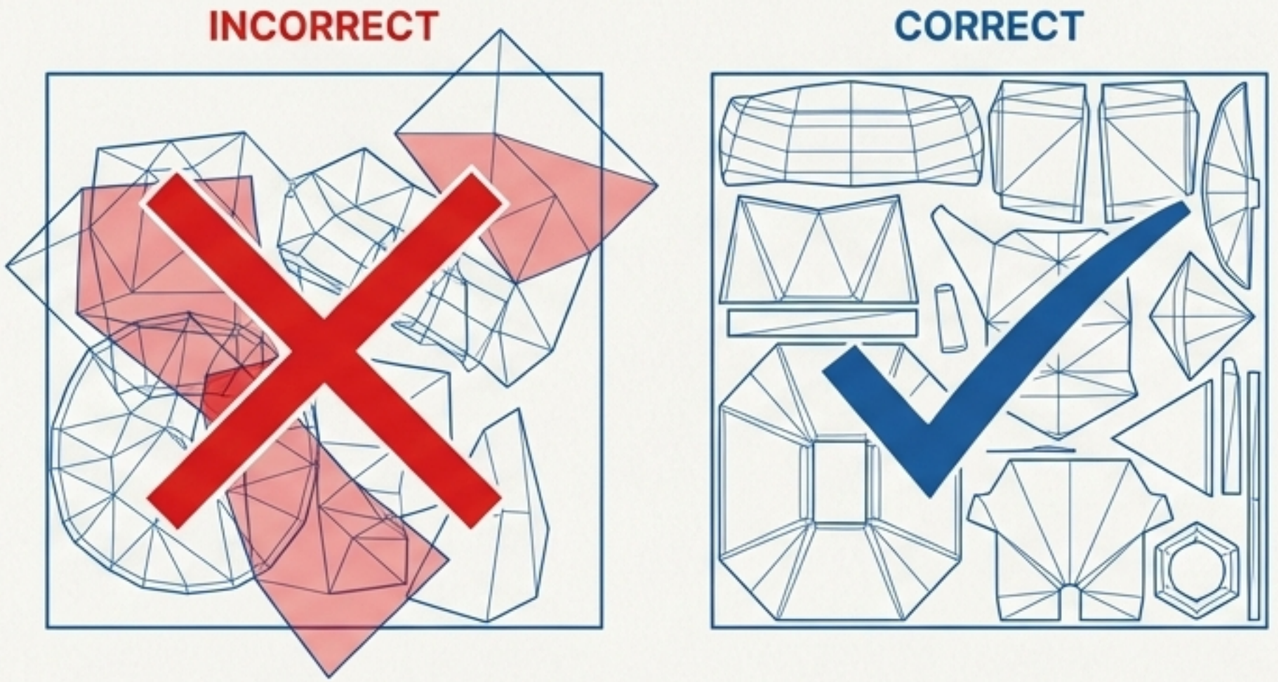


DIAGRAM 3

```
createPhysicsShape(3); // Cylinder
```

Key Attributes: The generated mesh is automatically assigned the **`IsPhysicsShape`** (boolean) and **`ShapeType`** (integer) attributes to identify it as a physics-only object.

Engineering Light Maps: Strict UV Requirements.



Requirement 1: Naming Convention

To be included in the light map generation process, a mesh name must end with the `_1m` suffix.

Example: `TankHull_1m`

Requirement 2: No Overlapping UV Faces

Every face in the light map UV set must occupy a unique space. Overlaps will cause light to bleed incorrectly between surfaces.

Requirement 3: Confine to 0-1 Texture Space

The entire UV layout must fit within the 0 to 1 texture coordinate space. Any UVs outside this range are invalid and will fail the process.

