CS 524 Task 4: Model-View-Controller Parser Architecture

This document provides an annotated description of the files provided in the archive for this task. Further details were addressed in this week's lectures. See the last page for download and installation instructions.

The input is case sensitive. Whitespace does not matter. The input and output filenames must be provided as arguments. The provide JAR serves as a completely working example.

The `define` command defines a three-dimensional box that may optionally contain other boxes. The variable identifier must be unique. Each has the following required elements:

- an arbitrary identifier, which is used only in the output.
- a volume in arbitrary units, defined as (length, width, height), corresponding to (*x*, *y*, *z*) in the coordinate system.
- a socket, which is the origin of the box. See the `exportToGnuplot` for more details.

In addition, it may have any number of subcomponents, each of which has its socket anchored at the `at` position on its parent box.

The `printXML` command takes the identifier from any `define` command and writes its output to the output file in XML format. See the examples for the format. It is an error to refer to a nonexistent identifier.

The `exportToGnuplot` command takes the identifier from any `define` command and writes its output to the screen in Gnuplot format. See the examples for the format. The socket of the box is anchored in the world by the second argument. It is an error to refer to a nonexistent identifier.

It is possible to print or export any component, which recursively applies to any subcomponents.

Test A

Description

This test generates a hull only.

Input File

```
// testA.blk: hull only

define a = (
            id="a.myHull"
            volume=[6,4,3]
            socket=[0,0,-1.5]
           );

printXML(a);

exportToGnuplot(a,[0,0,0]);
```

XML Output

Indentation is not expected.

```
<component identifier="a.myHull">
 <size>
  <triple x="6.0" y="4.0" z="3.0"/>
 </size>
 <socket>
  <triple x="0.0" y="0.0" z="-1.5"/>
 </socket>
</component>
```

Gnuplot Output

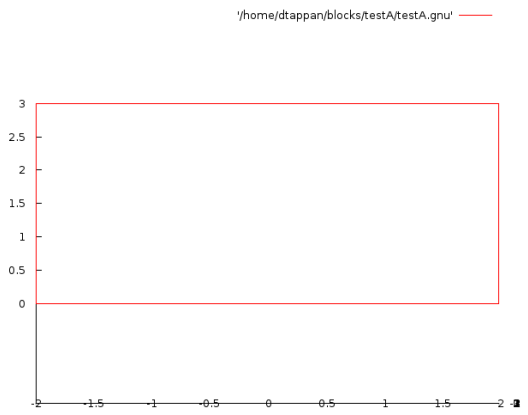
comments are required. Vertical spacing is critical.

```
# component [a.myHull] {
# top
-3.0 -2.0 3.0
3.0 -2.0 3.0
3.0 2.0 3.0
-3.0 2.0 3.0
-3.0 -2.0 3.0
```

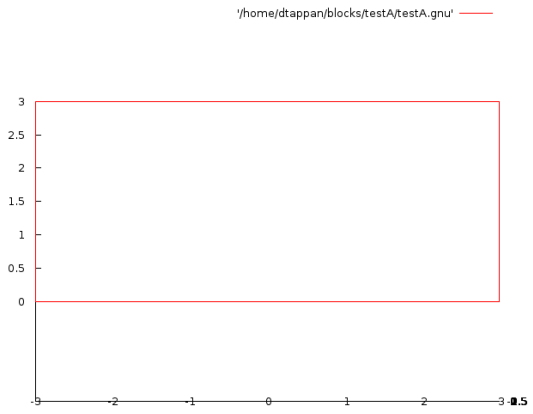
```
# bottom
-3.0 -2.0 0.0
3.0 -2.0 0.0
3.0 2.0 0.0
-3.0 2.0 0.0
-3.0 -2.0 0.0


# } component [a.myHull]
```

