# Description of S2PLOT Omegalib Module

*This document is meant to accompany the communication diagram attached.*

**S2Module**

- The s2plot module (s2Module) is the interface between Omegalib and the s2plot library.
- The s2Module is a descendant class of Omegalib's EngineModule. The s2Module constructor instantiates a new Omegalib EngineModule and it is added to the list of modules in the Omegalib Engine class.
- The Engine calls update on each EngineModule. Each EngineModule will then update its data structures every frame.
- The Engine and the modules, including s2Module, are contained in the CPU side thread, the CPU side thread completes all computations and data updates which are not associated with rendering.
- The module creates a RenderPass (s2RenderPass) which is attached to the Omegalib Renderer.
- The s2RenderPass has a callback to s2Module's draw function which then iterates over the primitive list calling each primitives draw function.
- Geometry references are kept as s2Geom objects in a vector for retrieval of pointers by the users callbacks and for deletion and update of geometry.
- The module uses three VBOs two to contain vertices and one to contain indices.
- One vertex VBO is for the CPU side thread to update and sort, while the other vertex VBO is for the GPU side thread to render. Once updating and sorting are completed the pointers are swapped.
- Vertices contain position, colour and normal as GL float types.
- The VBOs are bound to the graphics card at start up, transferring the data before rendering starts.
- To reduce the load on the CPU side, if the camera position has not changed significantly a sort is not triggered.
- When the camera position has changed beyond a threshold the primitives vector is re-sorted based on the current camera position and the position of the primitives vertices.
- The module has the add and delete object functions, which also add and delete the objects primitives from the primitives vector.
- When geometry is added the data structures are updated immediately and the VBOs transferred to the graphics card.
- Deletions are not performed on the vertices and indices vectors until more than 10% of the data in the data structures are stale.

### S2RenderPass

The RenderPass is contained in the rendering thread, its function is to make openGL calls and render data to the frustum. It is separate from the CPU thread

The RenderPass has two functions:

- Call the draw function in the module. This is a hang up of the Omegalib architecture design and could not be avoided. It achieves this by the module creating the RenderPass and passing a reference (pointer) to itself to the RenderPass through the Omegalib function setUserData(this). The RenderPass then retrieves the pointer to the module in its initialize function, which is called by the Renderer prior to the first render pass. This enables the RenderPass access to the draw function in the module.
- Setup the VBO buffers and shader. The RenderPass calls the initialiseGL function in the module on the initial render pass, this allows the binding of the data buffers and the creation and loading of the shader program. The initialiseGL function **must be called here** due to the OpenGL context not being created before this point in Omegalib. This is a hacky fix implemented in Omegalib because the initialise function of the users application is called before the display system (equalizer in this case) has been initialised. Subsequently, the buffers can not be bound or the shader program loaded until after this has happened.

### S2Factory

- The factory is used to separate the s2plot API from the module, it acts as a wrapper class enabling the separation of the interface from the implementation.
- The factory creates the object and then calls the modules add object function to add the geometry to the system, returning an integer id to the user to access the object in their callbacks.