Astronomy Research Project Proposal
An Object-Oriented Generic Stellar Modelling and Astronomy Library in C#

Sammy Yousef
University of Western Sydney, Nepean
Prepared as part of the subject “Astronomy Written Project” of the Astronomy Internet Masters Program, 2002.
Email: syousef@bigfoot.com

1 Abstract
Most existing Astronomy programming software tends to focus on the solution of a specific problem. There are few if any flexible libraries of Astronomical models which may be applied to a wide variety of problems. Yet there exists a commonality in the widely accepted principles and models in Math, Astronomy and Physics that exist in Astronomy. The goal of the intended research project is the creation of a first generation library of general Astronomical models written in a modern object-oriented environment, which could be used as the basis of, or contribute to, a wide variety of software. The initial library will be demonstrated via two or three well-written pieces of educational software used to demonstrate the diversity and generality of the underlying astronomical models. Extensions of the project could see the library grow in complexity and diversity to fill a niche in astronomy software, particularly if release is permitted in the public domain.

2 Introduction
As computers have matured, Astronomers have used them as a tool to simulate systems and solve problems that were previously out of their reach due to the amount of intense manual computation previously required. However, the majority of astronomy software available is written using procedural languages such as BASIC, C and FORTRAN, and is structured to solve a single problem, rather than for reuse.

As the science of programming has matured, so too have the languages and programming techniques employed, driven mainly by the need for timely, reliable and practical large scale implementations required for the increasingly competitive, complex and fast paced business world. These newer techniques generally require more mastery than older procedural methodologies and have in large part been avoided by astronomers in favour of older, simpler techniques. This is unfortunate as, newer flexible modeling techniques allow for code that is not only more intuitive and more readable but also reusable in a variety of situations. A well-written object-oriented library could support the creation of a variety of computer programs in several areas of astronomy.

3 Stellar Modeling and Astronomy Library
The author proposes to create a well structured object-oriented library under Microsoft’s new C# programming language and .NET platform, using Visual Studio .NET. This is one of the latest and most mature environments available, having been re-written from the ground up to support Microsoft Windows™ programming through a flexible and thoroughly object-oriented approach, built with the idea in mind of having learnt from past attempts at creating such environments (e.g. Microsoft Visual C™ with Microsoft Foundation Classes™, or Sunsoft Java™). Speed and interoperability are adequate compared to predecessor object-oriented languages based on C such as C++ and Java (though Java’s portability is lacking).
A single set of classes, comprising the stellar class library, will be used to model a variety of Astronomical phenomenon. Additionally classes will be split into groups for modeling Maths, Physics, and Astronomy, with each group building on and using concepts of the next. Through the varied operation and generic nature of the object classes, a number of varied demonstration programs, solving a variety of problems in Astronomy will be written using the library to demonstrate the effectiveness of this approach.

4 Functionality, Objectives, and Time Constraints

Key to the success of the project within the constraints of the Astronomy Research Project subject are:

1) The content of the library contains Astronomically significant functionality
2) That the scale of the work is achievable within the timeframe of the subject

To achieve the above objectives, the Astronomical content of the Library must be discussed with an appropriate supervisor. In order to demonstrate the generic nature of the library at least 2 distinct and separate sets of functionality must be modeled using the same set of classes. This, in turn requires at least 2 distinct and separate sets of demonstration code, employing the underlying library functionality.

Distinct phenomena that could be chosen for modeling include:

1) The relationships of the H-R diagram (for main sequence stars) - radius, mass, luminosity. Perhaps spectral shape. Perhaps comparison of stars (see H-R Calc). Equations to model include:
   a) \( L/L_{\text{Sun}} \sim (M/M_{\text{Sun}})^{(4.0\pm0.02)} \) for \( 0.4M_{\text{Sun}} < M < 5M_{\text{Sun}} \)
   b) \( L/L_{\text{Sun}} \sim (T/T_{\text{Sun}})^{6.67} \) (Main Sequence Temperature Luminosity Relation)
   a2) \( L/L_{\text{Sun}} \sim (M/M_{\text{Sun}})^{(3.6\pm0.1)} \) for \( 5M_{\text{Sun}} < M < 40M_{\text{Sun}} \) (Main Sequence Mass-Luminosity)

2) Stellar magnitudes (absolute, apparent), brightness and luminosity conversions. Also comparison between 2 stars. Equations to model include:
   a) \( b = L/4\pi r^2 \) (Luminosity/brightness at a distance)
   b) \( b/b_2 = L/L_2(r_2/r_1)^3 \) (Comparative Luminosity/brightness of 2 stars at a distance)
   c) \( b/b_s = (10/d)^3 \) (Absolute brightness of a star, at 10 parsecs distance)
   d) \( m = -2.5\log b + K_1 \) (Magnitude of a star)
   e) \( m - M = 5\log(d) - 5 = 5\log(d/10) \) (Distance Modulus)
   f) \( m_1 - m_2 = -2.5\log(b_1/b_2) \) (Difference in magnitude/brightness)

3) Spectra and Blackbody Radiation. Wein’s law, temperature, luminosity, maximum wavelength. Equations to model include:
   a) \( v = f\lambda \) (Velocity/wavelength relation)
   b) \( f = 1/T \) (Frequency/Period relation)
   c) \( E = hc/\lambda \) (Energy of a photon)
   d) \( F = \sigma T^4 \) (Stefan-Boltzmann Law)
   e) \( L = 4\pi R^2\sigma T^4 \) (Luminosity)
   f) \( \lambda_{\text{max}} = W/T \) (Wein’s Law)