Gnuplot Rendering

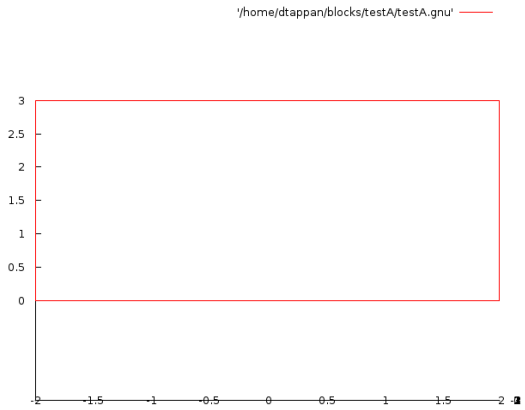
Top View



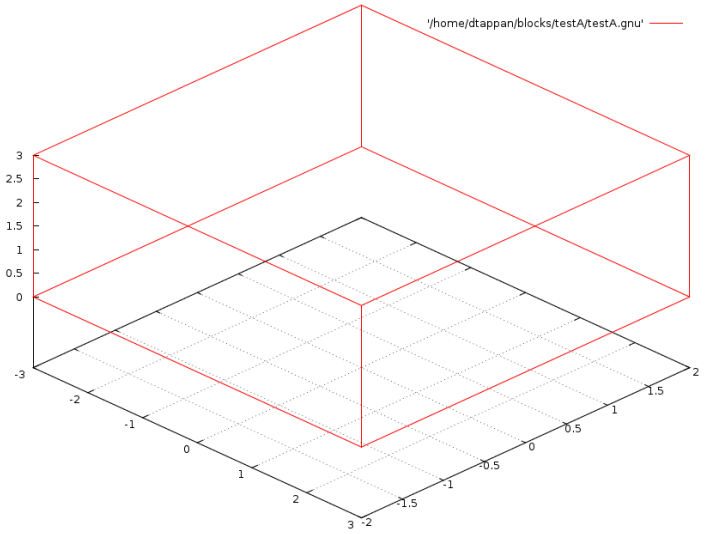
Side View



Front View



Orthogonal View



Test B

Description

This test generates a hull with a turret.

Input File

```
// testB.blk: hull with turret

define b = (
            id="b.myHull"
            volume=[6,4,3]
            socket=[0,0,-1.5]
            connectsTo
            (
             (
              id="b.myTurret"
              volume=[2,2,1]
              socket=[0,0,-0.5]
             ) at [-1,0,1.5]
            )
           );

printXML(b);

exportToGnuplot(b,[0,0,0]);
```

XML Output

```
<component identifier="b.myHull">
 <size>
  <triple x="6.0" y="4.0" z="3.0"/>
 </size>
 <socket>
  <triple x="0.0" y="0.0" z="-1.5"/>
 </socket>
 <connections>
  <mount>
   <component identifier="b.myTurret">
    <size>
     <triple x="2.0" y="2.0" z="1.0"/>
    </size>
    <socket>
     <triple x="0.0" y="0.0" z="-0.5"/>
    </socket>
   </component>
   <ball>
    <triple x="-1.0" y="0.0" z="1.5"/>
   </ball>
  </mount>
 </connections>
</component>
```

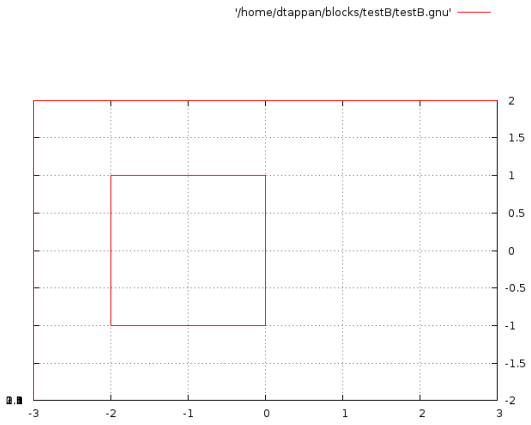
Gnuplot Output

```
# component [b.myHull] {
# top
-3.0 -2.0 3.0
3.0 -2.0 3.0
3.0 2.0 3.0
-3.0 2.0 3.0
-3.0 -2.0 3.0

# bottom
-3.0 -2.0 0.0
3.0 -2.0 0.0
3.0 2.0 0.0
-3.0 2.0 0.0
-3.0 -2.0 0.0


# subcomponents {
# component [b.myTurret] {
# top
-2.0 -1.0 4.0
```

