/Users/jchavez/dev/java.net/jgsl.dev.java.net/jgsl/dev/parser/JGSL_Parser.jj

```
1       /*
2   JGSL Grammar for JavaCC
3   */
4
5
6   options {
7    JAVA_UNICODE_ESCAPE = true;
8    STATIC = false;
9    IGNORE_CASE = true;
10  }
11
12
13
14  PARSER_BEGIN(JGSL_Parser)
15
16  package jgsl.parser;
17
18
19  import jgsl.io.ScriptMessage;
20  import jgsl.io.ScriptWarning;
21  import jgsl.io.ScriptError;
22  import jgsl.io.ScriptParserException;
23  import jgsl.io.ScriptParserUtil;
24
25  import jgsl.model.JGSLDouble;
26  import jgsl.model.JGSLInteger;
27  import jgsl.model.JGSLString;
28  import jgsl.model.JGSLColor;
29  import jgsl.model.JGSLScript;
30  import jgsl.model.Assignment;
31  import jgsl.model.Command;
32  import jgsl.model.Declaration;
33  import jgsl.model.Documentation;
34  import jgsl.model.Logical;
35  import jgsl.model.Statement;
36
37  import java.awt.Color;
38  import java.util.ArrayList;
39
40  /**
41   *
42   * @author zenarchitect
43   * @version $Id: JGSL_Parser.jj,v 1.5 2005/05/01 01:48:23 zenarchitect Exp $
44   */
45
46  public class JGSL_Parser {
47
```

```
48   private JGSLScript script = new JGSLScript();
49
50   public JGSLScript getScript() {
51      return script;
52   }
53
54   public static void main(String args[]) throws ParseException {
55        JGSL_Parser parser;
56        String filename = null;
57        long initTime = 0;
58        long parseTime = 0;
59        long startTime = 0;
60        long stopTime = 0;
61        if (args.length == 0)
62        {
63           System.out.println("jgsl parser:  Reading from standard input . . .");
64           parser = new JGSL_Parser(System.in);
65        } else if (args.length == 1)
66        {
67           filename = args[0];
68           System.out.println("jgsl parser:  Reading from file " + filename + " . . .");
69           try
70           {
71              startTime = System.currentTimeMillis();
72              parser = new JGSL_Parser(new java.io.FileInputStream(filename));
73              stopTime = System.currentTimeMillis();
74              initTime = stopTime - startTime;
75           } catch (java.io.FileNotFoundException e)
76           {
77              System.out.println("jgsl parser:  File " + filename + " not found.");
78              return;
79           }
80        } else
81        {
82           System.out.println("jgsl parser:  Usage is one of:");
83           System.out.println("        java jsgl.JSGL < <stdin>");
84           System.out.println("OR");
85           System.out.println("        java jsgl.JSGL inputfile");
86           return;
87        }
88        try
89        {
90           startTime = System.currentTimeMillis();
91           parser.parseScript();
92           stopTime = System.currentTimeMillis();
93           parseTime = stopTime - startTime;
94           System.out.println("jgsl parser: ");
95           System.out.println("   JGSL file parsed " + filename + " successfully in " + (initTime + parseTime) + "
ms.");
96           System.out.println("      initialization time = " + initTime + " ms.");
97           System.out.println("      parse time = " + parseTime + " ms.");
```

```
98        } catch (ParseException e)
99        {
100           System.out.println(e.getMessage());
101           System.out.println("jgsl parser:  Encountered errors during parse.");
