Computing and Software Department, McMaster University

MG and MIS Examples

Dr. Spencer Smith

February 12, 2009

MG and MIS Examples

- Modules with external interaction
- Assignment 3
- Parnas's Rational Design Process (finish up)
- Examples
  - A mesh generator
  - A maze tracing robot (see WebCT document)

Administrative Details

- What has happened to discussion on WebCT?
- Assignment 3 deadlines
  - Code due by midnight Feb 23
  - E-mail your partner your code by Feb 24
  - Lab report due by the beginning of class Mar 2
- Midterm exam
  - March 4 during tutorial time
  - In T29/105, not our usual classroom

Assignment 3

- Working with vector spaces and inner product spaces
- Taking advantage of OCaml's treatment of functions as first class citizens
- Our vector space will be the space of real continuous functions
- The submission of part 1 will consist of the following files:
  - `test_vectorT.ml` to test `vectorT.ml`
  - `test_vectorSpaceT.ml` to test `vectorSpaceT.ml`
  - `test_innerProductSpaceT.ml` to test `innerProductSpaceT.ml`
  - `test_assig3.ml` to test all of the above using a test suite.
  - A `Makefile` to make the executable `test_assig`

Modules with External Interaction

- In general, some modules may interact with the environment or other modules
- Environment might include the keyboard, the screen, the file system, motors, sensors, etc.
- Sometimes the interaction is informally specified using prose (natural language)
- Can introduce an environment variable
  - Name, type
  - Interpretation
- Environment variables include the screen, the state of a motor (on, direction of rotation, power level, etc.), the position of a robot

External Interaction Continued

- Some external interactions are hidden
  - Present in the implementation, but not in the MIS
  - An example might be OS memory allocation calls
- External interaction described in the MIS
  - Naming access programs of the other modules
  - Specifying how the other module's state variables are changed
  - The MIS should identify what external modules are used

MIS for GUI Modules

- Could introduce an environment variable
- window: sequence [RES_H][RES_V] of pixelT
  - Where window[r][c] is the pixel located at row r and column c, with numbering zero-relative and beginning at the upper left corner
  - Would still need to define pixelT
- Could formally specify the environment variable transitions
- More often it is reasonable to specify the transition in prose
- In some cases the proposed GUI might be shown by rough sketches

Display Point Masses Module Syntax

Exported Access Programs

Routine name	**In**	**Out**	**Exc.**
DisplayPointMassesApplet		DisplayPointMassesApplet	
paint			

Display Point Masses Module Semantics

Environment Variables

win : 2D sequence of pixels displayed within a web-browser

DisplayPointMassesApplet():

- transition: The state of the abstract object ListPointMasses is modified as follows:
  ListPointMasses.init()
  ListPointMasses.add(0, PointMassT(20, 20, 10)
  ListPointMasses.add(1, PointMassT(120, 200, 20)
  ...

paint():

- transition *win* := Modify window so that the point masses in ListPointMasses are plotted as circles. The centre of each circles should be the corresponding x and y coordinates and the radius should be the mass of the point mass.

Assignment 3 Vector Module

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
new vectorT	real $\rightarrow$ real	vectorT	
getf		real $\rightarrow$ real	
eval	real, real, integer	sequence of real	deltaNeg, nstepsNeg
evalPrint	real, real, integer		deltaNeg, nstepsNeg

Vector Module Semantics

Environment Variables

screen : two dimensional sequence of positions on the screen, which each position holding a character

State Variables

f: real $\rightarrow$ real

Access Routine Semantics

eval $(startx, deltax, nsteps)$:

- output: $out := < f(startx), f(startx + deltax), f(startx + 2 \cdot deltax), ..., f(startx + nsteps \cdot deltax) >$
- exception: $exc := ((deltax < 0) \Rightarrow \text{deltaNeg} | (nsteps < 0) \Rightarrow \text{nstepsNeg})$

Vector Module Semantics Continued

evalPrint $(startx, deltax, nsteps)$:

- transition: The state of *screen* is modified so that the sequence returned by eval (*startx*, *deltax*, *nsteps*) is displayed.
- exception: $exc := ((deltax < 0) \Rightarrow \text{deltaNeg} | (nsteps < 0) \Rightarrow \text{nstepsNeg})$

Parnas's Rational Design Process (RDP)

- SRS
- MG
- Uses Hierarchy (produced after all MISs)
- For each module
  - MIS
  - MID
- Implementation
- Testing
- Very successfully used on projects such as
  - The Darlington Nuclear Reactor shutdown system
  - The A7-E fighter jet