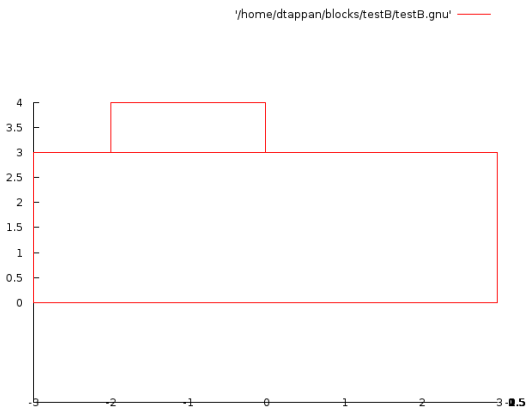
```
0.0 -1.0 4.0
0.0 1.0 4.0
-2.0 1.0 4.0
-2.0 -1.0 4.0

# bottom
-2.0 -1.0 3.0
0.0 -1.0 3.0
0.0 1.0 3.0
-2.0 1.0 3.0
-2.0 -1.0 3.0


# } component [b.myTurret]
# } subcomponents
# } component [b.myHull]
```

Gnuplot Rendering

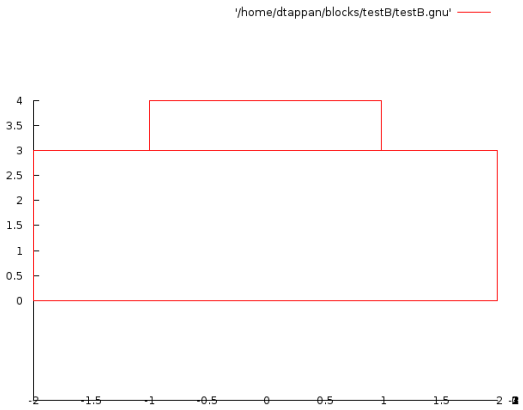
Top View



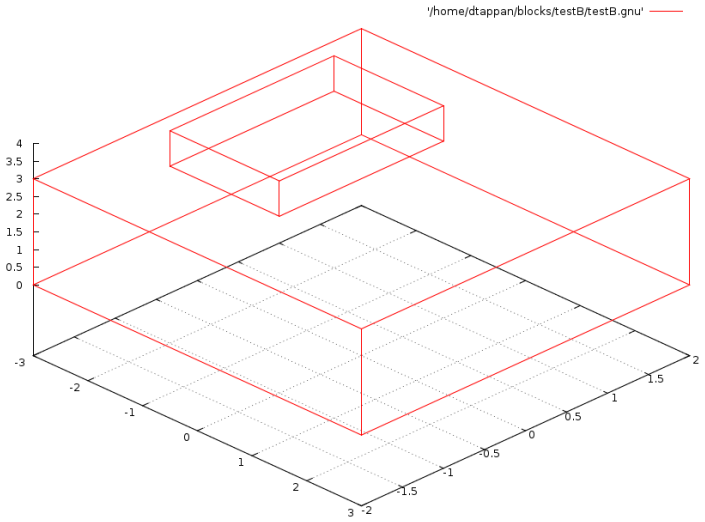
Side View



Front View



Orthogonal View



Test C

Description

This test generates a hull with a turret and a gun.