102        }
103    }
104
105  }
106
107  PARSER_END(JGSL_Parser)
108
109  /* WHITE SPACE */
110
111   SPECIAL_TOKEN :
112   {
113     " "
114   | "\t"
115   | "\n"
116   | "\r"
117   | "\f"
118   }
119
120   /* COMMENTS */
121
122   MORE :
123   {
124     "//" : IN_SINGLE_LINE_COMMENT
125   |
126     "#" : IN_SINGLE_LINE_COMMENT
127   |
128     <"/**" ~["/"]> { input_stream.backup(1); } : IN_FORMAL_COMMENT
129   |
130     "/*" : IN_MULTI_LINE_COMMENT
131   }
132
133
134   <IN_SINGLE_LINE_COMMENT>
135   SPECIAL_TOKEN :
136   {
137     <SINGLE_LINE_COMMENT: "\n" | "\r" | "\r\n" > : DEFAULT
138   }
139
140   <IN_FORMAL_COMMENT>
141   SPECIAL_TOKEN :
142   {
143     <FORMAL_COMMENT: "*/" > : DEFAULT
144   }
145
146   <IN_MULTI_LINE_COMMENT>
147   SPECIAL_TOKEN :
148   {
```

```
149   <MULTI_LINE_COMMENT: "*/" > : DEFAULT
150  }
151
152  <IN_SINGLE_LINE_COMMENT,IN_FORMAL_COMMENT,IN_MULTI_LINE_COMMENT>
153  MORE :
154  {
155    < ~[] >
156  }
157
158  /* RESERVED WORDS AND LITERALS */
159
160
161  // logic  TODO
162  TOKEN :
163  {
164    < AND: "and" >
165  | < OR: "or" >
166  | < NOT: "not" >
167  | < IF: "if" >
168  | < THEN: "then" >
169  | < ELSE: "else" >
170  | < ELSEIF: "elseif" >
171  | < TRUE: "true" >
172  | < FALSE: "false" >
173  }
174
175  // commands TODO
176  TOKEN :
177  {
178    < CLEAR: "clear" >
179  | < CANVAS: "canvas" >
180  | < DRAW: "draw" >
181  | < TEXT: "text" >
182  | < RECTANGLE: "rectangle" >
183  | < SQUARE: "square" >
184  | < CIRCLE: "circle" >
185  | < ELIPSE: "elipse" >
186  | < ARC: "arc" >
187  | < POLYGON: "polygon" >
188  | < LINE: "line" >
189  | < WAIT: "wait" >
190  | < LOG: "log">
191  | < DEBUG: "debug" >
192  | < ERROR: "error" >
193  | < WARNING: "warning" >
194  | < WRITE: "write" >
195  | < READ: "read" >
196  | < JGSL: "jgsl" >
197  | < VERSION: "version" >
198  }
199
```

```
200 // flow control TODO
201  TOKEN :
202  {
203    < BEGIN: "begin" >
204  | < END: "end" >
205  | < REPEAT: "repeat" >
206  | < LOOP: "loop" >
207  }
208
209 // colors
210  TOKEN :
211  {
212    < BLACK: "black" >
213  | < BLUE: "blue" >
214  | < DARK_GRAY: "dark_gray" >
215  | < GRAY: "gray" >
216  | < GREEN: "green" >
217  | < LIGHT_GRAY: "light_gray" >
218  | < MAGENTA: "magenta" >
219  | < ORANGE: "orange" >
220  | < PINK: "pink" >
221  | < RED: "red" >
222  | < WHITE: "white" >
223  | < YELLOW: "yellow" >
224  }
225
226 // painting objects
227 TOKEN :
228 {
229    < GRADIENT: "gradient" >
230  | < FILL: "fill" >
231  | < BORDER: "border" >
232  | < FOREGROUND: "foreground" >
233  | < BACKGROUND: "background" >
234  | < COLOR: "color" >
235 }
236
237 // language
238  TOKEN :
239  {
240    < DECLARE: "declare" >
241  | < DOC: "doc" >   // TODO
242  }
243
244
245
246
247  /* LITERALS */
248
249  TOKEN :
250  {
```

```
  < INTEGER_LITERAL:
      <DECIMAL_LITERAL>
    | <HEX_LITERAL>
    | <OCTAL_LITERAL>
  >
|
  < #DECIMAL_LITERAL: ["1"-"9"] (["0"-"9"])* >
|
  < #HEX_LITERAL: "0" ["x","X"] (["0"-"9","a"-"f","A"-"F"])+ >
|
  < #OCTAL_LITERAL: "0" (["0"-"7"])* >
|
  < FLOATING_POINT_LITERAL:
      (["0"-"9"])+ "." (["0"-"9"])*
  >
|
  < STRING_LITERAL:
    "\""
    (   (~["\"","\\","\n","\r"])
      | ("\\"
         ( ["n","t","b","r","f","\\","'","\""]
         | ["0"-"7"] ( ["0"-"7"] )?
