krma documentation

Part I: Getting krma installed

If you are building krma on it's own, first start by cloning the repository:
git clone git@github.com:awolven/krma.git

cd krma/
git submodule update --init --recursive

This will install cl-vulkan and the font engine.

Next, download the Khronos VulkanSDK for your platform.

If on macOS, once the installer has run, cd to ~/VulkanSDK/1.2.xxx.y/
Replace 1.2.xxx.y with the release version of the VulkanSDK you downloaded.
chmod + x install_vulkan.py
./install_vulkan.py

Next open krma/compile.sh in an editor. Change the path to GLSL_COMPILER to be the path to the bin directory in your VulkanSDK (i.e. edit the version number if necessary and uncomment only the path for your platform).

cd krma/
./compile.sh

This will compile the shaders.

Next open krma/submodules/cl-vulkan/ifc/load-libraries.lisp in an editor. There too edit the version number to point to the SDK for your platform.

If you cloned krma standalone, you need to put this line in your .sbclrc:
(pushnew "path/to/krma/" asdf:*central-registry* :test #'equalp)

This step assumes you have set up sbcl and quicklisp already.

Start lisp and evaluate:
(ql:quickload :krma)

Or if krma is a submodule of kons-9:
(ql:quickload :kons-9)

If you're running kons-9 you can then evaluate the expression:
(kons-9:run)

If you're running krma standalone, then you should evaluate the expression:
(krma::run)

To interact with Lisp while kons-9/krma is running you will need to have slime or sly installed.
From a REPL in slime/sly, try evaluating:
(krma:add-2d-line 0 0 100 100 :color #x0000ffff)

This should draw a diagonal blue line in the upper left hand corner of your window.
Try:
(krma:add-text "The quick brown fox jumped over the lazy dog." 100 200)

This is a basic demonstration of krma's retained-mode API.

Part II: krma basics

KRMA functionality is divided into two categories:

- The immediate mode facilities
- The retained mode facilities

The immediate mode facilities are intended to be used for the most dynamic and/or interactive parts of your application, consider:

1. Dragging or shaping geometry with the mouse.
2. Rapidly updating the mesh of an object, such as a swimming fish.

The retained mode facilities are designed for providing efficient rendering to objects that don't change their mesh much, but could be translating/scaling/rotating, changing colors, or responding to changes in lighting, such as a soccer ball.
The APIs:
Immediate mode drawing functions start with "draw-" or "scene-draw-". They must be executed at the appropriate location in the render thread (i.e. before being bufferized and before the vulkan or metal draw indexed command. There are functions that can be bound to the application object that will get executed at this appropriate location/time.

Retained mode drawing functions start with "add-" or "scene-add-". They are designed to be executed in other threads besides the render thread, and the work of the function is submitted to a queue in the render thread, so as not to modify draw lists at an inappropriate time, such as when the compactor thread is compacting draw-lists or when the render thread is bufferizing the CPU memory of the draw-list.

Goups:

In OpenGL, to draw geometry in a new location, you would set the matrix mode to modelview push the current modelview matrix, load an identity matrix and then call translate, scale or rotate, finally drawing the geometry in the new position and then pop the matrix to make the previous matrix current again.

In krma, it is a bit different. In immediate mode, you call the drawing commands with a "group" argument, which is a lisp atom. Then you alter the "group" in commands by calling krma:group-set-model-matrix-1, krma:group-apply-model-matrix-1, or krma:group-set-light-position-1. The functions ending in "-1" or starting with "draw-" or "scene-draw-" are intended to be called in the render thread. They do not submit work to a queue, the work is done immediately. The argument for "group" is a lisp atom, such as a keyword symbol. All the geometry in the group will be drawn with the same model matrix. Draw commands with different groups can be interleaved if necessary.

Currently in krma, the lighting settings also apply to groups, or if not set, will default to the scene lighting settings. Color, texture, and normals are properties of the geometry in both immediate mode or retained mode, but color from the geometry can be overridden by the group or the primitive.

Groups with the same name are not shared between immediate mode and retained mode. The usefulness of having the immediate mode and retained mode geometry share a group is arguable, and would only result in hash table locking the render thread, where we want to avoid locking.

In retained mode, you are issuing drawing commands from another thread besides the render thread. Retained mode was not a feature of OpenGL 1.x, so if you have only used OpenGL 1.1 you may not be familiar with the idea. It's called retained mode because the geometry persists in the draw lists for multiple iterations of the render thread. The architecture of this library submits work to the render thread in a manner that will not crash the render thread. This is because one modifies lisp programs while they are running. In the C++ development paradigm this is not important. In immediate mode, the draw lists are cleared at the start of the iteration. Immediate mode functions run in the render thread and can crash the render thread, but the render thread has been given an "ignore" restart, so while the render thread is halted in the debugger, the programmer can repair the broken immediate mode functions, and then select the "ignore" restart where the render thread will continue where it left off, hopefully not crashing again on the next frame. Immediate mode must redraw the geometry every frame, which is good for dynamic/interactive functionality, but puts extra load on the CPU.

Primitives:

Under the hood, there are two ways krma renders geometry. The ordinary use is for the draw call to the GPU to render the whole draw-list in one call. Another mode of drawing is for krma to render a draw-list in a succession of GPU draw calls. For example, line_strip and triangle_strip topologies require multiple draw calls, because if krma rendered the whole draw-list in one command, the strips would be connected to one another, and it is a poor use of memory to have a separate draw-lists for each triangle strip or line strip. Another reason to render draw-lists in a succession of GPU calls is so each "primitive" can potentially have it's own model matrix, color override, point size, line thickness (PC only) and lighting conditions. You also may want to be able to delete a primitive without deleting a whole group, or set the color of a primitive without setting the color of the whole group. The API functions which deal with primitives are of this pattern: "krma:add- ... -primitive", "krma:scene-add- ... -primitive" and "krma:primitive- ...". "Add" calls to the API, return a non-negative integer, which is called a "handle". The programmer can hold on to this handle and use it to delete or modify a primitive. Group commands such as "krma:delete-group" will delete the whole group, invalidating handles of any primitive in that group. The delete-group[s] function is for use with retained mode, because in immediate mode, if you want to delete a group, you simply don't draw it in the next frame. So primitives are generally used for retained mode, but in the case of line strips and triangle strips, they are also used for immediate mode.

Classes:

Krma provides a superclass for your application, which contains the driver objects necessary to make gpu calls. You can subclass this mixin and store any global application information there.

Krma also provides a superclass for your scene object, which contains the "draw-data" (i.e. packs of draw-lists work queues and hash tables) which enables krma to run. Subclass the krma-essential-scene-mixin to create your own scene classes. Drawing is always local to a scene.

Camera:

Krma provides accessor methods which can access the matrices in the camera. There are four matrices in the camera, the 2d projection matrix, the 2d view matrix, the 3d projection matrix and the 3d view matrix.   A scene should always have a default camera.

The API:

Draw & Add calls come in two varieties, one call is a shorthand call with keyword arguments which have defaults, which calls the second variety of calls, the "scene-draw-..." and "scene-add-..." which as their first argument take the scene object. "draw-..." and "add-..." calls pass in (application-scene *app*), the default scene, if not provided, as their first argument to the "scene-add-..." and "scene-draw-..." functions. The scene-add- and scene-draw- functions take positional arguments rather than keyword arguments for better performance.

Many drawing functions have an argument for "vertices", generate vertices lists from your application's objects.

This is a list of calls implemented so far:

Function: **add-2d-point-primitive** *[krma] x y &key color point-size matrix group scene*

> Retained-mode function, creates a primitive, returns a handle. Calls scene-add-2d-point-primitive with color defaulting to *default-color*, point-size defaulting to *default-point-size*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x and y must be real numbers.

Function: **add-2d-point** *[krma] x y &key color point-size group scene*

> Retained-mode function, returns no values. Calls scene-add-2d-point with color defaulting to *default-color*, point-size defaulting to *default-point-size*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments x and y must be real numbers.

Function: **draw-2d-point** *[krma] x y &key color point-size group scene*

> Immediate-mode function, returns no values. Calls scene-draw-2d-point with color defaulting to *default-color*, point-size defaulting to *default-point-size*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments x and y must be real numbers.

Function: **add-3d-point-primitive** *[krma] x y z &key color point-size matrix group scene*

> Retained-mode function, creates a primitive, returns a handle. Calls scene-add-3d-point-primitive with color defaulting to *default-color*, point-size defaulting to *default-point-size*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x, y and z must be real numbers.

Function: **add-3d-point** *[krma] x y z &key color point-size group scene*

> Retained-mode function, returns no values. Calls scene-add-3d-point with color defaulting to *default-color*, point-size defaulting to *default-point-size*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments x, y and z must be real numbers.

Function: **draw-3d-point** *[krma] x y z &key color point-size group scene*

Immediate-mode function, returns no values. Calls scene-draw-3d-point with color defaulting to *default-color*, point-size defaulting to *default-point-size*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments x, y and z must be real numbers.