Input File

```
// testC.blk: hull with turret and gun

define c = (
            id="c.myHull"
            volume=[6,4,3]
            socket=[0,0,-1.5]
            connectsTo
            (
             (
              id="c.myTurret"
              volume=[2,2,1]
              socket=[0,0,-0.5]
              connectsTo
              (
               (
                id="c.myGun"
                volume=[5,0.5,0.5]
                socket=[-2.5,0,0]
               ) at [1,0,0]
              )
             ) at [-1,0,1.5]
            )
           );
printXML(c);
exportToGnuplot(c,[0,0,0]);
```

XML Output

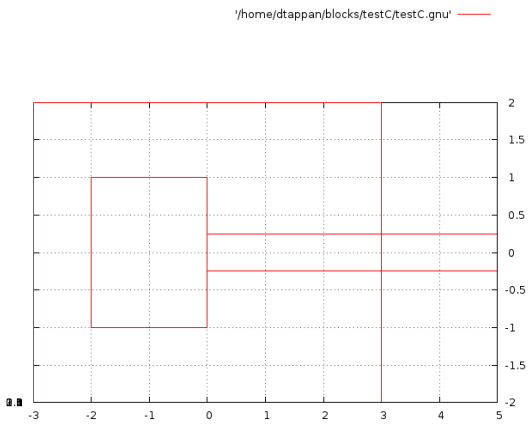
```
<component identifier="c.myHull">
 <size>
  <triple x="6.0" y="4.0" z="3.0"/>
 </size>
 <socket>
  <triple x="0.0" y="0.0" z="-1.5"/>
 </socket>
 <connections>
  <mount>
   <component identifier="c.myTurret">
    <size>
     <triple x="2.0" y="2.0" z="1.0"/>
    </size>
    <socket>
     <triple x="0.0" y="0.0" z="-0.5"/>
    </socket>
    <connections>
     <mount>
      <component identifier="c.myGun">
       <size>
        <triple x="5.0" y="0.5" z="0.5"/>
       </size>
       <socket>
        <triple x="-2.5" y="0.0" z="0.0"/>
       </socket>
      </component>
      <ball>
       <triple x="1.0" y="0.0" z="0.0"/>
      </ball>
     </mount>
    </connections>
   </component>
   <ball>
    <triple x="-1.0" y="0.0" z="1.5"/>
   </ball>
  </mount>
 </connections>
</component>
```

Gnuplot Output

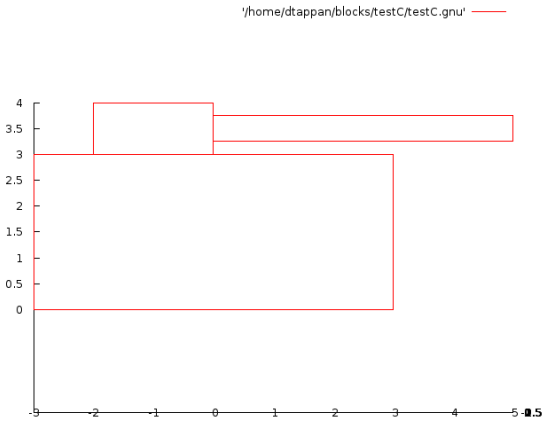
```
# component [c.myHull] {
# top
-3.0 -2.0 3.0
3.0 -2.0 3.0
3.0 2.0 3.0
-3.0 2.0 3.0
-3.0 -2.0 3.0

# bottom
-3.0 -2.0 0.0
3.0 -2.0 0.0
3.0 2.0 0.0
-3.0 2.0 0.0
-3.0 -2.0 0.0


# subcomponents {
# component [c.myTurret] {
# top
-2.0 -1.0 4.0
0.0 -1.0 4.0
0.0 1.0 4.0
-2.0 1.0 4.0
-2.0 -1.0 4.0

# bottom
-2.0 -1.0 3.0
0.0 -1.0 3.0
0.0 1.0 3.0
-2.0 1.0 3.0
-2.0 -1.0 3.0


# subcomponents {
# component [c.myGun] {
# top
0.0 -0.25 3.75
5.0 -0.25 3.75
5.0 0.25 3.75
0.0 0.25 3.75
0.0 -0.25 3.75

# bottom
0.0 -0.25 3.25
5.0 -0.25 3.25
5.0 0.25 3.25
0.0 0.25 3.25
0.0 -0.25 3.25


# } component [c.myGun]
# } subcomponents
# } component [c.myTurret]
# } subcomponents
# } component [c.myHull]
```