         | ["0"-"3"] ["0"-"7"] ["0"-"7"]
         )
        )
    )*
    "\""
  >
}

/* IDENTIFIERS */

TOKEN :
{
  < IDENTIFIER: <LETTER> (<LETTER>|<DIGIT>)* >
|
  < #LETTER:
    [
     "\u0024",
     "\u0041"-"\u005a",
     "\u005f",
     "\u0061"-"\u007a",
     "\u00c0"-"\u00d6",
     "\u00d8"-"\u00f6",
     "\u00f8"-"\u00ff",
     "\u0100"-"\u1fff",
     "\u3040"-"\u318f",
     "\u3300"-"\u337f",
     "\u3400"-"\u3d2d",
     "\u4e00"-"\u9fff",
     "\uf900"-"\ufaff"
```

```
302       ]
303     >
304   |
305     < #DIGIT:
306       [
307         "\u0030"-"\u0039",
308         "\u0660"-"\u0669",
309         "\u06f0"-"\u06f9",
310         "\u0966"-"\u096f",
311         "\u09e6"-"\u09ef",
312         "\u0a66"-"\u0a6f",
313         "\u0ae6"-"\u0aef",
314         "\u0b66"-"\u0b6f",
315         "\u0be7"-"\u0bef",
316         "\u0c66"-"\u0c6f",
317         "\u0ce6"-"\u0cef",
318         "\u0d66"-"\u0d6f",
319         "\u0e50"-"\u0e59",
320         "\u0ed0"-"\u0ed9",
321         "\u1040"-"\u1049"
322       ]
323     >
324   }
325
326
327   /* SEPARATORS */
328
329   TOKEN :
330   {
331     < LPAREN: "(" >
332   | < RPAREN: ")" >
333   | < SEMICOLON: ";" >
334   | < COLON: ":" >
335   | < COMMA: "," >
336   }
337
338   /* OPERATORS */
339
340   TOKEN :
341   {
342     < ASSIGN: "=" >
343   | < GT: ">" >
344   | < LT: "<" >
345   | < BANG: "!" >
346   | < EQ: "==" >
347   | < LE: "<=" >
348   | < GE: ">=" >
349   | < NE: "!=" >
350   | < SC_OR: "||" >
351   | < SC_AND: "&&" >
352   | < PLUS: "+" >
```

```
353 | < MINUS: "-" >
354 | < STAR: "*" >
355 | < SLASH: "/" >
356 | < MOD: "%" >
357 }
358
359
360 /*
361 <comments | documentation>*
362
363 <BEGIN>
364
365 <attributes | commands | comments | documentation>
366
367 <END>
368
369 */
370
371 /*****************************************
372  * THE JGSL GRAMMAR STARTS HERE          *
373  *****************************************/
374
375 /*
376  * Program structuring syntax follows.
377  */
378
379 void parseScript() :
380 {}
381 {
382    Script()
383 }
384
385 void Script() :
386 {}
387 {
388   (Documentation())*
389   [ ScriptBody() ]
390   <EOF>
391 }
392
393 void Documentation() :
394 {
395    Token doc;
396 }
397 {
398   <DOC>
399   doc = <STRING_LITERAL>
400   {
401     script.addDocumentation(doc.image);
402   }
403 }
```

```
404
405 void ScriptBody() :
406 {}
407 {
408   <BEGIN>
409   ( Command() | Declaration() | Assignment() | Documentation() )*
410   <END>
411 }
412
413 void Command() :
414 {}
415 {
416     try {
417         (
418             Canvas()
419             | Clear()
420             | Wait()
421             | Draw()
422             | DrawShape()
423             | DrawText()
424             | Log()
425         ) <SEMICOLON>
426     }
427     catch(ParseException e) {
428         error_skipto(SEMICOLON);
429     }
430 }
431
432
433 JAVACODE
434 void error_skipto(int kind) {
435     ParseException e = generateParseException();  // generate the exception object.
436
437     ScriptError se = new ScriptError(e.getMessage());
438     script.addError(se);
439
440     Token t;
441     do {
442     t = getNextToken();
443     } while (t.kind != kind);
444     // The above loop consumes tokens all the way up to a token of
445     // "kind".  We use a do-while loop rather than a while because the
446     // current token is the one immediately before the erroneous token
447     // (in our case the token immediately before what should have been
448     // "if"/"while".