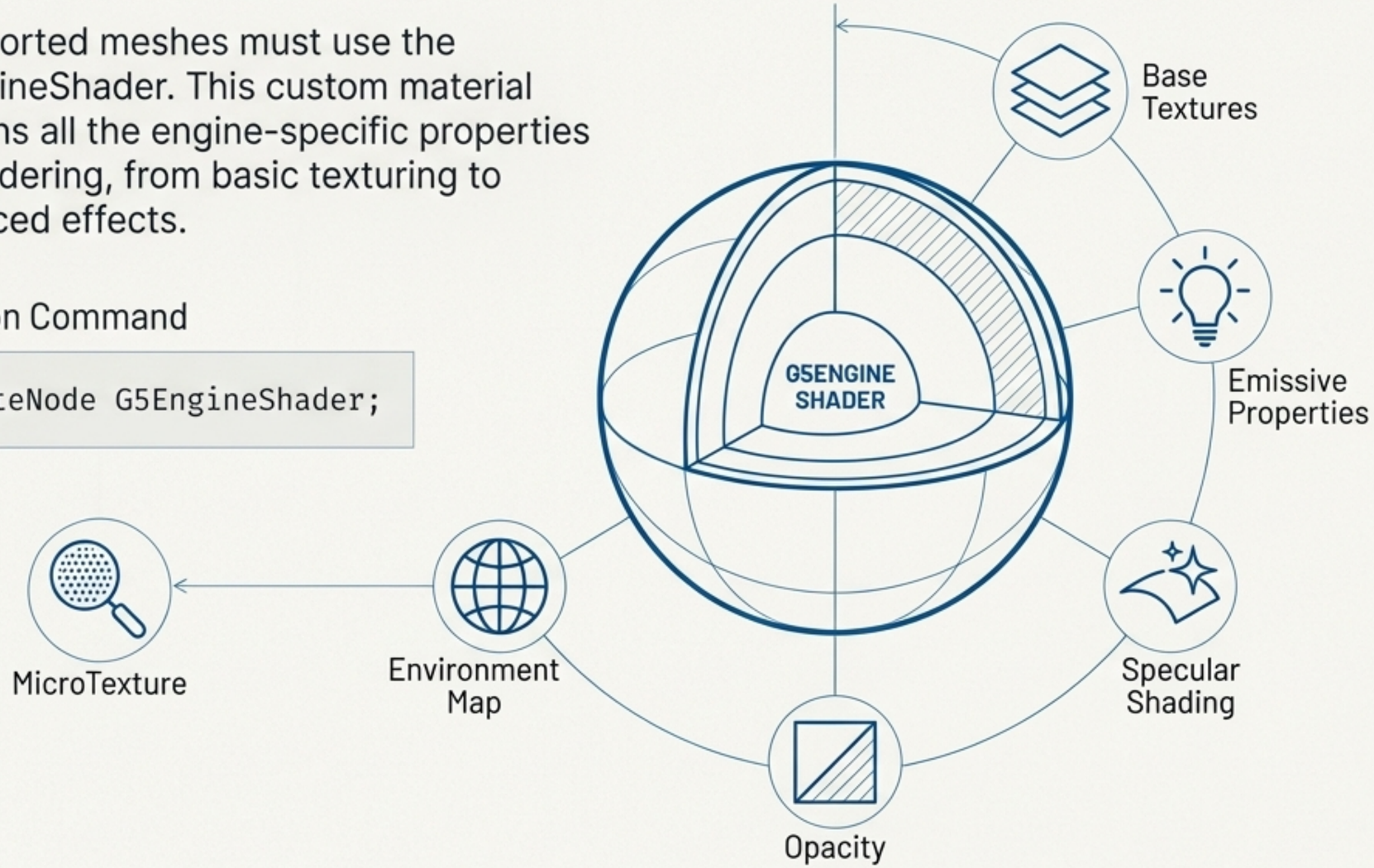
Best Practice Use a second, dedicated UV set for light maps. This allows you to keep your material UVs optimised for texture application while arranging light map UVs purely for packing efficiency and to meet the requirements above.

The G5EngineShader: An Asset's Digital Skin

All exported meshes must use the G5EngineShader. This custom material contains all the engine-specific properties for rendering, from basic texturing to advanced effects.

Creation Command

```
createNode G5EngineShader;
```



Key Attribute Groups

Core Maps
BaseTexture
normalCamera (Normal Map)
HeightMapTexture

Lighting Response
AmbientColor
DiffuseColor
UseSpecularShading
UseEmissiveEffect

Material Properties
IsDoubleSided
MixingMode (Transparency)
SurfaceType

Reflection
EnvironmentMapTexture

Detail
MicroTexture

G5EngineShader: Attribute Reference

Specular Shading

- **`UseSpecularShading`**: (Boolean) Toggles specular highlights on/off.
- **`SpecularPower`**: (Float) Controls highlight sharpness (high = glossy, low = matte).
- **`SpecularColor`**: (Color) Tints the highlight colour.
- **`SpecularTexture`**: (Texture) A map to control shininess across the surface.

Emissive Effect

- **`UseEmissiveEffect`**: (Boolean) Makes the material appear self-illuminated.
- **`EmissiveColor`**: (Color) Sets the colour of the emitted light.
- **`EmissiveTexture`**: (Texture) A mask for the emissive pattern.
- **`Intensity`**: (Float) The brightness of the emission.

Transparency

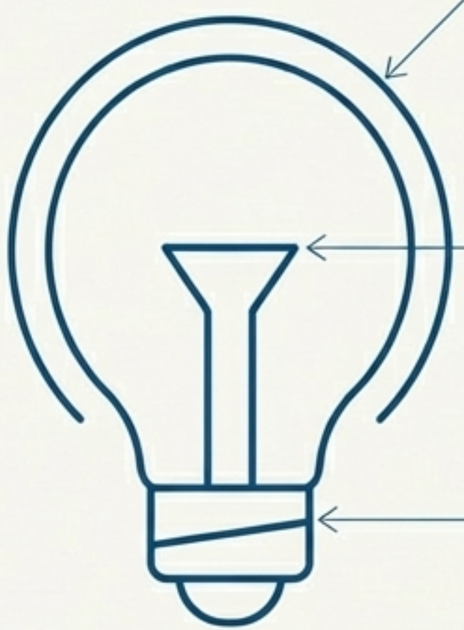
- **`MixingMode`**: (Integer) Controls the blending mode for transparency effects.
- **`Opacity`**: (Float) The overall transparency of the material (1.0 = opaque).
- **`OpacityMap`**: (Texture) A map controlling opacity across the surface.