Function: **add-2d-line-primitive** *[krma] x0 y0 x1 y1 &key color line-thickness matrix group scene*

Retained-mode function, creates a primitive, returns a handle. Calls scene-add-2d-line-primitive with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x0, y0, x1, and y1 are the endpoints of the line and must be real numbers.

Function: **add-2d-line** *[krma] x0 y0 x1 y1 &key color line-thickness group scene*

Retained-mode function, returns no values. Calls scene-add-2d-line with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x0, y0, x1 and y1 are the endpoints of the line and must be real numbers.

Function: **draw-2d-line** *[krma] x0 y0 x1 y1 &key color line-thickness group scene*

Immediate-mode function, returns no values. Calls scene-draw-2d-line with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x0, y0, x1 and y1 are the endpoints of the line and must be real numbers.

Function: **add-3d-line-primitive** *[krma] x0 y0 z0 x1 y1 z1 &key color line-thickness matrix group scene*

Retained-mode function, creates a primitive, returns a handle. Calls scene-add-3d-line-primitive with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x0, y0, z0, x1, y1 and z1 are the endpoints of the line and must be real numbers.

Function: **add-3d-line** *[krma] x0 y0 z0 x1 y1 z1 &key color line-thickness group scene*

Retained-mode function, returns no values. Calls scene-add-3d-line with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x0, y0, z0, x1, y1 and z1 are the endpoints of the line and must be real numbers.

Function: **draw-3d-line** *[krma] x0 y0 z0 x1 y1 z1 &key color line-thickness group scene*

Immediate-mode function, returns no values. Calls scene-draw-3d-line-primitive with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments x0, y0, z0, x1, y1 and z1 are the endpoints of the line and must be real numbers.

Function: **add-multicolor-2d-polyline-primitive** *[krma] vertices &key closed? line-thickness matrix group scene*

Retained-mode function, creates a primitive, returns a handle. Calls scene-add-multicolor-2d-polyline-primitive with closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required argument vertices should be of the form (list x0 y0 color0 x1 y1 color1 ... xn yn colorn) where the x and y values must be real numbers and the color value must be a color.

Function: **add-multicolor-2d-polyline** *[krma] vertices &key closed? line-thickness group scene*

Retained-mode function, returns no values. Calls scene-add-multicolor-2d-polyline with closed? defaulting to nil. line-thickness defaulting to *default-line-thickness*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required argument vertices should be of the form (list x0 y0 color0 x1 y1 color1 ... xn yn colorn) where the x and y values must be real numbers and the color value must be a color.

Function: **draw-multicolor-2d-polyline** *[krma] vertices &key closed? line-thickness group scene*

Immediate-mode function, returns no values. Calls scene-draw-multicolor-2d-polyline with closed? defaulting to nil. line-thickness defaulting to *default-line-thickness*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required argument vertices should be of the form (list x0 y0 color0 x1 y1 color1 ... xn yn colorn) where the x and y values must be real numbers and the color value must be a color.

Function: **add-2d-polyline-primitive** *[krma] vertices &key closed? color line-thickness matrix group scene*

Retained-mode function, creates a primitive, returns a handle. Calls scene-add-2d-polyline-primitive with closed? defaulting to nil, color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 x1 y1 ... xn yn) where the x and y values must be real numbers.

Function: **add-2d-polyline** *[krma] vertices &key closed? color line-thickness group scene*

Retained-mode function, returns no values. Calls scene-add-2d-polyline with closed? defaulting to nil, color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 x1 y1 ... xn yn) where the x and y values must be real numbers.

Function: **draw-2d-polyline** *[krma] vertices &key closed? color line-thickness group scene*

Immediate-mode function, returns no values. Calls scene-draw-2d-polyline with closed? defaulting to nil, color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 x1 y1 ... xn yn) where the x and y values must be real numbers.

Function: **add-2d-triangle-primitive** *[krma] x0 y0 x1 y1 x2 y2 &keycolor line-thickness matrix group scene*

Retained-mode function, creates a primitive outline of a triangle, returns a handle. Calls scene-add-2d-triangle-primitive with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required arguments represent the vertices of the triangle and must be real numbers.

Function: **add-2d-triangle** *[krma] x0 y0 x1 y1 x2 y2 &keycolor line-thickness group scene*

Retained-mode function, creates an outline of a triangle, returns no values. Calls scene-add-2d-triangle with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments represent the vertices of the triangle and must be real numbers.

Function: **draw-2d-triangle** *[krma] x0 y0 x1 y1 x2 y2 &keycolor line-thickness group scene*

Immediate-mode function, creates an outline of a triangle, returns no values. Calls scene-draw-2d-triangle with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments represent the vertices of the triangle and must be real numbers.

Function: **add-2d-rectangle-primitive** *[krma] x0 y0 x1 y1 &key color line-thickness matrix group scene*

Retained-mode function, creates a primitive outline of a rectangle, returns a handle. Calls scene-add-2d-rectangle-primitive with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required arguments represent the top-left and bottom-right corners of the rectangle, and must be real numbers.

Function: **add-2d-rectangle** *[krma] x0 y0 x1 y1 &key color line-thickness group scene*

Retained-mode function, creates an outline of a rectangle, returns no values. Calls scene-add-2d-rectangle with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*).

The required arguments represent the top-left and bottom-right corners of the rectangle, and must be real numbers.

Function: **draw-2d-rectangle** *[krma] x0 y0 x1 y1 &key color line-thickness group scene*

Immediate-mode function, creates an outline of a rectangle, returns no values. Calls scene-draw-2d-rectangle with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments represent the top-left and bottom-right corners of the rectangle, and must be real numbers.

Function: **add-2d-circular-arc-primitive** *[krma] center-x center-y radius start-angle end-angle &key closed? color line-thickness number-of-segments matrix group scene*

Retained-mode function, creates a primitive, a polyline representing the arc of a circle, returns a handle. Calls scene-add-2d-circular-arc-primitive with closed? defaulting to nil, color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, number-of-segments defaulting to 64, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required arguments should be real numbers and start-angle and end-angle are in radians.

Function: **add-2d-circular-arc** *[krma] center-x center-y radius start-angle end-angle &key closed? color line-thickness number-of-segments group scene*

Retained-mode function, creates a polyline representing the arc of a circle, returns no values. Calls scene-add-2d-circular-arc with closed? defaulting to nil, color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, number-of-segments defaulting to 64, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments should be real numbers and start-angle and end-angle are in radians.

Function: **draw-2d-circular-arc** *[krma] center-x center-y radius start-angle end-angle &key closed? color line-thickness number-of-segments group scene*

Immediate-mode function, creates a polyline representing the arc of a circle, returns no values. Calls scene-add-2d-circular-arc with closed? defaulting to nil, color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, number-of-segments defaulting to 64, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments should be real numbers and start-angle and end-angle are in radians.

Function: **add-2d-circle-primitive** *[krma] center-x center-y radius &key color line-thickness number-of-segments matrix group scene*

Retained-mode function, creates a primitive, a polyline representing the outline of a circle, returns a handle. Calls scene-add-2d-circle-primitive with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, number-of-segments defaulting to 64, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required arguments should be real numbers.

Function: **add-2d-circl**e *[krma] center-x center-y radius &key color line-thickness number-of-segments group scene*

Retained-mode function, creates a polyline representing the outline of a circle, returns a no values. Calls scene-add-2d-circle with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, number-of-segments defaulting to 64, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments should be real numbers.

Function: **draw-2d-circle** *[krma] center-x center-y radius &key color line-thickness number-of-segments group scene*

Immediate-mode function, creates a polyline representing the outline of a circle, returns a no values. Calls scene-draw-2d-circle with color defaulting to *default-color*, line-thickness defaulting to *default-line-thickness*, number-of-segments defaulting to 64, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments should be real numbers.

Function: **add-filled-2d-circle-primitive** *[krma] center-x center-y radius &key color number-of-sectors matrix group scene*

Retained-mode function, creates a primitive, a filled 2d circle, returns a handle. Calls scene-add-filled-2d-circle-primitive with color defaulting to *default-color*, number-of-sectors defaulting to 64, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required arguments should be real numbers.

Function: **add-filled-2d-circle** *[krma] center-x center-y radius &key color number-of-sectors group scene*

Retained-mode function, creates a filled 2d circle, returns no values. Calls scene-add-filled-2d-circle with color defaulting to *default-color*, number-of-sectors defaulting to 64, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments should be real numbers.

Function: **draw-filled-2d-circle** *[krma] center-x center-y radius &key color number-of-sectors group scene*

Immediate-mode function, creates a filled 2d circle, returns no values. Calls scene-draw-filled-2d-circle with color defaulting to *default-color*, number-of-sectors defaulting to 64, group defaulting to :default and scene defaulting to (application-scene *app*). The required arguments should be real numbers.