4) Stellar evolution simulation, similar to the STATSTAR program (Carroll and Ostlie 1996). Standard models of stellar evolution vary in complexity but all consists of an initial state, based on boundary conditions at the center \( (r = 0; L_0 = 0; \text{for } M_r = 0) \) and surface \( (P = 0; T = T_{\text{eff}}/2^{1/4}; \text{r = R}; L_0 = L; \text{for } M_0 = M) \) of the star and a series of iterative integrations to determine the state of the star from outer shell to core for each time step. The equations of stellar structure are a-d below:
   a) \( \frac{dM}{dr} = 4\pi r^2 \rho \) (Mass conservation)
   b) \( \frac{dP}{dr} = -GM\rho/r^2 \) (Hydrostatic equilibrium – the balance of pressure and gravity)
   c) \( T_{\text{eff}} = (L/4\pi R^2\sigma)^{1/4} \) (Eddington Approximation)
\[ \frac{dL}{dr} = 4\pi^2 \rho \]  
\hspace{2em} (Energy conservation)

5) Gravitational interactions (2-body, N-body) and the shape of small open clusters (eventually galaxies?). Some of the most basic equations include:

\begin{align*}
\text{a)} & \quad F = \frac{GM_1M_2}{r^2} \quad \text{(Universal Law of Gravitation)} \\
\text{c)} & \quad \sum \frac{dP}{dt} = 0 \quad \text{(Newton’s third law/ conservation of momentum)} \\
\text{b)} & \quad F = ma \quad \text{(Newton’s second law)} \\
\text{c)} & \quad \sum \frac{dL}{dt} = 0 \quad \text{(Conservation of angular momentum)}
\end{align*}

Fortunately, the proposed project for a generic Stellar Modeling and Astronomy Library is a highly scalable endeavor, on a number of fronts.

1) The number of areas to be modeled, and the depth at which they will be modeled is negotiable with a supervisor, with a minimum number of 2 areas required and any more than 3 being more than sufficient to demonstrate the library’s success.

2) While a graphical front-end is highly desirable for any demonstration code, it is not absolutely necessary to prove the workings of the library. Depending on time constraints one or more of the demonstration programs could be reduced to a command line client, allowing a focus on the Astronomical side of the code.

To further minimize risk, it is envisioned that the initial code library will:

1) Be limited to concepts covered in earlier subjects of the AIM program.

2) Be educational or demonstrative in nature, employing simplified models wherever possible. The result of the ARP project will therefore likely not be research grade software, though the resulting library may be extensible in such a direction.

5 Feasibility

A simple proof of concept accompanies this paper which shows 2 separate trivial command line programs, one for calculating the blackbody peak for a star based on its temperature or temperature based on peak wavelength, and the other for calculating the flux for a star (given radius, luminosity). Both are based on a common library the author has named GAMF (Generic Astronomy Modeling Framework). These are toy programs intended as a proof of concept only. Total time spent on the library is around 10-15 hours (including the Maths.Polynomial class, not actually used yet in the Astronomy routines). This proposal should therefore be well within reach of a 12 week effort. Note that the author works a commercial IT consultant, whose daily duties include object oriented computer programming.

6 Timeline

A brief outline of a 12 week project to build the proposed library, demonstration software and the required accompanying documentation are given. Modifications to the timeline may need to be made as a result of the outcome of the first 2 weeks of work.

Week 1  - Outline/negotiate astronomical content of the library and demonstration programs with instructor
Week 2  - Finalize outline from week 1, begin extending prototype to fulfill requirements.
Week 4  - Begin work on demonstration program 1
Week 5  - Begin work on demonstration program 2 (possibly also program 3)
Week 6  - Begin work on documentation of project and library
Week 8  - Complete library modeling. Bug fixes only from this point.
Week 9  - Complete demonstration programs. Bug fixes only from this point.
Week 10 - Thoroughly test code. Optimization of code if required.
Week 11 - Complete project documentation.
Week 12 - Present project

7 Library Prototype
A prototype of the library under the temporary name “Generic Astronomy Modeling Framework“ (GAMF) has been created, along with two demonstration programs and submitted with this report. The aim is to demonstrate:

1) The competency of the author in writing software, and in the use of the relatively new programming language and corresponding development environment.
1) That the project can be completed in the required framework.

8 Other Known Efforts
While other Astronomy libraries exist few are publicly available (as would be the intention with this library subject to UWS regulations). Fewer still are written in object-oriented languages, with generality or reuse in mind. Much of today’s software is single purpose, may have cumbersome or poorly written FORTRAN and/or BASIC components, and/or be likely to run only under a UNIX style operating system. A listing of existing software is beyond the scope of this document. However examples of well written reuse-oriented packages include AIPS++ (written in C++) for data reduction, and educational packages such as CLEA, which while not object-oriented, does show thought towards software reuse in the development of common simulated telescopes/sky views. This software is the exception and there is still plenty of scope for originality and contribution to Astronomy in the proposed Stellar Modeling and Astronomy code library.

9 Acknowledgements
I’d like to thank David Blank for his rapid and friendly feedback, instruction and guidance throughout the Astronomy Written Project Subject, and Alex Hons for his tireless work in support of the Astronomy Internet Masters program.

10 References
An Introduction to Modern Astrophysics
Bradley W. Carroll and Dale A. Ostlie 1996
ISBN 0-201-54730-9

Universe 5th Edition
Roger A. Freedman and William J. Kaufmann III
ISBN: 0716738236

The H-R Diagram
http://star-www.st-and.ac.uk/~pja3/as3003/structure01.pdf
H-R Calc for Windows (Software) Version 2.0
David C. Irizarry 1996

AIPS++ (Software)
http://aips2.nrao.edu/docs/aips++.html