449 }
450
451 void Clear() :
452 {
453     Token name;
454 }
```

```
{
   name = <CLEAR>
   {
      Command c = new Command(name.image);
      script.add(c);
   }
}


Color GetStandardColor() :
{
   Token value;
}
{
   try {
   ( value = <BLACK>
      | value = <BLUE>
      | value = <DARK_GRAY>
      | value = <GRAY>
      | value = <GREEN>
      | value = <LIGHT_GRAY>
      | value = <MAGENTA>
      | value = <ORANGE>
      | value = <PINK>
      | value = <RED>
      | value = <WHITE>
      | value = <YELLOW>
   )
   } catch (ParseException e) {
      ScriptError se = new ScriptError(e.getMessage());
      script.addError(se);
      return null;
   }
   {
//       Color color = Color.RED;
      String colorName = value.image.toLowerCase();
      if(colorName.equals("black")) {
         return Color.BLACK;
      }
      else if(colorName.equals("blue")) {
         return Color.BLUE;
      }
      else if(colorName.equals("dark_gray")) {
         return Color.DARK_GRAY;
      }
      else if(colorName.equals("gray")) {
         return Color.GRAY;
      }
      else if(colorName.equals("green")) {
         return Color.GREEN;
      }
```

```
506         else if(colorName.equals("light_gray")) {
507            return Color.LIGHT_GRAY;
508         }
509         else if(colorName.equals("magenta")) {
510            return Color.MAGENTA;
511         }
512         else if(colorName.equals("orange")) {
513            return Color.ORANGE;
514         }
515         else if(colorName.equals("pink")) {
516            return Color.PINK;
517         }
518         else if(colorName.equals("red")) {
519            return Color.RED;
520         }
521         else if(colorName.equals("white")) {
522            return Color.WHITE;
523         }
524         else if(colorName.equals("yellow")) {
525            return Color.YELLOW;
526         }
527      }
528   }
529
530   Color GetRGB() :
531   {
532      Token r;
533      Token g;
534      Token b;
535   }
536   {
537      ( r = <INTEGER_LITERAL> <COMMA> g = <INTEGER_LITERAL> <COMMA> b = <INTEGER_LITERAL>
538         | r = <STRING_LITERAL> <COMMA> g = <STRING_LITERAL> <COMMA> b = <STRING_LITERAL>
539      )
540      {
541   //       Color rgb = new Color(0,0,0);
542         int rInt = -1;
543         int gInt = -1;
544         int bInt = -1;
545
546         try {
547            rInt = ScriptParserUtil.parseInt(r.image);
548         }
549         catch(ScriptParserException e) {
550            ScriptError se = new ScriptError(e.getMessage(), r.beginLine, r.beginColumn);
551            script.addError(se);
552         }
553         try {
554            rInt = ScriptParserUtil.parseInt(g.image);
555         }
556         catch(ScriptParserException e) {
```

```
557           ScriptError se = new ScriptError(e.getMessage(), g.beginLine, g.beginColumn);
558           script.addError(se);
559        }
560        try {
561           rInt = ScriptParserUtil.parseInt(b.image);
562        }
563        catch(ScriptParserException e) {
564           ScriptError se = new ScriptError(e.getMessage(), b.beginLine, b.beginColumn);
565           script.addError(se);
566        }
567
568        Color c = null;
569
570        if(rInt != -1 && gInt != -1 && bInt != -1) {
571           c = new Color(rInt, gInt, bInt);
572        }
573
574        return c;
575
576     }
577  }
578
579
580
581  Color GetColor() :
582  {
583     Color c;
584  }
585  {
586     ( c = GetStandardColor() | c = GetRGB())
587     {
588        return c;
589     }
590  }
591
592  void Canvas() :
593  {
594     Token name = null;
595     Token width = null;
596     Token height = null;
597     Color bgcolor = null;
598     Color fgcolor = null;
599     Token title = null;
600     ArrayList attributes = new ArrayList();
601  }
602  {
603        name = <CANVAS> (<COLON> attributes = CanvasAttributes())?