Function: **add-multicolor-3d-polyline-primitive** *[krma] vertices &key closed? line-thickness matrix group scene*

Retained-mode function, creates a primitive, a multicolored 3d polyline, returns a handle. Calls scene-add-multicolored-3d-polyline-primitive with closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x y and z values should be real numbers and the color values should represent a color.

Function: **add-multicolor-3d-polyline** *[krma] vertices &key closed? line-thickness group scene*

Retained-mode function, creates a multicolored 3d polyline, returns a no values. Calls scene-add-multicolored-3d-polyline with closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x y and z values should be real numbers and the color values should represent a color.

Function: **draw-multicolor-3d-polyline** *[krma] vertices &key closed? line-thickness group scene*

Immediate-mode function, creates a multicolored 3d polyline, returns a no values. Calls scene-draw-multicolored-3d-polyline with closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x y and z values should be real numbers and the color values should represent a color.

Function: **add-3d-polyline-primitive** *[krma] vertices &key color closed? line-thickness matrix group scene*

Retained-mode function, creates a primitive, a 3d polyline, returns a handle. Calls scene-add-3d-polyline-primitive with color defaulting to *default-color*, closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x y and z values should be real numbers.

Function: **add-3d-polyline** *[krma] vertices &key color closed? line-thickness group scene*

Retained-mode function, creates a 3d polyline, returns a no values. Calls scene-add-3d-polyline with color defaulting to *default-color*, closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x y and z values should be real numbers.

Function: **draw-3d-polyline** *[krma] vertices &key color closed? line-thickness group scene*

Immediate-mode function, creates a 3d polyline, returns a no values. Calls scene-draw-3d-polyline with color defaulting to *default-color*, closed? defaulting to nil, line-thickness defaulting to *default-line-thickness*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x y and z values should be real numbers.

Function: **add-filled-2d-triangle-list-primitive** *[krma] vertices &key color matrix group scene*

Retained-mode function, creates a primitive, a series of filled 2d triangles, returns a handle. Calls scene-add-filled-2d-triangle-list-primitive with color defaulting to *default-color*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 x10 y10 x20 y20 x01 y01 x11 y11 x21 y21 ... x0n y0n x1n y1n x2n y2n) where the x and y values represent vertices of a triangle in a series of triangles.

Function: **add-filled-2d-triangle-list** *[krma] vertices &key color group scene*

Retained-mode function, creates a series of filled 2d triangles, returns a no values. Calls scene-add-filled-2d-triangle-list with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 x10 y10 x20 y20 x01 y01 x11 y11 x21 y21 ... x0n y0n x1n y1n x2n y2n) where the x and y values represent vertices of a triangle in a series of triangles.

Function: **draw-filled-2d-triangle-list** *[krma] vertices &key color group scene*

Immediate-mode function, creates a series of filled 2d triangles, returns a no values. Calls scene-draw-filled-2d-triangle-list with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 x10 y10 x20 y20 x01 y01 x11 y11 x21 y21 ... x0n y0n x1n y1 x2n y2n) where the x and y values represent vertices of a triangle in a series of triangles.

Function: **add-filled-2d-rectangle-list-primitive** *[krma] vertices &key color matrix group scene*

Retained-mode function, creates a primitive, a series of filled 2d rectangles, returns a handle. Calls scene-add-filled-2d-rectangle-list-primitive with color defaulting to *default-color*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 x10 y10 x01 y01 x11 y11 ... x0n y0n x1n y1n) where the x0's, and y0's and the x1's and y1's represent the top-left and bottom-right of a series of rectangles.

Function: **add-filled-2d-rectangle-list** *[krma] vertices &key color group scene*

Retained-mode function a series of filled 2d rectangles, returns no values. Calls scene-add-filled-2d-rectangle-list with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 x10 y10 x01 y01 x11 y11 ... x0n y0n x1n y1n) where the x0's, and y0's and the x1's and y1's represent the top-left and bottom-right of a series of rectangles.

Function: **draw-filled-2d-rectangle-list** *[krma] vertices &key color group scene*

Immediate-mode function a series of filled 2d rectangles, returns no values. Calls scene-draw-filled-2d-rectangle-list with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 x10 y10 x01 y01 x11 y11 ... x0n y0n x1n y1n) where the x0's, and y0's and the x1's and y1's represent the top-left and bottom-right of a series of rectangles.

Function: **add-textured-2d-rectangle-list-primitive** *[krma] vertices &key color texture matrix group scene*

Retained-mode function, creates a primitive, a series of textured 2d rectangles, returns a handle. Calls scene-add-textured-2d-rectangle-list-primitive with color defaulting to *default-color*, texture defaulting to *white-texture*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 u00 v00 x10 y10 u10 v10 x01 y01 u01 v01 x11 y11 u11 v11 ... x0n y0n u0n v0n x1n y1n u1n v1n) where the x0's, and y0's and the x1's and y1's represent the top-left and bottom-right of a series of rectangles and the u0's and v0's are the normalized texture coordinates for the top-left corner and the u1's and v1's are the normalized texture coordinates for the bottom-right corner. of each rectangle.

Function: **add-textured-2d-rectangle-list** *[krma] vertices &key color texture group scene*

Retained-mode function, creates a series of textured 2d rectangles, returns no values. Calls scene-add-textured-2d-rectangle-list-primitive with color defaulting to *default-color*, texture defaulting to *white-texture*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 u00 v00 x10 y10 u10 v10 x01 y01 u01 v01 x11 y11 u11 v11 ... x0n y0n u0n v0n x1n y1n u1n v1n) where the x0's, and y0's and the x1's and y1's represent the top-left and bottom-right of a series of rectangles and the u0's and v0's are the normalized texture coordinates for the top-left corner and the u1's and v1's are the normalized texture coordinates for the bottom-right corner. of each rectangle.

Function: **draw-textured-2d-rectangle-list** *[krma] vertices &key color texture group scene*

Immediate-mode function, creates a series of textured 2d rectangles, returns no values. Calls scene-draw-textured-2d-rectangle-list with color defaulting to *default-color*, texture defaulting to *white-texture*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x00 y00 u00 v00 x10 y10 u10 v10 x01 y01 u01 v01 x11 y11 u11 v11 ... x0n y0n u0n v0n x1n y1n u1n v1n) where the x0's, and y0's and the x1's and y1's represent the top-left and bottom-right of a series of rectangles and the u0's and v0's are the normalized texture coordinates for the top-left corner and the u1's and v1's are the normalized texture coordinates for the bottom-right corner. of each rectangle.

Function: **add-filled-2d-convex-polygon-primitive** *[krma] vertices &keycolor matrix group scene*

Retained-mode function, creates a primitive, a filled 2d convex polygon, returns a handle. Calls scene-add-filled-2d-convex-polygon-primitive with color defaulting to *default-color*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 x1 y1 ... xn yn) where the x's, and y's represent a vertex of the polygon.

Function: **add-filled-2d-convex-polygon** *[krma] vertices &keycolor group scene*
Retained-mode function, creates a filled 2d convex polygon, returns a no values. Calls scene-add-filled-2d-convex-polygon with color defaulting to *default-color*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 x1 y1 ... xn yn) where the x's, and y's represent a vertex of the polygon.

Function: **draw-filled-2d-convex-polygon** *[krma] vertices &keycolor group scene*
Immediate-mode function, creates a filled 2d convex polygon, returns a no values. Calls scene-draw-filled-2d-convex-polygon with color defaulting to *default-color*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, should be of the form (list x0 y0 x1 y1 ... xn yn) where the x's, and y's represent a vertex of the polygon.

Function: **add-filled-3d-triangle-list-primitive** *[krma] vertices &key color shading-style light-position matrix group scene*
Retained-mode function, creates a primitive, a filled 3d triangle list, returns a handle. Calls scene-add-filled-3d-triangle-list-primitive-flat or scene-add-filled-3d-triangle-list-primitive-diffuse depending on whether shading-style is :diffuse or :flat, with color defaulting to *default-color*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y and z, in multiples of three (nine real numbers in each triangle sub-sequence). When shading-style is :diffuse, however, each vertex subsequence must be x, y, z, normal-x, normal-y, and normal-z, in multiples of three (eighteen real numbers in each triangle sub-sequence). Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **add-filled-3d-triangle-list** *[krma] vertices &key color shading-style group scene*
Retained-mode function, creates a filled 3d triangle list, returns a no-values. Calls scene-add-filled-3d-triangle-list-flat or scene-add-filled-3d-triangle-list-diffuse depending on whether shading-style is :diffuse or :flat, with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y and z, in multiples of three (nine real numbers in each triangle sub-sequence). When shading-

style is :diffuse, however, each vertex subsequence must be x, y, z, normal-x, normal-y, and normal-z, in multiples of three (eighteen real numbers in each triangle sub-sequence). Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **draw-filled-3d-triangle-list** *[krma] vertices &key color shading-style group scene*
Immediate-mode function, creates a filled 3d triangle list, returns a no-values. Calls scene-draw-filled-3d-triangle-list-flat or scene-draw-filled-3d-triangle-list-diffuse depending on whether shading-style is :diffuse or :flat, with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y and z, in multiples of three (nine real numbers in each triangle sub-sequence). When shading-style is :diffuse, however, each vertex subsequence must be x, y, z, normal-x, normal-y, and normal-z, in multiples of three (eighteen real numbers in each triangle sub-sequence). Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **add-filled-3d-triangle-strip-primitive** *[krma] vertices &keycolor shading-style light-position matrix group scene*
Retained-mode function, creates a primitive, a filled 3d triangle strip, returns a handle. Calls scene-add-filled-3d-triangle-strip-primitive-flat or scene-add-filled-3d-triangle-strip-primitive-diffuse depending on whether shading-style is :diffuse or :flat, with color defaulting to *default-color*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y and z. When shading-style is :diffuse, however, each vertex sub-sequence must be x, y, z, normal-x, normal-y, and normal-z.

