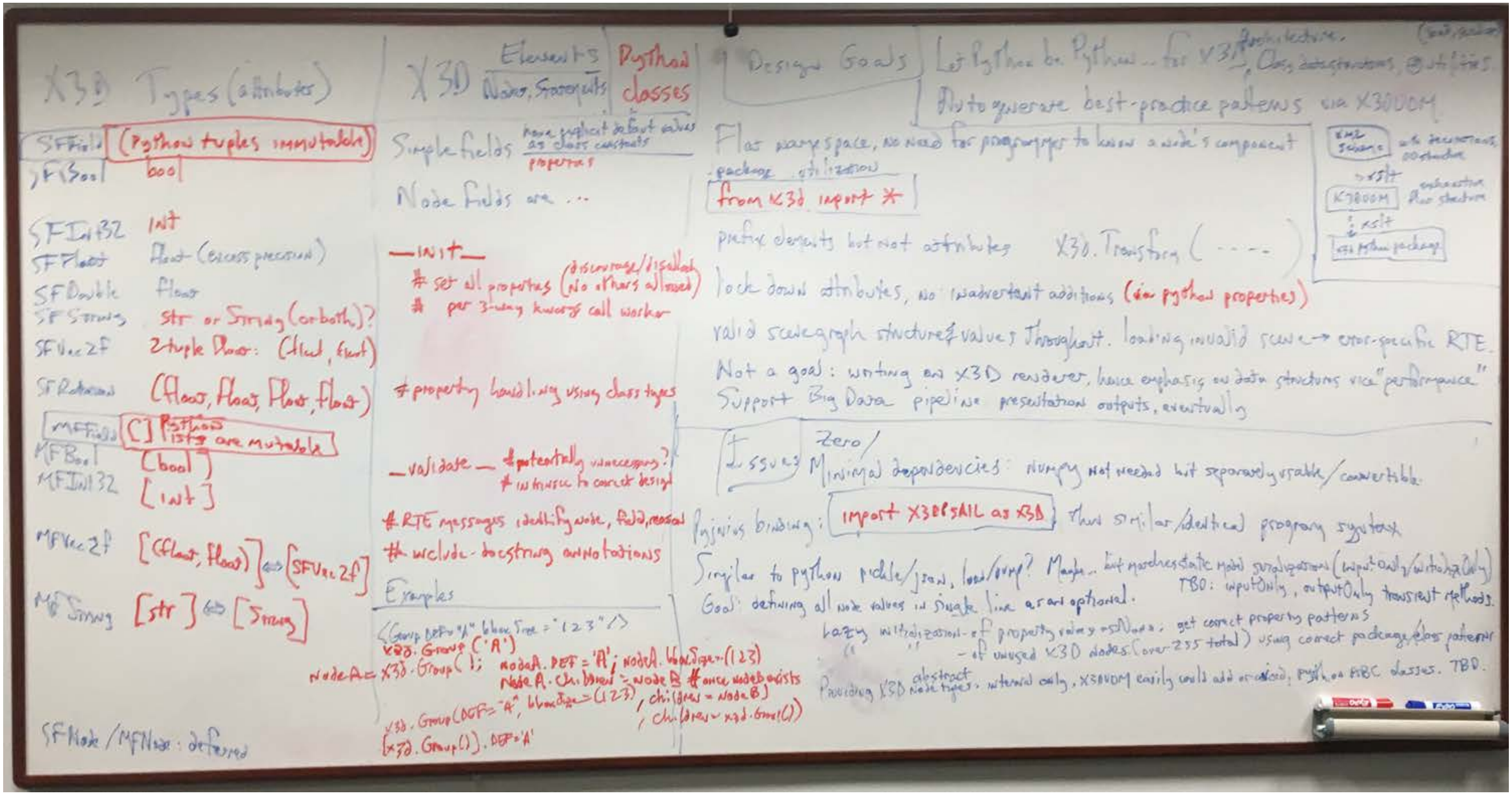
X3D Types (attributes)
SFField (Python tuples immutable)
SFBool bool
SFInt32 int
SFFloat float (excess precision)
SFDouble float
SFString str or String (or both)?
SFVec2f 2-tuple float: (float, float)
SFRotation (float, float, float, float)
MFField [] Python lists are mutable
MFBool [bool]
MFInt32 [int]
MFVec2f [(float, float)] ⇔ [SFVec2f]
MFString [str] ⇔ [String]
SFNode / MFNode: deferred
X3D Elements Nodes, Statements
Python classes
Simple fields
Node fields are ...
__init__
set all properties (No others allowed)
property handling using class types
validate
RTE messages identify node, field, reason
include docstring annotations
Examples
X3D.Group('A')
[x3d.Group()].DEF='A'
Design Goals
Let Python be Python ... for X3D
Autogenerate best-practice patterns via X3DUOM
from x3d import *
prefix elements but not attributes X3D.Transform(----)
lock down attributes, no inadvertent additions (via python properties)
valid scenegraph structure & values throughout. loading invalid scene → error-specific RTE.
Not a goal: writing an X3D renderer, hence emphasis on data structures vice "performance"
Support Big Data pipeline presentation outputs, eventually
Issues
Zero/
Minimal dependencies: numpy not needed but separately usable/convertible.
binding: import X3DJSAIL as X3D thus similar/identical program syntax
Similar to python pickle/json, load/dump?
TBD: inputOnly, outputOnly transient methods.
Goal: defining all node values in single line as an option.
get correct property patterns
of unused X3D nodes (over 255 total)
Providing X3D abstract node types: internal only, X3DUOM easily could add or avoid; python ABC classes. TBD.