604           <LPAREN> (width = <INTEGER_LITERAL> | width = <STRING_LITERAL>) <COMMA>
605           (height = <INTEGER_LITERAL> | height = <STRING_LITERAL>) <COMMA>
606           bgcolor = GetColor() <COMMA>
607           fgcolor = GetColor()
```

```
608         ( <COMMA> title = <STRING_LITERAL>)? <RPAREN>
609     {
610         ArrayList parameters = new ArrayList(4);
611 
612         if(bgcolor != null) {
613            parameters.add(new JGSLColor("background", bgcolor));
614         }
615         else {
616            parameters.add(new JGSLColor("background", Color.WHITE));
617            ScriptWarning se = new ScriptWarning("Setting canvas background to WHITE.");
618            script.addWarning(se);
619         }
620 
621         if(fgcolor != null) {
622            parameters.add(new JGSLColor("foreground", fgcolor));
623         }
624         else {
625            parameters.add(new JGSLColor("foreground", Color.BLACK));
626            ScriptWarning se = new ScriptWarning("Setting canvas foreground to BLACK.");
627            script.addWarning(se);
628         }
629 
630         addIntParam(parameters, "width", width);
631         addIntParam(parameters, "height", height);
632 
633         if(title != null) {
634            parameters.add(new JGSLString("title", title.image));
635         }
636         Command c = new Command(name.image, attributes, parameters);
637         script.add(c);
638     }
639 }
640 
641 
642 ArrayList CanvasAttributes() :
643 {
644    Token attrib;
645 }
646 {
647    attrib = <RED>
648    {
649       ArrayList attributes = new ArrayList(1);
650       return attributes;
651    }
652 }
653 
654 
655 void Wait() :
656 {
657    Token name;
658    Token durationSeconds;
```

```
659 }
660 {
661     name = <WAIT> <LPAREN> (durationSeconds = <INTEGER_LITERAL> | durationSeconds =
<STRING_LITERAL>) <RPAREN>
662     {
663         ArrayList parameters = new ArrayList(1);
664         addIntParam(parameters, "duration", durationSeconds);
665         Command c = new Command(name.image, parameters);
666         script.add(c);
667     }
668 }
669
670 ArrayList DrawAttributes() :
671 {
672     Token attrib;
673 }
674 {
675     attrib = <RED>
676     {
677         ArrayList attributes = new ArrayList(1);
678         return attributes;
679     }
680 }
681
682 void Draw() :
683 {
684     Token name;
685     Token x;
686     Token y;
687     ArrayList attributes = new ArrayList();
688 }
689 {
690     ( name = <DRAW> (<COLON> attributes = DrawAttributes())?
691         <LPAREN> (x = <INTEGER_LITERAL> | x = <STRING_LITERAL>) <COMMA>
692             (y = <INTEGER_LITERAL> | y = <STRING_LITERAL>)
693             <RPAREN>)
694      {
695             ArrayList parameters = new ArrayList(2);
696             addIntParam(parameters, "x", x);
697             addIntParam(parameters, "y", y);
698             addIntParam(parameters, "x", x);
699             addIntParam(parameters, "y", y);
700             Command c = new Command(name.image, attributes, parameters);
701             script.add(c);
702      }
703 }
704
705 //<COLON> (Attributes())? (<LPAREN> Parameters() <RPAREN>)?
