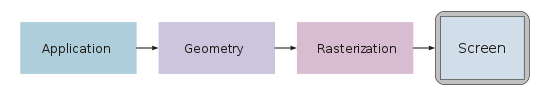
Diagram

Description automatically generatedText

Description automatically generated with medium confidence



Diagram

Description automatically generated

* Homogeneous coordinates

4x4 matrices

Homogeneous coordinates are ubiquitous in computer graphics because they allow common vector operations such as [translation](https://en.wikipedia.org/wiki/Translation_(geometry)), [rotation](https://en.wikipedia.org/wiki/Rotation_(mathematics)), [scaling](https://en.wikipedia.org/wiki/Scaling_(geometry)) and [perspective projection](https://en.wikipedia.org/wiki/Perspective_projection) to be represented as a matrix by which the vector is multiplied. By the chain rule, any sequence of such operations can be multiplied out into a single matrix, allowing simple and efficient processing

* Some interesting things
  + Frame rate independent animation
* Talk about optimisation
* But before we talk about optimising we need to measure

Benchmarking

Deltatime VIM

Data buffer strategies

2D / 1D DVR etc

Profiling massively affects performance (x10!!!)

* Know when to stop benchmarking and optimising (have a goal in mind, for me it was > 60 FPS for the teapot)
* But do identify the hot spots
* Don’t bother optimising things that look slow but might not be called often enough to make a difference
* Profiling will cause slowdowns! (Schroedingbug?)
* Simple is better! Complicated scheme with DVRs and double buffering offered about the same performance but was much harder to work with. Trying to optimise out the pixel buffer was premature optimisation
* Picture stuff is slow, can do some tricks like cache the rect
* LabVIEW is weird, sometimes the code will suddenly run slowly for no apparent reason, restarting LabVIEW will fix it!

Diagram

Description automatically generated

Naïve implementation is actually quickest with real world data

But clever low level technique is quickest with synthethic data.

Can draw the wrong conclusions from benchmarking and optimisation