Function: **draw-filled-3d-triangle-strip** *[krma] vertices &keycolor shading-style group scene*
Immediate-mode function, creates a filled 3d triangle strip, returns a no values. Calls scene-draw-filled-3d-triangle-strip-flat or scene-draw-filled-3d-triangle-strip-diffuse depending on whether shading-style is :diffuse or :flat, with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y and z. When shading-style is :diffuse, however, each vertex sub-sequence must be x, y, z, normal-x, normal-y, and normal-z.

Function: **add-textured-3d-triangle-list-primitive** *[krma] vertices &key color texture shading-style light-position matrix group scene*
Retained-mode function, creates a primitive, a textured 3d triangle list, returns a handle. Calls scene-add-textured-3d-triangle-list-primitive-flat when shading-style is :flat, currently errors

with shading-style :diffuse, with color defaulting to *default-color*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y, z, u, and v, where u and v are the normalized texture coordinates of the vertex. Vertices must be in multiples of three (fifteen real numbers in each triangle sub-sequence). When shading-style is :diffuse, however, each vertex subsequence must be x, y, z, normal-x, normal-y, normal-z, u and v in multiples of three (twenty-four real numbers in each triangle sub-sequence). Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **add-textured-3d-triangle-list** *[krma] vertices &key color texture shading-style group scene*

Retained-mode function, creates a textured 3d triangle list, returns no-values. Calls scene-add-textured-3d-triangle-list-flat when shading-style is :flat, currently errors with shading-style :diffuse, with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y, z, u, and v, where u and v are the normalized texture coordinates of the vertex. Vertices must be in multiples of three (fifteen real numbers in each triangle sub-sequence). When shading-style is :diffuse, however, each vertex subsequence must be x, y, z, normal-x, normal-y, normal-z, u and v in multiples of three (twenty-four real numbers in each triangle sub-sequence). Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **draw-textured-3d-triangle-list** *[krma] vertices &key color texture shading-style group scene*

Immediate-mode function, creates a textured 3d triangle list, returns no-values. Calls scene-draw-textured-3d-triangle-list-flat when shading-style is :flat, currently errors with shading-style :diffuse, with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y, z, u, and v, where u and v are the normalized texture coordinates of the vertex. Vertices must be in multiples of three (fifteen real numbers in each triangle sub-sequence). When shading-style is :diffuse, however, each vertex subsequence must be x, y, z, normal-x, normal-y, normal-z, u and v in multiples of three (twenty-four real numbers in each triangle sub-sequence). Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **add-textured-3d-triangle-strip-primitive** *[krma] vertices &key color texture shading-style light-position matrix group scene*

Retained-mode function, creates a primitive, a textured 3d triangle strip, returns a handle. Calls scene-add-textured-3d-triangle-strip-primitive-flat when shading-style is :flat, currently errors with shading-style :diffuse, with color defaulting to *default-color*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y, z, u and v, where u and v are the normalized texture coordinates of that vertex.

Function: **draw-textured-3d-triangle-strip** *[krma] vertices &key color texture shading-style group scene*

Immediate-mode function, creates a textured 3d triangle strip, returns a no values. Calls scene-draw-textured-3d-triangle-strip-primitive-flat when shading-style is :flat, currently errors with shading-style :diffuse, with color defaulting to *default-color*, group defaulting to :default and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat must be composed of at least three vertex sub-sequences of x, y, z, u and v, where u and v are the normalized texture coordinates of that vertex.

Function: **add-filled-3d-convex-polygon***[krma] vertices &key color shading-style light-position matrix group scene*

Retained-mode function, creates a primitive, a filled 3d convex polygon, returns a handle. Calls scene-add-filled-3d-convex-polygon-primitive-diffuse or scene-draw-filled-3d-convex-polygon-flat, depending on whether shading style is :diffuse or :flat, light-position defaults to nil, color defaulting to *default-color*, matrix defaulting to nil (identity), group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's represent a vertex of the polygon. When shading-style is :diffuse, vertices should be of the form (list x0 y0 z0 nx0 ny0 nz0 x1 y1 z1 nx1 ny1 nz1 ... xn yn zn nxn nyn nzn) where the x, y and z's represent a coordinates of a vertex and nx, ny and nz represent the normal at that vertex. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **add-filled-3d-convex-polygon***[krma] vertices &key color shading-style group scene*

Retained-mode function, creates a filled 2d convex polygon, returns no values. Calls scene-add-filled-3d-convex-polygon-diffuse or scene-add-filled-3d-convex-polygon-flat, depending on whether shading-style is :diffuse or :flat with color defaulting to *default-color*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's represent a vertex of the polygon. When shading-style is :diffuse, vertices should be of the form (list x0 y0 z0 nx0 ny0 nz0 x1 y1 z1 nx1 ny1 nz1 ... xn yn zn nxn nyn nzn) where the x, y and z's represent a coordinates of a vertex and nx, ny and nz represent the normal at that vertex. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **draw-filled-3d-convex-polygon** *[krma] vertices &key color shading-style group scene*

Immediate-mode function, creates a filled 3d convex polygon, returns no values. Calls scene-draw-filled-3d-convex-polygon-diffuse or scene-draw-filled-3d-convex-polygon-flat, depending on whether shading-style is :diffuse or :flat, with color defaulting to *default-color*, group defaulting to nil (no group) and scene defaulting to (application-scene *app*). The required argument, vertices, when shading-style is :flat should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's represent a vertex of the polygon. When shading-style is :diffuse, vertices should be of the form (list x0 y0 z0 nx0 ny0 nz0 x1 y1 z1 nx1 ny1 nz1 ... xn yn zn nxn nyn nzn) where the x, y and z's represent a coordinates of a vertex and nx, ny and nz represent the normal at that vertex. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is up.

Function: **add-filled-sphere-primitive** *[krma] origin-x origin-y origin-z radius &key color resolution shading-style light-position matrix group scene*

Retained-mode function, creates a primitive of a filled sphere, returns a handle. Calls scene-add-filled-sphere-primitive-diffuse when shading style is :diffuse, currently errors with any other shading style, with color defaulting to *default-color*, resolution defaulting to 64, light-position defaulting to nil, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments should be real numbers. radius should be positive.

Function: **add-filled-sphere** *[krma] origin-x origin-y origin-z radius &key color resolution shading-style group scene*

Retained-mode function, creates a filled sphere, returns a no values. Calls scene-add-filled-sphere-diffuse when shading style is :diffuse, currently errors with any other shading style, with color defaulting to *default-color*, resolution defaulting to 64, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments should be real numbers.   radius should be positive.

Function: **draw-filled-sphere** *[krma] origin-x origin-y origin-z radius &key color resolution shading-style group scene*

Immediate-mode function, creates a filled sphere, returns a no values. Calls scene-draw-filled-sphere-diffuse when shading style is :diffuse, currently errors with any other shading style, with color defaulting to *default-color*, resolution defaulting to 64, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments should be real numbers.   radius should be positive.

Function: **add-text-primitive** *[krma] string pos-x pos-y &key color font matrix group scene*

Retained-mode function, creates a primitive of a text string, returns a handle. Calls scene-add-text-primitive with color defaulting to *default-color*, font defaulting to *font*, matrix defaulting to nil (identity), group defaulting to nil (no group), and scene defaulting to (application-scene *app*). The required arguments should be real numbers. pos-x and pos-y represent the upper left corner of the text.

Function: **add-text** *[krma] string pos-x pos-y &key color font group scene*
Retained-mode function, creates text, returns a no values. Calls scene-add-text with color defaulting to *default-color*, font defaulting to *font*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments should be real numbers. pos-x and pos-y represent the upper left corner of the text.

Function: **draw-text** *[krma] string pos-x pos-y &key color font group scene*
Immediate-mode function, creates text, returns a no values. Calls scene-draw-text with color defaulting to *default-color*, font defaulting to *font*, group defaulting to :default, and scene defaulting to (application-scene *app*). The required arguments should be real numbers. pos-x and pos-y represent the upper left corner of the text.