706
707 void Log() :
708 {}
```

```
709 {
710    ( Message() | Warning() | Debug() | Error() )
711 }
712
713 void Message() :
714 {
715    Token name;
716    Token message;
717 }
718 {
719    name = <LOG> <LPAREN> (message = <STRING_LITERAL>) <RPAREN>
720    {
721       ArrayList parameters = new ArrayList(1);
722       parameters.add(new JGSLString("message", message.image));
723       Command c = new Command(name.image, parameters);
724       script.add(c);
725    }
726 }
727
728 void Warning() :
729 {
730    Token name;
731    Token message;
732 }
733 {
734    name = <WARNING> <LPAREN> (message = <STRING_LITERAL>) <RPAREN>
735    {
736       ArrayList parameters = new ArrayList(1);
737       parameters.add(new JGSLString("message", message.image));
738       Command c = new Command(name.image, parameters);
739       script.add(c);
740    }
741 }
742
743 void Error() :
744 {
745    Token name;
746    Token message;
747 }
748 {
749    name = <ERROR> <LPAREN> (message = <STRING_LITERAL>) <RPAREN>
750    {
751       ArrayList parameters = new ArrayList(1);
752       parameters.add(new JGSLString("message", message.image));
753       Command c = new Command(name.image, parameters);
754       script.add(c);
755    }
756 }
757
758 void Debug() :
759 {
```

```
760     Token name;
761     Token message;
762  }
763  {
764     name = <DEBUG> <LPAREN> (message = <STRING_LITERAL>) <RPAREN>
765     {
766        ArrayList parameters = new ArrayList(1);
767        parameters.add(new JGSLString("message", message.image));
768        Command c = new Command(name.image, parameters);
769        script.add(c);
770     }
771  }
772
773  void DrawShape() :
774  {}
775  {
776     (
777        DrawLine()
778        | DrawRectangle()
779        | DrawSquare()
780        | DrawCircle()
781        | DrawElipse()
782        | DrawArc()
783        | DrawPolygon()
784        )
785  }
786
787  void DrawLine() :
788  {
789     Token name;
790     Token x1;
791     Token y1;
792     Token x2;
793     Token y2;
794     ArrayList attributes = new ArrayList();
795  }
796  {
797     ( name = <LINE> (<COLON> attributes = LineAttributes())?
798        <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
799           (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
800           (x2 = <INTEGER_LITERAL> | x2 = <STRING_LITERAL>) <COMMA>
801           (y2 = <INTEGER_LITERAL> | y2 = <STRING_LITERAL>) <RPAREN>)
802      {
803           ArrayList parameters = new ArrayList(4);
804           addIntParam(parameters, "x1", x1);
805           addIntParam(parameters, "y1", y1);
806           addIntParam(parameters, "x2", x2);
807           addIntParam(parameters, "y2", y2);
808           Command c = new Command(name.image, attributes, parameters);
809           script.add(c);
810      }
```

```
811 }
812
813 ArrayList LineAttributes() :
814 {
815    Color c;
816 }
817 {
818    c = GetColor()
819    {
820       ArrayList attributes = new ArrayList(1);
821       attributes.add(new JGSLColor("c", c));
822       return attributes;
823    }
824 }
825
826
827
828 void DrawRectangle() :
829 {
830    Token name;
831    Token x1;
832    Token y1;
833    Token width;
834    Token height;
835    ArrayList attributes = new ArrayList();
836 }
837 {
838    ( name = <RECTANGLE> (<COLON> attributes = RectangleAttributes())?
839       <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
840          (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
841          (width = <INTEGER_LITERAL> | width = <STRING_LITERAL>) <COMMA>
842          (height = <INTEGER_LITERAL> | height = <STRING_LITERAL>) <RPAREN>)
843     {
844          ArrayList parameters = new ArrayList(4);
845          addIntParam(parameters, "x1", x1);
846          addIntParam(parameters, "y1", y1);
847          addIntParam(parameters, "width", width);
848          addIntParam(parameters, "height", height);
849          Command c = new Command(name.image, attributes, parameters);
850          script.add(c);
851     }
852 }
853
854 ArrayList RectangleAttributes() :
855 {
856    Color c;
857 }
858 {
859    c = GetColor()
860    {
861       ArrayList attributes = new ArrayList(1);
```

```
862         attributes.add(new JGSLColor("c", c));
863         return attributes;
864     }
865 }
866
867
868 void DrawSquare() :
869 {
870     Token name;
871     Token x1;
872     Token y1;
873     Token width;
874     ArrayList attributes = new ArrayList();
875 }
876 {
877     ( name = <SQUARE> (<COLON> attributes = SquareAttributes())?
878         <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
879             (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
880             (width = <INTEGER_LITERAL> | width = <STRING_LITERAL>)  <RPAREN>)
881      {
882             ArrayList parameters = new ArrayList(4);
883             addIntParam(parameters, "x1", x1);
884             addIntParam(parameters, "y1", y1);
885             addIntParam(parameters, "width", width);
886             Command c = new Command(name.image, attributes, parameters);
887             script.add(c);
888      }
889 }
890
891 ArrayList SquareAttributes() :
892 {
893     Color c;
894 }
895 {
896     c = GetColor()
897     {
898         ArrayList attributes = new ArrayList(1);
899         attributes.add(new JGSLColor("c", c));
900         return attributes;
901     }
902 }
903
904 void DrawCircle() :
905 {
906     Token name;
907     Token x1;
908     Token y1;
909     Token radius;
910     ArrayList attributes = new ArrayList();
911 }
912 {
```

```
913     ( name = <CIRCLE> (<COLON> attributes = CircleAttributes())?