Gnuplot Rendering

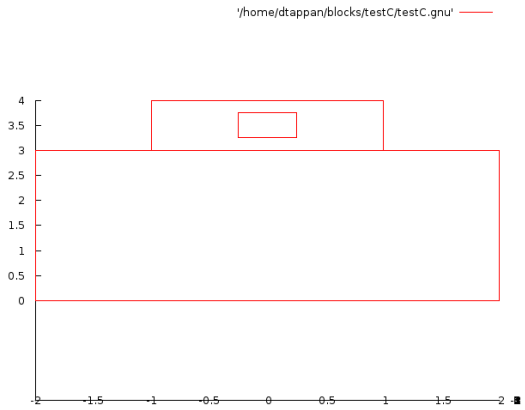
Top View



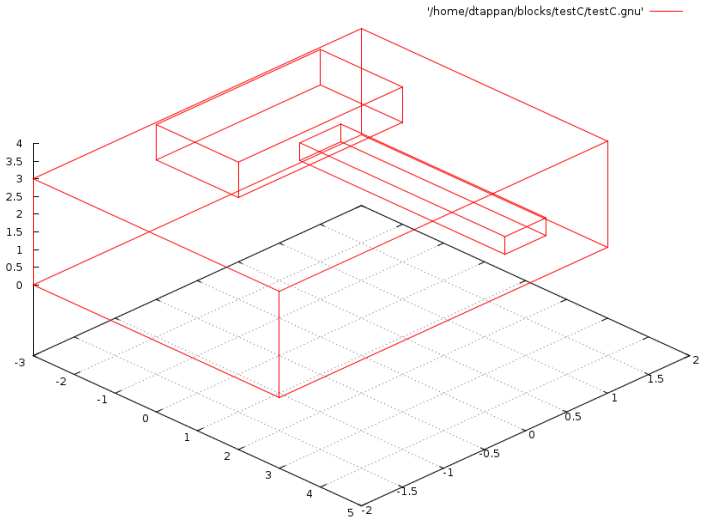
Side View



Front View



Orthogonal View



Test D

Description

This test generates a hull with a turret, gun, and sensor.

Input File

```
// testD.blk: hull with turret, gun, and sensor

define d = (
            id="d.myHull"
            volume=[6,4,3]
            socket=[0,0,-1.5]
            connectsTo
            (
             (
              id="d.myTurret"
              volume=[2,2,1]
              socket=[0,0,-0.5]
              connectsTo
              (
               (
                id="d.myGun"
                volume=[5,0.5,0.5]
                socket=[-2.5,0,0]
               ) at [1,0,0]
               (
                id="d.mySensor"
                volume=[0.5,0.5,0.5]
                socket=[0,0,-0.25]
               ) at [-0.5,0.5,0.5]
              )
             ) at [-1,0,1.5]
            )
           );

printXML(d);

exportToGnuplot(d,[0,0,0]);
```

XML Output

```
<component identifier="d.myHull">
 <size>
  <triple x="6.0" y="4.0" z="3.0"/>
 </size>
 <socket>
  <triple x="0.0" y="0.0" z="-1.5"/>
 </socket>
 <connections>
  <mount>
   <component identifier="d.myTurret">
    <size>
     <triple x="2.0" y="2.0" z="1.0"/>
    </size>
    <socket>
     <triple x="0.0" y="0.0" z="-0.5"/>
    </socket>
    <connections>
     <mount>
      <component identifier="d.myGun">
       <size>
        <triple x="5.0" y="0.5" z="0.5"/>
       </size>
       <socket>
        <triple x="-2.5" y="0.0" z="0.0"/>
       </socket>
      </component>
      <ball>
       <triple x="1.0" y="0.0" z="0.0"/>
      </ball>
     </mount>
     <mount>
      <component identifier="d.mySensor">
       <size>
```

```
        <triple x="0.5" y="0.5" z="0.5"/>
       </size>
       <socket>
        <triple x="0.0" y="0.0" z="-0.25"/>
       </socket>
      </component>
      <ball>
       <triple x="-0.5" y="0.5" z="0.5"/>
      </ball>
     </mount>
    </connections>
   </component>
   <ball>
   <triple x="-1.0" y="0.0" z="1.5"/>
   </ball>
  </mount>
 </connections>
</component>
```

