Document number:
Revises:
Date: 2019-10-07
Project: ISO JTC1/SC22/WG21: Programming Language C++
Audience: LEWGI, SG6
Reply to: Vincent Reverdy
University of Illinois at Urbana-Champaign
vince.rev@gmail.com

Towards a standard unit systems library

Abstract

Time may finally have come to design a standard library to handle units in C++. However, unit systems are far more complex entities than one may first think. Designing a generic and robust standard library to handle unit systems is likely to be a challenging task. Any mistake on the fundamental underlying mathematical and physical concepts will inevitably blow up at some point for some users. In this preliminary paper we list some of the requirements that should be met by a standard unit systems library.

Contents

1 Requirements

- The mathematical and physical concepts should match the ones defined by the experts in metrology from the Bureau International des Poids et Mesures (BIPM) which is in charge of the definition of the International System of Units. The general C++ standardization policy is to rely on existing practices, and the BIPM has been working for years to standardize definitions of the mathematical and physical concepts involved in unit systems.
- Even if the standard library only provides one system of units, users should be able to define their own system of units using the same concepts and machinery. This is important to maximize interoperability and avoid conceptual mistakes being made in many user-defined units libraries.
- The library should provide both compile-time and runtime units since manipulating units at runtime is a very common task for many scientific and engineering softwares.
- Units should be defined outside of unit systems, but should be injected in unit systems before any conversion can take place. As an example, the concept of a meter is independent from the SI or the CGS system, but it only makes sense to use it in conversion within a given unit systems. Unit systems should form closed universes independent from each other.
- The library should handle the fact that two different kind of quantities, such as torque and work, should be kept separate even if they have the same dimension. This is also true for dimensionless quantities.
- As a result, for the most genericity, the association between a unit and a dimension should be system-dependent. As surprising as it seems, kilograms model mass only if mass is a fundamental dimension of the underlying physics model of the unit system. Some authors may consider that in geometrized unit systems length, time, and mass all correspond to the same dimension.
- Dimensions should not necessarily be always simplified. For example, in cosmology, the Hubble parameter is commonly expressed in $\mathrm{km\,s^{-1}\,Mpc^{-1}}$ which is homogeneous to the inverse of a time. However, not simplifying kilometers and Megaparsecs may be important for users in that context.
- The concept of basic and derived units should be system-dependent too. In the CGS system, the gram is a basic unit, while in the SI system the corresponding basic unit is the kilogram.
- Relationships between units should at least include multiplicative prefixes of different bases (*e.g.* kilo, mega…or kibi, mebi…), linear functions (*e.g.* Fahrenheit/Celsius), and non-linear scales (*e.g.* dB). The library should handle unit conversions in a truly generic way. For example, it should be easy to define currency conversion functions whose rate is updated in real time.
- The implicit or explicit aspects of conversions should be used to avoid by all means implicit loss of precision. Adding miles and kilometers should probably be prevented without explicit conversions. Adding meters and kilometers can probably be implicit if it maximizes the precision. These rules should probably be handled by unit systems.
- Typing units and quantities should be easy and intuitive for users. The same is true for defining new unit systems and displaying units. Error messages at compile-time should be clear. This is not incompatible with a complex and rich concept and type system. Most of the intricacies should be left for advanced users as customization points. An Embedded Domain Specific Language (EDSL) for units relying on metaprogramming and reflection may be at least considered as a viable approach.
- The question of units for vectors and tensors should be kept in mind. An evolution path should

be available.

- The question of measurement uncertainties should also be kept in mind. The design of unit systems should not prevent the integration of more advanced metrology concepts in later revisions.
- Experts from a wide variety of application domains should be invited to share unusual units, quantities, dimensions or unit systems used in their domains. This would help to better define the limits in terms of genericity the standard library is aiming for. A preliminary list is proposed hereafter.

2 A list of interesting cases to be handled

- The SI system evolved over time. In 2019, a significant redesign of the base units took place.
- The value of physics constants such as the ones published by the NIST can evolve over time thanks to better instruments and metrology techniques.
- Torque and work can be seen as two different physics concepts even if they can be expressed in the same unit.
- The concept of space and time in General Relativity are different than in Newtonian physics. Unit systems usually assume an underlying physics framework that can vary from one system to another. Converting quantities expressed in the same unit but relying on incompatible physics models should not be possible.
- In General Relativity, measured quantities depend on the observer. Converting measurements from one observer to another requires to know the geometry of spacetime between the two observers.
- Units may depend of time and space. For example, because of inflation the value of US dollars evolve over time.
- Fahrenheit and Celsius conversions are more complicated than just a multiplicative factor.
- Angles may be considered diffently in different unit systems.
- Currencies conversion rates evolve over time.
- Decibels do not correspond to a linear scale.
- In cosmology, the Hubble parameter is commonly expressed in $\mathrm{km\,s^{-1}\,Mpc^{-1}}$ which is homogeneous to the inverse of a time, but the full unit convey extra information and should not be implicitly simplified.
- In optics, diopters are homogeneous to the inverse of a length. In spectroscopy, the wavenumber is also homogeneous to the inverse of a length. However, the two refer to different physics concepts.
- In CGS the basic mass unit is the gram, while in SI the basic mass unit is the kilogram.
- There are multiple definitions of the year. Some include leap seconds, some do not, some depend on astronomical variations…
- In the same way, in an astronomical sense, the duration of a day on Earth evolved over time.
- Metric ounces are an approximation of US dry ounces.
- SI has a unique prefix system. Some other systems like the United States customary units have different conversion factors depending on the dimension considered: inch/foot/yard conversions factors and ounce/pound conversion factor are not unified.

Readers with unusual unit systems or complicated scenarios are welcomed to contact the author of this paper who will keep this list updated in future revisions of this proposal.

3 Acknowledgements

This work has been made possible thanks to the National Science Foundation through the awards CCF-1647432 and SI2-SSE-1642411.