914       <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
915         (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
916         (radius = <FLOATING_POINT_LITERAL> | radius = <STRING_LITERAL>)  <RPAREN>)
917     {
918         ArrayList parameters = new ArrayList(4);
919         addIntParam(parameters, "x1", x1);
920         addIntParam(parameters, "y1", y1);
921         addDecimalParam(parameters, "radius", radius);
922         Command c = new Command(name.image, attributes, parameters);
923         script.add(c);
924     }
925 }
926
927 ArrayList CircleAttributes() :
928 {
929     Color c;
930 }
931 {
932     c = GetColor()
933     {
934       ArrayList attributes = new ArrayList(1);
935       attributes.add(new JGSLColor("c", c));
936       return attributes;
937     }
938 }
939
940 void DrawElipse() :
941 {
942     Token name;
943     Token x1;
944     Token y1;
945     Token width;
946     Token height;
947     ArrayList attributes = new ArrayList();
948 }
949 {
950     ( name = <ELIPSE> (<COLON> attributes = ElipseAttributes())?
951       <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
952         (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
953         (width = <INTEGER_LITERAL> | width = <STRING_LITERAL>) <COMMA>
954         (height = <INTEGER_LITERAL> | height = <STRING_LITERAL>) <RPAREN>)
955     {
956         ArrayList parameters = new ArrayList(4);
957         addIntParam(parameters, "x1", x1);
958         addIntParam(parameters, "y1", y1);
959         addIntParam(parameters, "width", width);
960         addIntParam(parameters, "height", height);
961         Command c = new Command(name.image, attributes, parameters);
962         script.add(c);
963     }
```

```
964  }
965
966  ArrayList ElipseAttributes() :
967  {
968     Color c;
969  }
970  {
971     c = GetColor()
972     {
973        ArrayList attributes = new ArrayList(1);
974        attributes.add(new JGSLColor("c", c));
975        return attributes;
976     }
977  }
978
979  void DrawArc() :
980  {
981     Token name;
982     Token x1;
983     Token y1;
984     Token width;
985     Token height;
986     Token startAngle;
987     Token arcAngle;
988     ArrayList attributes = new ArrayList();
989  }
990  {
991     ( name = <ARC> (<COLON> attributes = ArcAttributes())?
992        <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
993           (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
994           (width = <INTEGER_LITERAL> | width = <STRING_LITERAL>) <COMMA>
995           (height = <INTEGER_LITERAL> | height = <STRING_LITERAL>) <COMMA>
996           (startAngle = <INTEGER_LITERAL> | startAngle = <STRING_LITERAL>) <COMMA>
997           (arcAngle = <INTEGER_LITERAL> | arcAngle = <STRING_LITERAL>)
998           <RPAREN>)
999      {
1000          ArrayList parameters = new ArrayList(4);
1001          addIntParam(parameters, "x1", x1);
1002          addIntParam(parameters, "y1", y1);
1003          addIntParam(parameters, "width", width);
1004          addIntParam(parameters, "height", height);
1005          addIntParam(parameters, "startAngle", startAngle);
1006          addIntParam(parameters, "arcAngle", arcAngle);
1007          Command c = new Command(name.image, attributes, parameters);
1008          script.add(c);
1009     }
1010 }
1011
1012 ArrayList ArcAttributes() :
1013 {
1014    Color c;
```

```
1015 }
1016 {
1017    c = GetColor()
1018    {
1019       ArrayList attributes = new ArrayList(1);
1020       attributes.add(new JGSLColor("c", c));
1021       return attributes;
1022    }
1023 }
1024
1025 void DrawPolygon() :
1026 {
1027    Token name;
1028    Token x1;
1029    Token y1;
1030    Token width;
1031    Token height;
1032    Token startAngle;
1033    Token arcAngle;
1034    ArrayList attributes = new ArrayList();
1035    ArrayList parameters = new ArrayList();
1036 }
1037 {
1038    ( name = <POLYGON> (<COLON> attributes = PolygonAttributes())?