Special Properties

- **`SurfaceType`**: (Integer) Classifies the material for specific game interactions (e.g., metal, glass).
- **`LightingModel`**: (Integer) Selects the lighting calculation model.
- **`SubstanceType`**: (Integer) May affect physics interactions.

Section 6.0: Configuring Lighting Assets.

Standard Maya lights require additional G5-specific attributes to function correctly in-game. These control performance, baked lighting, and falloff behaviour.



`DecayValue` : (Float) Controls how quickly light intensity diminishes over distance. Higher values mean a faster falloff.

`StaticFactor`: (Float) A multiplier for this light's contribution to pre-computed static lighting (light maps).

`GenerateShadows`: (Boolean) The master switch for enabling or disabling shadow casting for this light in-game. Disabling improves performance.

The Initialisation Command

Select a light and run the **`addLightProperties`** command.

```
addLightProperties;
```

Section 10.0: The Assembly Line: Executing the Export.

The Core Command `exportG5Resource`

This is the low-level plugin command that performs the export. It takes numerous parameters to control which data components are included.

The Recommended Interface `doExportScene`

For most uses, call the **`doExportScene`** MEL function. It is a comprehensive wrapper that handles scene loading, pathing, and calls **`exportG5Resource`** with the correct parameters. parameters.

The GUI

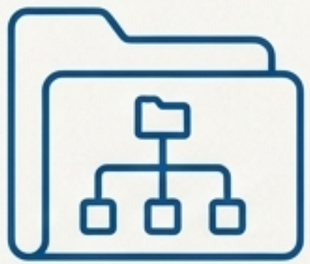
Export Dialogue

The 'G5Engine' shelf provides access to a graphical export dialogue (**`MS2ExportPlugin.mel`**). This is the most straightforward method for single-asset exports.

Key Options:

- ☑ The dialogue provides checkboxes for **`ExportModel`**, **`ExportSkin`**, **`ExportAnimation`**, **`ExportLights`**, and the critical **`ExportShapes`** for physics.

A Professional's Checklist: Best Practices for Clean Exports.



☆ 1. Maintain Scene Organisation

Group geometry logically under parent transforms. Parent physics colliders correctly under their visual meshes. Use the `DoNotExport` attribute liberally on helper objects instead of hiding them.

☆ 2. Validate Before Every Export

Always run validation checks. The most common errors are forgetting to run `addMeshProperties` on a mesh or failing to assign a `G5EngineShader`. Use Maya's cleanup tools to fix geometry errors like non-manifold faces.

☆ 3. Test Immediately In-Engine

An export can complete without errors but still be semantically wrong. Immediately load the asset in-game to verify scale, collision, materials, and overall behaviour. Catching issues early is critical.

Troubleshooting Common Faults

Symptom	Solution
G5 properties are missing from the Attribute Editor after running a command.	Check the Script Editor for errors. Ensure the correct object type (e.g., a mesh or light) was selected before execution.
The object appears in-game but has no behaviour and is static.	The `ClassName` attribute on the `G5Entity` is incorrect or misspelled. Verify it exactly matches a class name in the project's `.txt` script files.
Lighting on a model appears inside-out or is black.	Normals are inverted. Select the mesh and use Maya's tools to reverse or conform its normals so they point outwards.
The asset fails to export, citing a specific object.	The most likely cause is that `addMeshProperties` was not run on that object, or it has no `G5EngineShader` assigned.

Technical Reference: Command & Attribute Quick View

Core Commands

Command	Purpose
`addMeshProperties`	Registers all engine attributes on a mesh.
`addLightProperties`	Registers all engine attributes on a light.
`createPhysicsShape(int)`	Generates collision geometry (1=Box, 2=Capsule, 3=Cylinder).
`doExportScene(...)`	The recommended batch export wrapper function.

Essential Attributes

Attribute	Object Type	Description
`ClassName`	G5Entity	Links the asset to a runtime behaviour script.
`DoNotExport`	Mesh	Excludes the object from the final export.
`IsWalkMesh`	Mesh	Marks a surface as a walkable navigation area.
`IsPhysicsShape`	Physics Shape	Identifies the object as a physics-only collider.
`GenerateShadows`	Light	Master toggle for a light's ability to cast shadows.
`UseEmissiveEffect`	Shader	Enables self-illumination for a material.

The G5 Pipeline at a Glance: From Setup to Engine.