Function: **scene-add-2d-point-primitive** *[krma] scene group model-matrix point-size color x y*
Retained-mode function, returns a handle for a 2d point primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), point-size should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. and x and y must be real numbers. Dispatches actual work to render thread. To delete the point, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-2d-point** *[krma] scene group point-size color x y*
Retained-mode function, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, point-size should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. and x and y must be real numbers. Dispatches actual work to render thread. To delete the point, you must delete the entire group.

Function: **scene-draw-2d-point***[krma] scene group point-size color x y*
Immediate-mode function, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, point-size should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. and x and y must be real numbers. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-3d-point-primitive** *[krma] scene group model-matrix point-size color x y z*

Retained-mode function, returns a handle for a 3d point primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom,   possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), point-size should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, and x, y and z must be real numbers. Dispatches actual work to render thread. To delete the point, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-3d-point** *[krma] scene group point-size color x y z*

Retained-mode function, adds a point to retained-mode draw-lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, point-size should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, and x and y must be real numbers. Dispatches actual work to render thread. To delete the point, you must delete the entire group.

Function: **scene-draw-3d-point** *[krma] scene group point-size color x y z*

Immediate-mode function, draws a 3d point, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, point-size should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, and x, y and z must be real numbers. Performs work in current thread, which should be the render thread.

Function: **scene-add-2d-line-primitive** *[krma] scene group model-matrix line-thickness color x0 y0 x1 y1*

Retained-mode function, returns a handle for a 2d line primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0 and x1, y1 must be real numbers which represent the endpoints of the line. Dispatches actual work to render thread. To delete the line segment, you must delete the primitive using the handle.

Function: **scene-add-2d-line** *[krma] scene group line-thickness color x0 y0 x1 y1*

Retained-mode function, adds a 2d line segment to draw lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0 and x1, y1 must be real numbers which represent the endpoints of the line. Dispatches actual work to render thread. To delete the line segment, you must delete the entire group.

Function: **scene-draw-2d-line** *[krma] scene group line-thickness color x0 y0 x1 y1*

Immediate-mode function, draws a 2d line segment. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0 and x1, y1 must be real numbers which represent the endpoints of the line. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-3d-line-primitive** *[krma] scene group model-matrix line-thickness color x0 y0 z0 x1 y1 z1*

Retained-mode function, returns a handle for a 3d line primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, z0 and x1, y1, z1 must be real numbers which represent the endpoints of the line. Dispatches actual work to render thread. To delete the line segment, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-3d-line** *[krma] scene group line-thickness color x0 y0 z0 x1 y1 z1*

Retained-mode function, adds a 3d line segment to the draw lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, z0 and x1, y1, z1 must be real numbers which represent the endpoints of the line. Dispatches actual work to render thread. To delete the line you must delete the entire group.

Function: **scene-draw-3d-line** *[krma] scene group line-thickness color x0 y0 z0 x1 y1 z1*

Immediate-mode function, draws a 3d line segment. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, z0 and x1, y1, z1 must be real numbers which represent the endpoints of the line. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-2d-polyline-primitive** *[krma] scene group model-matrix closed? line-thickness color vertices*

Retained-mode function, returns a handle for a 2d polyline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. vertices should be of the form (list x0 y0 x1 y1 ... xn yn) where the x's and the y's are vertex points of the polyline and must be real numbers. Dispatches actual work to render thread. To delete the polyline, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-2d-polyline** *[krma] scene group closed? line-thickness color vertices*

Retained-mode function, adds a 2d polyline to the draw lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. vertices should be of the form (list x0 y0 x1 y1 ... xn yn) where the x's and the y's are vertex points of the polyline and must be real numbers. Dispatches actual work to render thread. To delete the polyline, you must delete the entire group.

Function: **scene-draw-2d-polyline** *[krma] scene group closed? line-thickness color vertices*

Immediate-mode function, draws a 2d polyline. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. vertices should be of the form (list x0 y0 x1 y1 ... xn yn) where the x's and the y's are vertex points of the polyline and must be real numbers. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-2d-triangle-primitive** *[krma] scene group model-matrix line-thickness color x0 y0 x1 y1 x2 y2*

Retained-mode function, returns a handle for a 2d triangle outline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, x1, y1, x2 and y2 are the three vertex coordinates of the triangle and must be real numbers. Dispatches actual work to render thread. To delete the triangle, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-2d-triangle** *[krma] scene group line-thickness color x0 y0 x1 y1 x2 y2*

Retained-mode function, adds a 2d triangle outline to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, x1, y1, x2 and y2 are the three vertex coordinates of the triangle and must be real numbers.   Dispatches actual work to render thread. To delete the triangle, you must delete the entire group.

Function: **scene-draw-2d-triangle** *[krma] scene group line-thickness color x0 y0 x1 y1 x2 y2*

Immediate-mode function, draws a 2d triangle outline, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, x1, y1, x2 and y2 are the three vertex coordinates of the triangle and must be real numbers. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-2d-rectangle-primitive** *[krma] scene group model-matrix line-thickness color x0 y0 x1 y1*

Retained-mode function, returns a handle for a 2d rectangle outline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, x1, and y1 are the top-left and bottom-right corners of the rectangle and must be real numbers.   Dispatches actual work to render thread. To delete the rectangle, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-2d-rectangle** *[krma] scene group line-thickness color x0 y0 x1 y1*

Retained-mode function, adds a 2d rectangle outline to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, x1, and y1 are the top-left and bottom-right corners of the rectangle and must be real numbers.   Dispatches actual work to render thread. To delete the rectangle, you must delete the entire group.

Function: **scene-draw-2d-rectangle** *[krma] scene group line-thickness color x0 y0 x1 y1*

Immediate-mode function, draws a 2d rectangle outline, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an a non-null atom, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. x0, y0, x1, and y1 are the top-left and bottom-right corners of the rectangle and must be real numbers. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-multicolor-2d-polyline-primitive** *[krma] scene group model-matrix closed? line-thickness vertices*

Retained-mode function, returns a handle for a multicolored 2d polyline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. vertices should be of the form (list x0 y0 color0 x1 y1 color1 ... xn yn colorn) where the x's and the y's are vertex points of the polyline and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer.   Dispatches actual work to render thread. To delete the polyline, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-multicolor-2d-polyline** *[krma] scene group closed? line-thickness vertices*

Retained-mode function, adds a multicolored 2d polyline to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. vertices should be of the form (list x0 y0 color0 x1 y1 color1 ... xn yn colorn) where the x's and the y's are vertex points of the polyline and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer.   Dispatches actual work to render thread. To delete the polyline, you must delete the entire group.

Function: **scene-draw-multicolor-2d-polyline** *[krma] scene group closed? line-thickness vertices*

Immediate-mode function, draws a multicolored 2d polyline, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. vertices should be of the form (list x0 y0 color0 x1 y1 color1 ... xn yn colorn) where the x's and the y's are vertex points of the polyline and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-2d-circular-arc-primitive** *[krma] scene group model-matrix closed? line-thickness color*

center-x center-y radius start-angle end-angle &optional number-of-segments

Retained-mode function, returns a handle for a 2d circular arc outline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. center-x, and center-y must be real numbers, radius must be a positive real number, start-angle and end-angle are real numbers, measured in radians. number-of-segments must be a positive integer, and defaults to 64. Dispatches actual work to render thread. To delete the arc, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-2d-circular-arc** *[krma] scene group closed? line-thickness color center-x center-y radius start-angle end-angle &optional number-of-segments*

Retained-mode function, adds a 2d circular arc outline to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. center-x, and center-y must be real numbers, radius must be a positive real number, start-angle and end-angle are real numbers, measured in radians. number-of-segments must be a positive integer, and defaults to 64. Dispatches actual work to render thread. To delete the arc, you must delete the entire group.

Function: **scene-draw-2d-circular-arc** *[krma] scene group closed? line-thickness color center-x center-y radius start-angle end-angle &optional number-of-segments*

Immediate-mode function, draws 2d circular arc outline, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. center-x, and center-y must be real numbers, radius must be a positive real number, start-angle and end-angle are real numbers, measured in radians. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-2d-circle-primitive** *[krma] scene group model-matrix line-thickness color center-x center-y radius &optional number-of-segments*

Retained-mode function, returns a handle for a 2d circle outline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), line-thickness should be a positive real number. color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. center-x, and center-y must be real numbers, radius must be a positive real number. number-of-segments must be a positive integer, and defaults to 64. Dispatches actual work to render thread. To delete the arc, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-2d-circle** *[krma] scene group line-thickness color center-x center-y radius &optional number-of-segments*

Retained-mode function, adds a 2d circle outline to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, line-thickness should be a positive real number. color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. center-x, and center-y must be real numbers, radius must be a positive real number. number-of-segments must be a positive integer, and defaults to 64. Dispatches actual work to render thread. To delete the arc, you must delete the entire group.

