polymesh and subdmesh

In addition to the standard attributes and user attributes supported on both `polymesh` and `subdmesh` locations, KtoA adds a few additional attributes it can translate.

polymesh and subdivision

`polymesh` locations can have subdivision activated (in the `ArnoldObjectSettings` node) and KtoA will allow them through as subdivision meshes, but with a warning. It is preferred to use the `subdmesh` location type.

Additional Attributes

All of the mesh-related attributes from `ArnoldObjectSettings` (stored in the `arnoldStatements` attribute group) are applied. In addition, the following are recognized:

- `geometry.point.Pref`: the same size as `geometry.point.P`, this is a per-vertex reference position typically used for applying projective textures or noise at a bind-pose and having it stick.
- `geometry.creaseIndices`: there are two modes for this attribute:
    - An integer array of pairs of indices referencing vertices (points from `geometry.point.P`). E.g. if a subdmesh has 600 vertices, `creaseIndices` with values 1, 2 will make a creased edge from vertex 1 to vertex 2.
    - Runs of vertex indices, each run length indicated by geometry.creaseLengths. E.g. if a subdmesh has 600 vertices, geometry.creaseLengths has a three values 4, 3, 3, and creaseIndices has values 1, 2, 5, 7, 2, 6, 7, 9, 10, 12, then it will make creased edges from vertices 1 to 2, 2 to 5, 5 to 7, 2 to 6, 6 to 7, 9 to 10, and 10 to 12. Note that the runs can be conceptually thought of as 1,2,5,7; 2,6,7; 9,10,12.
- `geometry.creaseLengths`: if present, `geometry.creaseIndices` contains runs of vertex indices instead of being restricted to pairs of vertex indices. When not present, it is implied that each length is 2, producing pairs of indices.
- `geometry.creaseSharpness`: the OpenSubdiv-compatible float crease sharpness, one per pair or run of crease indices. If `creaseLengths` is present, there will be the same number of crease sharpnesses as entries in `creaseLengths` and otherwise there will be one per pair of indices if not. This will produce a hard/perfect crease up to the subdivision level that matches, and then will soften for any subdivision levels beyond that. E.g. if the subdivision level is set to 4, and a crease sharpness is set to 2.5, then it will produce a hard crease through subdivision levels 1 and 2, a half-hard crease on subdivision level 3, and a regular subdivision on level 4.
- `geometry.cornerIndices`: an array of vertex indices indicating which should be sharpened.
- `geometry.cornerSharpness`: like `creaseSharpness` but applies to a vertex to create sharp corners. This will have one float value per index in `cornderIndices`.