Render state tracking

Relevant issue: https://github.com/defold/defold/issues/6179

Brief

Currently, when executing a render list (or calling draw) from the renderer, we don't keep track of what the state is that has already been set for certain parameters (blending, stencil ops, face winding). This can cause issues between render objects A and B where A sets a few of these parameters and B doesn't. In this case B will use the same render state as A even though that is usually not the wanted effect.

Furthermore, since we don't track any state changes in the renderer, there is also the potentially added cost of triggering unnecessary state changes between draw calls since it's really when the actual draw call happens that we need to have an updated draw state.

I.e if you set render.set_blend_func(...) 10x times from the render script before a draw call, we will still issue 10x glBlendFunc calls even though we only need the last one before drawing.

Technical

To solve this issue, we could ask the graphics module to give us the latest set value and keep track of what values have changed during the execution of the draw list. For OpenGL, this usually means calling glGet(..) with the parameter you want to get, which is very costly because of cpu/gpu synchronization. In vulkan (and other modern graphics adapters), we already have this functionality available through a 64-bit "pipeline" variable which can easily be compared in order to know which states that need to be updated. The pipeline variable is updated when the renderer calls a certain draw state function, such as Enable/Disable state, or setting the blending function.

Implementation

- Expose a way to get draw state values from the graphics adapters before executing the list of render objects
    - For vulkan and "other" graphics adapters, this should be easy since we already have a small representation of the current render state
        - Note: This representation must be expand to support all the state changes we all, currently the pipeline objects doesn't store values for separate stencils for examples, which the renderobjects technically can set (only rive so far from what I know)
    - For OpenGL, we need to add a similar design either by:
        - Using the current pipeline structure setup
            - Pros: We can apply the pipeline before drawing and only set the states that have actually changed between two draw calls

        - Cons: A little bit of work to make sure it's correct, the OpenGL state is the "ground truth" so we don't want to make mistakes here and have an out-of-sync representation
        - Note: If people do some very special OpenGL rendering outside of our renderer, the state can get out-of-sync, but I don't know if we can solve that gracefully other than hope that people don't mess about with the state.
    - Exposing via glGet
        - Pros: Easy to implement, it's just a call for the thing you want to query
        - Cons: Slow! Probably not a good idea at all..