TrML
Triangle Manipulation Language

Qishu Chen
Xuechen Feng
Lianhao Qu
Yu Wan
Wanqiu Zhang

Columbia University
December 2012
Introduction-TrML

- A simple programming language that allows user to express trigonometry concept, and construct/solve complex trigonometry problems.
  - C-like structure
  - Functional language

- Allow programmers to easily express trigonometry concepts and solve trigonometry problems.
TrML Tutorial

- There are two data types in TrML: value and triangle. Value is a floating point number, and triangle is a triangle in 2D plane.

@This is a comment
@assign 4.0 to value i
value i 4.0;
@assign three vertex values to triangle ABC
triangle ABC V [(1.1, 2.2),(3.3, 4.4),(5.5, 6.6)];
@assign three side-length values to triangle DEF
triangle DEF L [4.2, 3.5, 3.6];
TrML Tutorial

@Sample code: “Hello World!”
initialize:

rule:

operation:

prints("Hello \nWorld!\n");
TrML Tutorial

initialize:
value i 4.0;
value sum 0.0;
rule:
operation:
while(i > 0){
    sum = sum + i;
    i = i - 1;
}
prints("The sum of ");
printv(i);
prints(" is:");
printv(sum);
@the result should be: The sum of 4.0 is 10.0
Block Diagram

Block Diagram:

1. .trml file
2. Scanner/Lexer/AST
3. Parser
4. Compiler
5. Interpreter
6. Output (result)
AST

- program
  - initialization tuple
    - value declaration list
    - triangle declaration list
  - rule list
  - operation list
    - conditional operations, while operation, rule call, assignment
  - line, vertex declaration
Compiler

• Internal structure:
  • Rule table
  • Environment table
  • Operation variable
  • One stack register

• Code structure:
  • Environment variable followed by “rul” followed by rules definition followed by “opt” followed by operations definition
Interpreter

- Java Based
- Two arguments lists
  - Rule Argument, [rule counter]
  - Operation Argument, [operation counter]
- Global variable list
- Register stack
- 30+ instruction sets
Summary

**Main goals:**
- Acquire language and compiler design experience
- Have a coherent design and implement it correctly and in-time

**Outcome:**
- TrML is a comprehensive and simple language
- Implementation was finished before the deadline and the compiler follows the design specification
Summary

Suggestions for the future:

- Getting a head start:
  All group members were on the same page with starting early, but actually coordinating and forming the right pace for the team could still be improved.

- Pick a topic with passion:
  Pick a topic that most members are passionate about will make the experience worthwhile and enjoyable.
Testing code

- @keyw||d "initialize:" starts triangle initialization phase
  - initialize:
    - @initialize triangle with 2-D vertex location
      - triangle ABC V [(1.1, 2.2), (3.3, 4.4), (5.5, 6.6)];
    - @initialize triangle with line segment length
      - triangle DEF L [4.2, 3.5, 3.6];
  - value agl 10.0;
  - value opq 5.0;

- @keyw||d "rules:" starts rules construction phase
  - rules:
    - identical_triangle (triangle Tri_1, triangle Tri_2)
      - 
      - [triangle Tri_1.sideA == triangle Tri_2.sideA] &&
        [triangle Tri_1.sideB == triangle Tri_2.sideB] &&
        [triangle Tri_1.sideC == triangle Tri_2.sideC]
      - || [triangle Tri_1.sideA == triangle Tri_2.sideB] &&
        [triangle Tri_1.sideB == triangle Tri_2.sideC] &&
        [triangle Tri_1.sideC == triangle Tri_2.sideA]
      - || [triangle Tri_1.sideA == triangle Tri_2.sideC] &&
        [triangle Tri_1.sideB == triangle Tri_2.sideA] &&
        [triangle Tri_1.sideC == triangle Tri_2.sideB]
    - ) {true};

- @keyw||d "operations:" starts operation && calculation phase
  - operations:
    - agl = rule identical_triangle (triangle ABC, triangle ABC);
    - opq = 5.0;
    - printv (value agl);
    - if (value agl) {
        - prints ("ABC and DEF are identical");
      - }
    - if (1.0) {
      - 
      - prints ("is regular triangle");
    - }