Function: **scene-draw-2d-circle** *[krma] scene group line-thickness color center-x center-y radius &optional number-of-segments*

Immediate-mode function, draws a 2d circle outline, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, line-thickness should be a positive real number. color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. center-x, and center-y must be real numbers, radius must be a positive real number. number-of-segments must be a positive integer, and defaults to 64. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-3d-polyline-primitive** *[krma] scene group model-matrix closed? line-thickness color vertices*

Retained-mode function, returns a handle for a 3d polyline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. vertices should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's are the vertex points of the polyline and must be real numbers.   Dispatches actual work to render thread. To delete the polyline, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-3d-polyline** *[krma] scene group closed? line-thickness color vertices*

Retained-mode function, adds a 3d polyline to the draw lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. vertices should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's are the vertex points of the polyline and must be real numbers.   Dispatches actual work to render thread. To delete the polyline, you must delete the entire group.

Function: **scene-draw-3d-polyline** *[krma] scene group closed? line-thickness color vertices*

Immediate-mode function, draws a 3d polyline. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. vertices should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's are the vertex points of the polyline and must be real numbers.   Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-multicolor-3d-polyline-primitive** *[krma] scene group model-matrix closed? line-thickness vertices*

Retained-mode function, returns a handle for a multicolored 3d polyline primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. vertices should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x, y and z's are vertex points of the polyline and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer.   Dispatches actual work to render thread. To delete the polyline, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-multicolor-3d-polyline** *[krma] scene group closed? line-thickness vertices*

Retained-mode function, adds a multicolored 3d polyline to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. vertices should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x, y and z's are vertex points of the polyline and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer.   Dispatches actual work to render thread. To delete the polyline, you must delete the entire group.

Function: **scene-draw-multicolor-3d-polyline** *[krma] scene group closed? line-thickness vertices*

Immediate-mode function, draws a multicolored 3d polyline, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, closed? should be a boolean, which specifies whether to draw a segment between the last vertex and the first vertex, line-thickness should be a positive real number. vertices should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x, y and z's are vertex points of the polyline and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-2d-triangle-list-primitive** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 2d triangle list primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity),   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 x10 y10 x20 y20 x01 y01 x11 y11 x21 y21 ... x0n y0n x1n y1n x2n y2n) where the x and y values represent vertices of a triangle in a series of triangles and must be real numbers,   Dispatches actual work to render thread. To delete the triangle list, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-2d-triangle-list** *[krma] scene group color vertices*

Retained-mode function, adds a filled 2d triangle list to the draw-lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom,   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 x10 y10 x20 y20 x01 y01 x11 y11 x21 y21 ... x0n y0n x1n y1n x2n y2n) where the x and y values represent vertices of a triangle in a series of triangles and must be real numbers,   Dispatches actual work to render thread. To delete the triangle list, you must delete the entire group.

Function: **scene-draw-filled-2d-triangle-list** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 2d triangle list. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom,   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 x10 y10 x20 y20 x01 y01 x11 y11 x21 y21 ... x0n y0n x1n y1n x2n y2n) where the x and y values represent vertices of a triangle in a series of triangles and must be real numbers,   Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-2d-triangle-strip-primitive** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 2d triangle strip primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity),   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 x1 y1 ... xn yn) where the x and y values represent successive vertices of a triangle strip and must be real numbers,   Dispatches actual work to render thread. To delete the triangle strip, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-draw-filled-2d-triangle-strip** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 2d triangle strip. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom,   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 x1 y1 ... xn yn) where the x and y values represent successive vertices of a triangle strip and must be real numbers. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-2d-rectangle-list-primitive** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 2d rectangle list primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 x10 y10 x10 y10 x11 y11 ... x0n y0n x1n y1n) where each pair of successive x and y's represent the top-left corner followed by the bottom-right corner of each rectangle and must be real numbers,   Dispatches actual work to render thread. To delete the rectangle list, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-2d-rectangle-list** *[krma] scene group color vertices*

Retained-mode function, adds a filled 2d rectangle list to the draw-lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 x10 y10 x10 y10 x11 y11 ... x0n y0n x1n y1n) where each pair of successive x and y's represent the top-left corner followed by the bottom-right corner of each rectangle and must be real numbers. Dispatches actual work to render thread. To delete the rectangle list, you must delete the entire group.

Function: **scene-draw-filled-2d-rectangle-list** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 2d rectangle list. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 x10 y10 x10 y10 x11 y11 ... x0n y0n x1n y1n) where each pair of successive x and y's represent the top-left corner followed by the bottom-right corner of each rectangle and must be real numbers. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-textured-2d-rectangle-list-primitive** *[krma] scene group model-matrix texture color vertices*

Retained-mode function, returns a handle for a textured 2d rectangle list primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 u00 v00 x10 y10 u10 v10 x01 y01 u01 v01 x11 y11 u11 v11 ... x0n y0n u0n v0n x1n y1n u1n v1n) where each pair of successive x, y, u and v represent the top-left corner followed by the bottom-right corner of each rectangle with their normalized texture coordinates, and must be real numbers. There must be at least one pair of the sequence x, y, u, v. Dispatches actual work to render thread. To delete the rectangle list, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-textured-2d-rectangle-list** *[krma] scene group texture color vertices*

Retained-mode function, adds a textured 2d rectangle list to the draw-lists. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 u00 v00 x10 y10 u10 v10 x01 y01 u01 v01 x11 y11 u11 v11 ... x0n y0n u0n v0n x1n y1n u1n v1n) where each pair of successive x, y, u and v represent the top-left corner followed by the bottom-right corner of each rectangle with their normalized texture coordinates, and must be real numbers. There must be at least one pair of the sequence x, y, u, v. Dispatches actual work to render thread. To delete the rectangle list, you must delete the entire group.

Function: **scene-draw-textured-2d-rectangle-list** *[krma] scene group texture color vertices*

Immediate-mode function, draws a textured 2d rectangle list. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 u00 v00 x10 y10 u10 v10 x01 y01 u01 v01 x11 y11 u11 v11 ... x0n y0n u0n v0n x1n y1n u1n v1n) where each pair of successive x, y, u and v represent the top-left corner followed by the bottom-right corner of each rectangle with their normalized texture coordinates, and must be real numbers. There must be at least one pair of the sequence x, y, u, v. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-2d-convex-polygon-primitive** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 2d convex polygon primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 x1 y1 ... xn yn) where each successive x and y are the vertices of the polygon, and must be real numbers. There must be at least three x, y pairs in vertices. Dispatches actual work to render thread. To delete the polygon, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-2d-convex-polygon** *[krma] scene group color vertices*

Retained-mode function, adds a filled 2d convex polygon to the draw-lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 x1 y1 ... xn yn) where each successive x and y are the vertices of the polygon, and must be real numbers. There must be at least three x, y pairs in vertices. Dispatches actual work to render thread. To delete the polygon, you must delete the entire group.

Function: **scene-draw-filled-2d-convex-polygon** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 2d convex polygon, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 x1 y1 ... xn yn) where each successive x and y are the vertices of the polygon, and must be real numbers. There must be at least three x, y pairs in vertices. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-2d-circle-primitive** *[krma] scene group model-matrix color center-x center-y radius*

&optional number-of-sectors

Retained-mode function, returns a handle for a filled 2d circle primitive. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, center-x and center-y should be real numbers, radius should be a positive real number, number-of-sectors defaults to 64. Dispatches actual work to render thread. To delete the circle, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-2d-circle** *[krma] scene group color center-x center-y radius &optional number-of-sectors*

Retained-mode function, adds a filled 2d circle to the draw lists, returns no values. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, center-x and center-y should be real numbers, radius should be a positive real number, number-of-sectors defaults to 64. Dispatches actual work to render thread. To delete the circle, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-draw-filled-2d-circl**e *[krma] scene group color center-x center-y radius &optional number-of-sectors*

Immediate-mode function, draws a filled 2d circle. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, center-x and center-y should be real numbers, radius should be a positive real number, number-of-sectors defaults to 64. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-3d-triangle-list-primitive-flat** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 3d triangle list primitive. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 z00 x10 y10 z10 x20 y20 z20 x01 y01 z01 x11 y11 z11 x21 y21 z21... x0n y0n z0n x1n y1n z1n x2n y2n z2n) where the x, y and z values represent vertices of a triangle in a series of triangles and must be real numbers. There must be at least three sets of x, y and z, and additional vertices come as 3 sets each. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the triangle list, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-3d-triangle-list-flat** *[krma] scene group color vertices*

Retained-mode function, adds a filled 3d triangle list to the draw-lists, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 z00 x10 y10 z10 x20 y20 z20 x01 y01 z01 x11 y11 z11 x21 y21 z21... x0n y0n z0n x1n y1n z1n x2n y2n z2n) where the x, y and z values represent vertices of a triangle in a series of triangles and must be real numbers. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the triangle list, you must delete the entire group.