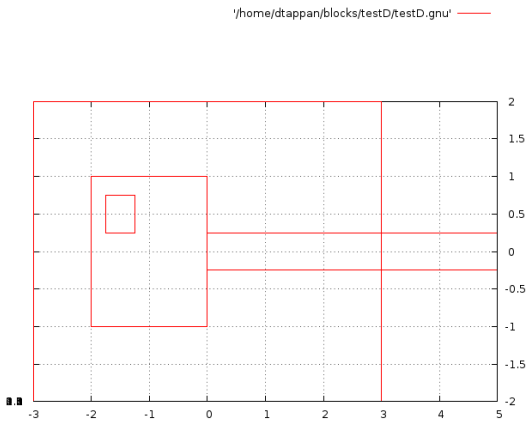
Gnuplot Output

```
# component [d.myHull] {
# top
-3.0 -2.0 3.0
3.0 -2.0 3.0
3.0 2.0 3.0
-3.0 2.0 3.0
-3.0 -2.0 3.0

# bottom
-3.0 -2.0 0.0
3.0 -2.0 0.0
3.0 2.0 0.0
-3.0 2.0 0.0
-3.0 -2.0 0.0


# subcomponents {
# component [d.myTurret] {
# top
-2.0 -1.0 4.0
0.0 -1.0 4.0
0.0 1.0 4.0
-2.0 1.0 4.0
-2.0 -1.0 4.0

# bottom
-2.0 -1.0 3.0
0.0 -1.0 3.0
0.0 1.0 3.0
-2.0 1.0 3.0
-2.0 -1.0 3.0


# subcomponents {
# component [d.myGun] {
# top
0.0 -0.25 3.75
5.0 -0.25 3.75
5.0 0.25 3.75
0.0 0.25 3.75
0.0 -0.25 3.75

# bottom
0.0 -0.25 3.25
5.0 -0.25 3.25
5.0 0.25 3.25
0.0 0.25 3.25
0.0 -0.25 3.25


# } component [d.myGun]
# component [d.mySensor] {
# top
-1.75 0.25 4.5
-1.25 0.25 4.5
```

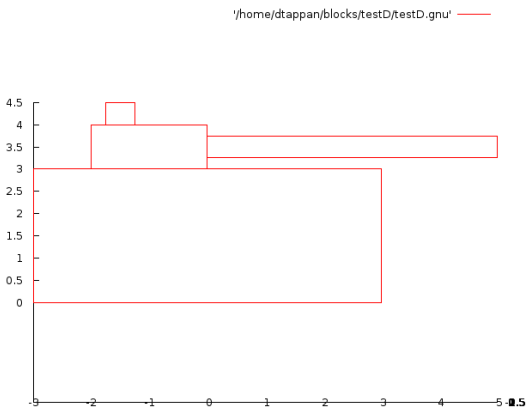
```
-1.25 0.75 4.5
-1.75 0.75 4.5
-1.75 0.25 4.5

# bottom
-1.75 0.25 4.0
-1.25 0.25 4.0
-1.25 0.75 4.0
-1.75 0.75 4.0
-1.75 0.25 4.0


# } component [d.mySensor]
# } subcomponents
# } component [d.myTurret]
# } subcomponents
# } component [d.myHull]
```