RDP - Views

- As well as the MG, the modular decomposition should be displayed using a variety of views
- An obvious one is the Uses Hierarchy
- The Uses Hierarchy can be formed once the MIS for all modules is complete
- The Uses Hierarchy can be represented
  - Graphically (if it isn't too large and complex)
  - Using a binary matrix

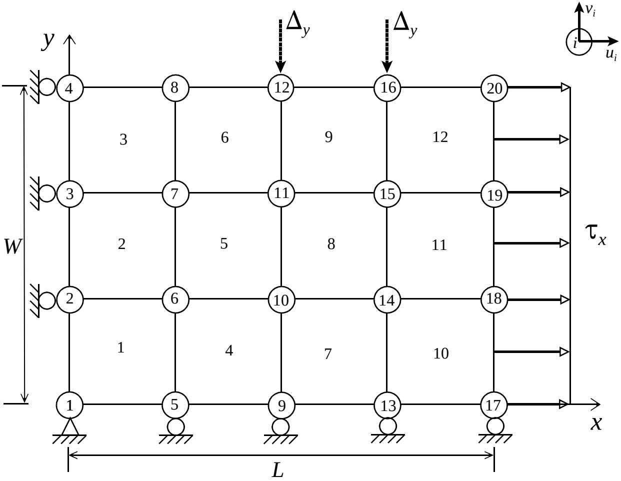
RDP - MG

- Criteria for a good secret
  - One module one secret (if possible)
  - Secrets should often be nouns (data structure, algorithm, hardware, ...)
  - Secrets are often phrased as "How to ... "

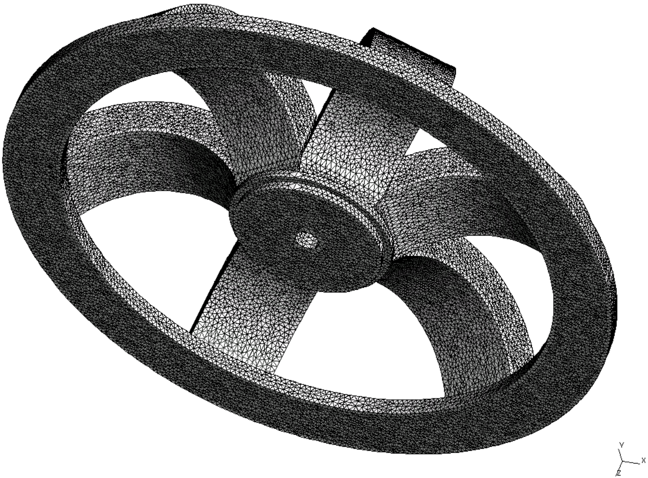
RDP - MID

- Another document that is often helpful is the Module Internal Design (MID) for each module
- The MID provides the implementation of the module; that is, it shows how we will deliver on what is promised in the MIS
- The MID is requirements for the code represented at a higher level of abstraction than the code
- The MID uses the syntax of the selected programming language
- The MID shows decisions like whether to use a static array, or dynamic memory allocation and pointers

Mesh Generator Simple Rectangular Geometry



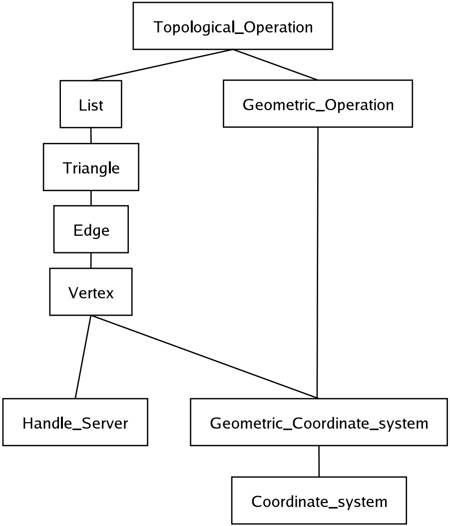
Mesh Generator Complex Circular Geometry



Mesh Generator Example: Design Goals

- Independent and flexible representation for each mesh entity
- Complete separation of geometric data from the topology
- The mesh generator should work with different coordinate systems
- A flexible data structure to store sets of vertices, edges and triangles
- Different mesh generation algorithms with a minimal amount of local changes

Mesh Generator Uses Hierarchy



Dr. v. Mohrenschildt's Maze Tracing Robot Example