Function: **scene-draw-filled-3d-triangle-list-flat** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 2d triangle list, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 z00 x10 y10 z10 x20 y20 z20 x01 y01 z01 x11 y11 z11 x21 y21 z21... x0n y0n z0n x1n y1n z1n x2n y2n z2n) where the x, y and z values represent vertices of a triangle in a series of triangles and must be real numbers. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-3d-triangle-list-primitive-diffuse** *[krma] scene group model-matrix color vertices light-position*

Retained-mode function, returns a handle for a filled 3d triangle list primitive. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 z00 x10 y10 z10 x20 y20 z20 x01 y01 z01 x11 y11 z11 x21 y21 z21... x0n y0n z0n x1n y1n z1n x2n y2n z2n) where the x, y and z values represent vertices of a triangle in a series of triangles and must be real numbers. There must be at least three sets of x, y and z, and additional vertices come as 3 sets each. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the triangle list, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-3d-triangle-list-diffuse** *[krma] scene group color vertices*

Retained-mode function, adds a filled 3d triangle list to the draw-lists, returns no values. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 z00 x10 y10 z10 x20 y20 z20 x01 y01 z01 x11 y11 z11 x21 y21 z21... x0n y0n z0n x1n y1n z1n x2n y2n z2n) where the x, y and z values represent vertices of a triangle in a series of triangles and must be real numbers. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the triangle list, you must delete the entire group.

Function: **scene-draw-filled-3d-triangle-list-diffuse** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 2d triangle list, returns no values. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom,   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x00 y00 z00 x10 y10 z10 x20 y20 z20 x01 y01 z01 x11 y11 z11 x21 y21 z21... x0n y0n z0n x1n y1n z1n x2n y2n z2n) where the x, y and z values represent vertices of a triangle in a series of triangles and must be real numbers. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-3d-triangle-strip-primitive-flat** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 3d triangle strip primitive. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity),   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z values represent successive vertices of a triangle strip and must be real numbers. There must be at least three vertices. Dispatches actual work to render thread. To delete the triangle strip, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-draw-filled-3d-triangle-strip-flat** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 3d triangle strip, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z values represent successive vertices of a triangle strip and must be real numbers. There must be at least 3 vertices. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-3d-triangle-strip-primitive-diffuse** *[krma] scene group model-matrix color vertices light-position*

Retained-mode function, returns a handle for a filled 3d triangle strip primitive. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity),   color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z values represent successive vertices of a triangle strip and must be real numbers. There must be at least three vertices. light-position must either be a 3d-vectors:vec4 or null. Dispatches actual work to render thread. To delete the triangle strip, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-draw-filled-3d-triangle-strip-diffuse** *[krma] scene group color vertices*

Retained-mode function, returns a handle for a filled 3d triangle strip primitive. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z values represent successive vertices of a triangle strip and must be real numbers. There must be at least three vertices. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-3d-convex-polygon-primitive-diffuse** *[krma] scene group model-matrix color vertices light-position*

Retained-mode function, returns a handle for a filled 3d convex polygon primitive. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where each successive x, y and z are the vertices of the polygon, and must be real numbers. There must be at least three x, y, z triplets in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-3d-convex-polygon-diffuse** *[krma] scene group color vertices*

Retained-mode function, adds a filled 3d convex polygon to the draw-lists, returns no values. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where each successive x, y and z are the vertices of the polygon, and must be real numbers. There must be at least three x, y, z triplets in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the entire group.

Function: **scene-draw-filled-3d-convex-polygon-diffuse** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 3d convex polygon, returns no values. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices must be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where each successive x, y and z are the vertices of the polygon, and must be real numbers. There must be at least three x, y, z triplets in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-3d-convex-polygon-primitive-flat** *[krma] scene group model-matrix color vertices*

Retained-mode function, returns a handle for a filled 3d convex polygon primitive. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's represent a vertex of the polygon and must be real numbers. There must be at least three x, y, z triplets in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-3d-convex-polygon-flat** *[krma] scene group color vertices*

Retained-mode function, adds a filled 3d convex polygon to the draw-lists, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's represent a vertex of the polygon and must be real numbers. There must be at least three x, y, z triplets in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the entire group.

Function: **scene-draw-filled-3d-convex-polygon-flat** *[krma] scene group color vertices*

Immediate-mode function, draws a filled 3d convex polygon, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices should be of the form (list x0 y0 z0 x1 y1 z1 ... xn yn zn) where the x, y and z's represent a vertex of the polygon and must be real numbers. There must be at least three x, y, z triplets in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-multicolor-3d-convex-polygon-primitive-diffuse** *[krma] scene group model-matrix vertices light-position*

Retained-mode function, returns a handle for a multicolored 3d convex polygon primitive. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), vertices must be of the form (list x0 y0 z0 nx0 ny0 nz0 color0 x1 y1 z1 nx1 ny1 nz1 color1 ... xn yn zn nxn nyn nzn colorn) where each successive x, y z, nx, ny, nz and color are the vertices of the polygon and the normal at that vertex, and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. There must be at least three x, y, z, nx, ny, nz, color seven-tuples in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-multicolor-3d-convex-polygon-diffuse** *[krma] scene group vertices*

Retained-mode function, adds a multicolored 3d convex polygon to the draw lists, returns no values. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, vertices must be of the form (list x0 y0 z0 nx0 ny0 nz0 color0 x1 y1 z1 nx1 ny1 nz1 color1 ... xn yn zn nxn nyn nzn colorn) where each successive x, y z, nx, ny, nz and color are the vertices of the polygon and the normal at that vertex, and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. There must be at least three x, y, z, nx, ny, nz, color seven-tuples in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the entire group.

Function: **scene-draw-multicolor-3d-convex-polygon-diffuse** *[krma]scene group vertices*

Immediate-mode function, draws a multicolored 3d convex polygon, returns no values. Displays with diffuse shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, vertices must be of the form (list x0 y0 z0 nx0 ny0 nz0 color0 x1 y1 z1 nx1 ny1 nz1 color1 ... xn yn zn nxn nyn nzn colorn) where each successive x, y z, nx, ny, nz and color are the vertices of the polygon and the normal at that vertex, and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. There must be at least three x, y, z, nx, ny, nz, color seven-tuples in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-multicolor-3d-convex-polygon-primitive-flat** *[krma] scene group model-matrix vertices*

Retained-mode function, returns a handle for a multicolored 3d convex polygon primitive. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), vertices should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x, y and z's represent a vertex of the polygon and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. There must be at least three x, y, z, color quads in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-multicolor-3d-convex-polygon-flat** *[krma] scene group vertices*

Retained-mode function, adds a multicolored 3d convex polygon to the draw lists, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, vertices must be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where each successive x, y z and color are the vertices of the polygon, and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. There must be at least three x, y, z, color quads in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the entire group.

Function: **scene-draw-multicolor-3d-convex-polygon-flat** *[krma] scene group vertices*

Immediate-mode function, draws a multicolored 3d convex polygon, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, vertices should be of the form (list x0 y0 z0 color0 x1 y1 z1 color1 ... xn yn zn colorn) where the x, y and z's represent a vertex of the polygon and must be real numbers, color values can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer. There must be at least three x, y, z, color quads in vertices. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-textured-3d-triangle-list-primitive-flat** *[krma] scene group model-matrix texture color vertices*

Retained-mode function, returns a handle for a textured 3d triangle list primitive. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices is a list must be composed of sub-sequences of x, y, z, u and v, where u and v are the normalized texture coordinates at that vertex. There must be at least three sub-sequences of x, y, z, u and v to make a triangle and additional triangles come in 3 sub-sequences each. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the triangle list, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-textured-3d-triangle-list-flat** *[krma] scene group texture color vertices*

Retained-mode function, adds a textured 3d convex polygon to the draw-lists, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices is a list and must be composed of sub-sequences of x, y, z, u and v, where u and v are the normalized texture coordinates at that vertex. There must be at least three sub-sequences of x, y, z, u and v to make a triangle and additional triangles come in 3 sub-sequences each. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Dispatches actual work to render thread. To delete the polygon, you must delete the entire group.