Gnuplot Rendering

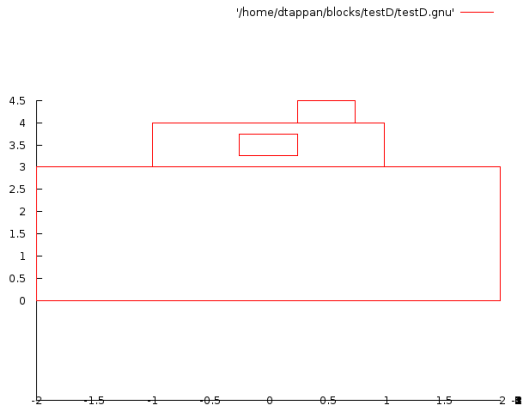
Top View



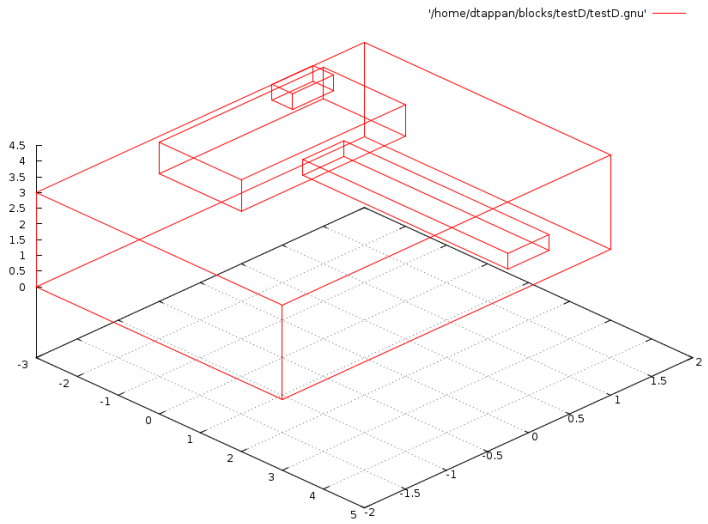
Side View



Front View



Orthogonal View



JavaCC Grammar

See `language.html` for complete details. See `language_demo.jj` for additional hints.

```
parse               ::= ( Command )* <EOF>
Connections         ::= <CONNECTS_TO> <LPAREN> ( ComponentDefinition <AT> Triple )+ <RPAREN>
Command             ::= ( Define | ExportToGnuplot | PrintXML ) <SEMICOLON>
ComponentDefinition ::= <LPAREN> Identifier Volume Socket ( Connections )? <RPAREN>
Define              ::= <DEFINE> Variable <ASSIGN> ComponentDefinition
ExportToGnuplot     ::= <EXPORT_TO_GNUPLOT> <LPAREN> Variable <COMMA> Triple <RPAREN>
Identifier          ::= <ID> <ASSIGN> <LITERAL_STRING>
PrintXML            ::= <PRINT_XML> <LPAREN> Variable <RPAREN>
Socket              ::= <SOCKET> <ASSIGN> Triple
Triple              ::= <LBRACKET> <LITERAL_NUMBER> <COMMA> <LITERAL_NUMBER> <COMMA> <LITERAL_NUMBER> <RBRACKET>
Variable            ::= <IDENTIFIER>
Volume              ::= <VOLUME> <ASSIGN> Triple
```

JavaCC Installation

Do the following in Eclipse:

Help → Install New Software.

Work with: `http://eclipse-javacc.sourceforge.net/`

Click the checkbox for JavaCC Eclipse Plug-in .

Next, accept terms, etc. Accept the unsigned content.

It will install the plug-in and restart Eclipse.

Create a project for `cs524builder`

Right-click `language_demo.jj` in the Project Explorer window.

Select Compile with JavaCC .

Run it from the green circle with the white arrow in it. You first have to set up the run configuration to point to `BuilderDemo`, along with the input filename (`demo.txt`) and output filename (your choice, not used in the demo) in the *Arguments->Program arguments*. The black down arrow right of the green circle opens this dialog through *Run Configurations*.

Gnuplot Commands

The Gnuplot installation will vary depending on your platform. See `http://www.gnuplot.info/`

To set the scale, use the *Apply autoscale* button.

To render: `splot 'testA.gnu' with lines`

To set the perspective:

top:	`set view 0,0`
side:	`set view 90,0`
front:	`set view 90,90`
orthogonal:	`set view 45,45`