Parallel & Distributed Computing: Lecture 3

Alberto Paoluzzi

October 2, 2018

1. Introduction to course projects 1/2

2. PLaSM crumbs

3. LAR crumbs

Introduction to course projects 1/2

Goals/constraints of course projects

1. Students have to produce a significant addition/enhancement to an opensource project of geometric modeling in Julia

Goals/constraints of course projects

1. Students have to produce a significant addition/enhancement to an opensource project of geometric modeling in Julia
2. Let you fork a project in GitHub, clone on the local machine, develop new files, say `topic` in 4 directories:
   src/topic.jl
   test/topic.jl
   doc/topic.md
   examples/topic.jl

Goals/constraints of course projects

1. Students have to produce a significant addition/enhancement to an opensource project of geometric modeling in Julia
2. Let you fork a project in GitHub, clone on the local machine, develop new files, say `topic` in 4 directories:
   src/topic.jl
   test/topic.jl
   doc/topic.md
   examples/topic.jl
3. At the end of course ask for PR (Pull Request)

Goals/constraints of course projects

1. Students have to produce a significant addition/enhancement to an opensource project of geometric modeling in Julia
2. Let you fork a project in GitHub, clone on the local machine, develop new files, say `topic` in 4 directories:
   src/topic.jl
   test/topic.jl
   doc/topic.md
   examples/topic.jl
3. At the end of course ask for PR (Pull Request)
4. Discuss by mail the new features

Goals/constraints of course projects

1. Students have to produce a significant addition/enhancement to an opensource project of geometric modeling in Julia
2. Let you fork a project in GitHub, clone on the local machine, develop new files, say `topic` in 4 directories:
   src/topic.jl
   test/topic.jl
   doc/topic.md
   examples/topic.jl
3. At the end of course ask for PR (Pull Request)
4. Discuss by mail the new features
5. If approved, the student project is pushed to the starting repo.

Plasm.jl and LinearAlbebraicRepresentation.jl

www.plasm.org

Geometric Programming: A Programming Approach to Geometric Design

https://github.com/cvdlab/Plasm.jl

https://github.com/cvdlab/LinearAlbebraicRepresentation.jl

Plasm.jl extensions IDEAS

Currently used only for visualization of geometric models (cellular complexes)

Possible project tasks

Recuperate the Backus' programming at Function Level, in a parallelized environment (either on Multicore CPUs or GPUs)

LinearAlbebraicRepresentation.jl IDEAS

Dimension-independent modeling of cellular complexes and geometric assemblies

Possible project tasks

- Optimize/parallelize some existing functions

LinearAlbebraicRepresentation.jl IDEAS

Dimension-independent modeling of cellular complexes and geometric assemblies

Possible project tasks

- Optimize/parallelize some existing functions
- Extend the library with spline curves and surfaces (functions defined piecewise by polynomials)

LinearAlbebraicRepresentation.jl IDEAS

Dimension-independent modeling of cellular complexes and geometric assemblies

Possible project tasks

- Optimize/parallelize some existing functions
- Extend the library with spline curves and surfaces (functions defined piecewise by polynomials)
- Complete the optimal Boolean operations implementation

PLaSM crumbs

design language PLaSM

PLaSM is a geometry-oriented extension of a subset of FL

FL Language

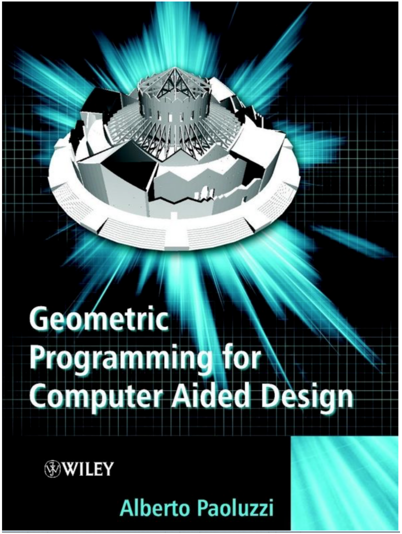
FL (programming at Function Level) is a language developed by the Functional Programming Group of IBM Research Division at Almaden (USA) [BWW90, BWWLA89]. The FL language, on the line of the Backus' Turing lecture [Backus78] introduces an algebra over programs and has an awesome expressive power.

PLaSM Language

PLaSM, (the Programming LAnguage for Solid Modeling) is a “design language” for geometric and solid parametric design, developed by the CAD Group at the Universities “La Sapienza” and “Roma Tre” [PS92, PPV95]. The language is strongly inFLuenced by FL. With few sintactical differences, it can be considered a geometric extension of a FL subset.

GP4CAD book

A. Paoluzzi, Geometric Programming for Computer-Aided Design, Wiley, 2003. (free download from `uniroma3.it` domain)



Pyplasm.jl

PLaSM was implemented in Common Lisp, Scheme, Python & C++

Current implementation is pyplasm

- download Python 3.6 from Anaconda3

run julia 0.6

```
julia> Pkg.clone("https://github.com/cvdlab/Plasm.jl")
julia> using Plasm
julia> using PyCall
julia> @pyimport pyplasm as p   # python interface to C++ kern
julia> p.VIEW(p.CUBE(0.5,0.5,0.5))       #testing C++ viewer
```

Pyplasm.jl

PLaSM was implemented in Common Lisp, Scheme, Python & C++

Current implementation is pyplasm

- download Python 3.6 from Anaconda3
- follow the instructions on the README file

run julia 0.6

```
julia> Pkg.clone("https://github.com/cvdlab/Plasm.jl")
julia> using Plasm
julia> using PyCall
julia> @pyimport pyplasm as p    # python interface to C++ kern
julia> p.VIEW(p.CUBE(0.5,0.5,0.5))       #testing C++ viewer
```

Pyplasm.jl

PLaSM was implemented in Common Lisp, Scheme, Python & C++

Current implementation is pyplasm

- download Python 3.6 from Anaconda3
- follow the instructions on the README file
- run the file `install_plasm.jl` here

run julia 0.6

```
julia> Pkg.clone("https://github.com/cvdlab/Plasm.jl")
julia> using Plasm
julia> using PyCall
julia> @pyimport pyplasm as p   # python interface to C++ ker
julia> p.VIEW(p.CUBE(0.5,0.5,0.5)       #testing C++ viewer
```

LAR crumbs

Go explore the package

https://github.com/cvdlab/LinearAlgebraicRepresentation.jl

git clone or fork

```
$ git clone https://github.com/cvdlab/LinearAlgebraicRepresent
```

add via the package manager to jour Julia environment

```
$ Julia

julia> Pkg.add("LinearAlbebraicRepresentation")
julia> using LinearAlbebraicRepresentation
julia> Lar = LinearAlbebraicRepresentation
help>  LinearAlbebraicRepresentation. <tab> <tab>
```

Look at the documentation

```
julia> Pkg.add("Documenter")
```

on the local LinearAlbebraicRepresentation.jl repo

```
$ cd <repo>/docs
$ julia6 make.jl          % julia 0.6

$ open build/index.html
```

look at the package documentation including internal docs of package functions.