Function: **scene-draw-textured-3d-triangle-list-flat** *[krma] scene group texture color vertices*

Immediate-mode function, draws a textured 3d convex polygon, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices is a list and must be composed of sub-sequences of x, y, z, u and v, where u and v are the normalized texture coordinates at that vertex. There must be at least three sub-sequences of x, y, z, u and v to make a triangle and additional triangles come in 3 sub-sequences each. Vertices should be oriented counter clockwise, according to the right-hand-rule, so that the front face is out/up. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-textured-3d-triangle-strip-primitive-flat** *[krma] scene group model-matrix texture color vertices*

Retained-mode function, returns a handle for a textured 3d triangle strip primitive. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), texture should be a texture such as return from make-vulkan-texture, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices should be of the form (list x0 y0 z0 u0 v0 x1 y1 z1 u1 v1 ... xn yn zn un vn) where the x, y and z's represent a vertex of the polygon and must be real numbers. u and v are the normalized texture coordinates of that vertex, and must be real numbers between zero and one. There must be at least three x, y, z, u and v quints in vertices. Dispatches actual work to render thread. To delete the triangle strip, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-draw-textured-3d-triangle-strip-flat** *[krma] scene group texture color vertices*

Immediate-mode function, draws a textured 3d triangle strip, returns no values. Displays with flat shading. Required arguments: scene must be of type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, vertices should be of the form (list x0 y0 z0 u0 v0 x1 y1 z1 u1 v1 ... xn yn zn un vn) where the x, y and z's represent a vertex of the polygon and must be real numbers. u and v are the normalized texture coordinates of that vertex, and must be real numbers between zero and one. There must be at least three x, y, z, u and v quints in vertices. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-filled-sphere-primitive-diffuse** *[krma] scene group model-matrix color origin-x origin-y origin-z radius light-position &optional resolution*

Retained-mode function, returns a handle for a filled sphere primitive. Displays with diffuse shading. scene must be of the type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, origin-z, origin-y and origin-z must be real numbers, radius must be a positive real number, light-position should either be nil or a 3d-vectors:vec3, resolution should be a positive integer and defaults to 64. Dispatches actual work to render thread. To delete the sphere, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-filled-sphere-diffuse** *[krma] scene group color origin-x origin-y origin-z radius &optional resolution*

Retained-mode function, adds a filled sphere to the draw-lists, returns no values. Displays with diffuse shading. scene must be of the type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, origin-z, origin-y and origin-z must be real numbers, radius must be a positive real number, resolution should be a positive integer and defaults to 64. Dispatches actual work to render thread. To delete the sphere, you must delete the entire group.

Function: **scene-draw-filled-sphere-diffuse** *[krma] scene group color origin-x origin-y origin-z radius &optional resolution*

Immediate-mode function, draws a filled sphere, returns no values. Displays with diffuse shading. scene must be of the type krma-essential-scene-mixin, group must be a non-null atom, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, origin-z, origin-y and origin-z must be real numbers, radius must be a positive real number, resolution should be a positive integer and defaults to 64. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **scene-add-text-primitive** *[krma] scene group model-matrix font color pos-x pos-y string*

Retained-mode function, renders and returns a handle for a text primitive. scene must be of the type krma-essential-scene-mixin, group must be an atom, possibly nil (meaning not associated with a group), model-matrix must either be a 3d-matrices:mat4 or nil (nil effectively means identity), font is a font object as returned by vulkan-make-font, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, pos-x and pos-y represent the top left corner of the text and must be real numbers, string is the string you wish to render. Dispatches actual work to render thread. To delete the text, you must delete the primitive using the handle or delete the entire group, if any.

Function: **scene-add-text** *[krma] scene group font color pos-x pos-y string*

Retained-mode function, adds text to the draw lists, returns no values. scene must be of the type krma-essential-scene-mixin, group must be a non-null atom, font is a font object as returned by vulkan-make-font, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, pos-x and pos-y represent the top left corner of the text and must be real numbers, string is the string you wish to render. Dispatches actual work to render thread. To delete the text, you must delete the entire group.

Function: **scene-draw-text** *[krma] scene group font color pos-x pos-y string*

Retained-mode function, draws text, returns no values. scene must be of the type krma-essential-scene-mixin, group must be a non-null atom, font is a font object as returned by vulkan-make-font, color can either be a 4 component vector who's elements are real numbers between zero and one, or a 32 bit unsigned integer, pos-x and pos-y represent the top left corner of the text and must be real numbers, string is the string you wish to render. Performs work in current thread, which should be the render thread. Effects of this function only last for the current frame.

Function: **reinstance-primitive-1***[krma] draw-data new-handle handle &key group model-matrix point-size line-thickness color-override light-position font*

Retained-mode function. Re-instances a primitive given a handle with the option to set a new group, model-matrix, point-size, line-thickness, color-override, light-position and/or font. References the same vertices in the draw-lists. Performs work in current thread, which should be the render thread.

Function: **reinstance-primitive** *[krma] scene handle &key group model-matrix point-size line-thickness color-override light-position font*

Retained-mode function, returns a handle. Re-instances a primitive given a handle with the option to set a new group, model-matrix, point-size, line-thickness, color-override, light-position and/or font. References the same vertices in the draw-lists. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **primitive-set-color-1** *[krma] draw-data handle color*

Retained-mode function. Returns no values. Sets the color override of the primitive, normally renderer uses the vertex color.   Performs work in current thread, which should be the render thread.

Function: **primitive-set-color***[krma] scene handle color*

Retained-mode function. Returns no values. To be run in a thread outside of the render thread. Sets the color override of the primitive, normally renderer uses the vertex color.   To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **primitive-set-light-position-1***[krma] draw-data handle pos*

Retained-mode function. Returns no values. Sets the light position of the primitive, normally renderer uses the scene light position, unless group light position is set. Performs work in current thread, which should be the render thread.

Function: **primitive-set-light-position***[krma] scene handle pos*

Retained-mode function. Returns no values. Sets the light position of the primitive, normally renderer uses the scene light position, unless group light position is set. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **primitive-set-transform-1***[krma] draw-data handle matrix*

Retained-mode function. Returns no values. Sets the model-matrix of the primitive, renderer composes the primitive model matrix with the group model matrix. Performs work in current thread, which should be the render thread.

Function: **primitive-set-transform***[krma] scene handle matrix*
Retained-mode function. Sets the model-matrix of the primitive, renderer composes the primitive model matrix with the group model matrix. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **primitive-apply-transform-1***[krma] draw-data handle matrix*
Retained-mode function. Returns no values. Applies the matrix to the existing model-matrix of the primitive, renderer composes the primitive model matrix with the group model matrix. Performs work in current thread, which should be the render thread.

Function: **primitive-apply-transform***[krma] scene handle matrix*
Retained-mode function. Returns no values. Applies the matrix to the existing model-matrix of the primitive, renderer composes the primitive model matrix with the group model matrix. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **primitive-set-line-thickness-1***[krma] draw-data handle thickness*
Retained-mode function. Returns no values. Sets the line-thickness of the primitive. Performs work in current thread, which should be the render thread.

Function: **primitive-set-line-thickness***[krma] scene handle thickness*
Retained-mode function. Returns no values. Sets the line-thickness of the primitive. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **delete-primitive-1***[krma] draw-data handle*
Retained-mode function. Returns no values. Deletes the primitive with the handle. Performs work in current thread, which should be the render thread.

Function: **delete-primitives***[krma] draw-data list-of-handles*
Retained-mode function. Returns no values. Deletes the primitives with the handles.   To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **delete-primitive***[krma] scene handle*
Retained-mode function. Returns no values. Calls delete-primitives with (list handle).   To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **delete-groups-1***[krma] draw-data list-of-groups*
Returns no values. Deletes the groups in list-of-groups. Performs work in current thread, which should be the render thread.

Function: **delete-groups***[krma] scene list-of-groups*

Retained-mode function. Returns no values. Deletes the groups in list-of-groups. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **delete-group***[krma] scene group*

Retained mode function. Returns no values. Calls delete-group on (list group). To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **group-set-color-override-1***[krma] draw-data group color*

Returns no values. Sets the color override of the group. Performs work in current thread, which should be the render thread.

Function: **group-set-color-override***[krma] scene group color*

Returns no values. Sets the color override of the group. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **group-set-model-matrix-1***[krma] draw-data group matrix*

Returns no values. Sets the model matrix of the group. Performs work in current thread, which should be the render thread.

Function: **group-set-model-matrix***[krma] scene group matrix*

Returns no values. Sets the model matrix of the group. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **group-apply-model-matrix-1***[krma] draw-data group matrix*

Returns no values. Transforms the existing model matrix of the group with matrix and set it to the group model matrix. Performs work in current thread, which should be the render thread.

Function: **group-apply-model-matrix***[krma] scene group matrix*

Returns no values. Transforms the existing model matrix of the group with matrix and set it to the group model matrix. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **group-set-light-position-1***[krma] draw-data group position*

Returns no values. Sets the light position of the group. Performs work in current thread, which should be the render thread.

Function: **group-set-light-position***[krma] scene group position*

Returns no values. Sets the light position of the group. To be run in a thread outside of the render thread. Dispatches actual work to render thread.

Function: **ensure-group-1***[krma] draw-data group*
Returns no values. Ensures the existence of a group if one chooses to set the properties of the group before actually calling "draw-" or "add-" functions.

Macro: **rm-dispatch-to-render-thread***[krma] (scene draw-data-var) &body body*
Returns no values. Provides access to the code in body to the retained-mode draw-data and executes in the render thread.   To be executed in a thread outside of the render thread.

Accessor Method: **im-draw-data***[krma] scene*
Returns the immediate-mode draw-data given the scene. Useful for group mutators run in the render thread.