1039       <LPAREN> (PolygonParameters(parameters))
1040          <RPAREN>)
1041     {
1042          Command c = new Command(name.image, attributes, parameters);
1043          script.add(c);
1044     }
1045 }
1046
1047 ArrayList PolygonAttributes() :
1048 {
1049    Color c;
1050 }
1051 {
1052    c = GetColor()
1053    {
1054       ArrayList attributes = new ArrayList(1);
1055       attributes.add(new JGSLColor("c", c));
1056       return attributes;
1057    }
1058 }
1059
1060 void PolygonParameters(ArrayList parameters) :
1061 {
1062    Token x1 = null;
1063    Token y1 = null;
1064 }
1065 {
```

```
1066    ((x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
1067          (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) (<COMMA>)? )*
1068    {
1069       addIntParam(parameters, "x", x1);
1070       addIntParam(parameters, "y", y1);
1071    }
1072 }
1073
1074 void DrawText() :
1075 {
1076    Token name;
1077    Token x1;
1078    Token y1;
1079    Token text;
1080    ArrayList attributes = new ArrayList();
1081 }
1082 {
1083    ( name = <TEXT> (<COLON> attributes = TextAttributes())?
1084       <LPAREN> (x1 = <INTEGER_LITERAL> | x1 = <STRING_LITERAL>) <COMMA>
1085          (y1 = <INTEGER_LITERAL> | y1 = <STRING_LITERAL>) <COMMA>
1086          (text = <STRING_LITERAL>)
1087          <RPAREN>)
1088     {
1089          ArrayList parameters = new ArrayList(3);
1090          addIntParam(parameters, "x1", x1);
1091          addIntParam(parameters, "y1", y1);
1092          parameters.add(new JGSLString("text", text.image));
1093
1094          Command c = new Command(name.image, attributes, parameters);
1095          script.add(c);
1096     }
1097 }
1098
1099
1100 JAVACODE
1101 void addIntParam(ArrayList parameters, String paramName, Token t) {
1102    try {
1103       if(t == null) {
1104          String msg = "Unable to convert value to a number.";
1105          ScriptError se = new ScriptError(msg, t.beginLine, t.beginColumn);
1106          script.addError(se);
1107       }
1108       parameters.add(new JGSLInteger(paramName, t.image));
1109    }
1110    catch(NumberFormatException e) {
1111       String msg = "Unable to convert value " + t.image + " to a number.";
1112       ScriptError se = new ScriptError(msg, t.beginLine, t.beginColumn);
1113       script.addError(se);
1114    }
1115 }
1116
```

```
1117 JAVACODE
1118 void addDecimalParam(ArrayList parameters, String paramName, Token t) {
1119     try {
1120         parameters.add(new JGSLDouble(paramName, t.image));
1121     }
1122     catch(NumberFormatException e) {
1123         String msg = "Unable to convert value " + t.image + " to a number.";
1124         ScriptError se = new ScriptError(msg, t.beginLine, t.beginColumn);
1125         script.addError(se);
1126     }
1127 }
1128
1129 ArrayList TextAttributes() :
1130 {
1131     Color c;
1132 }
1133 {
1134     c = GetColor()
1135     {
1136         ArrayList attributes = new ArrayList(1);
1137         attributes.add(new JGSLColor("c", c));
1138         return attributes;
1139     }
1140 }
1141
1142
1143 void Declaration() :
1144 {}
1145 {
1146     ( <DECLARE> (DeclareColor() |
1147         (<IDENTIFIER> (<ASSIGN> (<IDENTIFIER> | <INTEGER_LITERAL> | <STRING_LITERAL> |
<FLOATING_POINT_LITERAL>) )? ) ) ) <SEMICOLON>
1148 }
1149
1150 void DeclareCanvas() :
1151 {
1152     Token type;
1153     Token id;
1154     Token tokenValue = null;
1155 }
1156 {
1157     type = <CANVAS> id = <IDENTIFIER> (<ASSIGN> tokenValue = <CANVAS>)?
1158     {
1159         String value = null;
1160         if(tokenValue != null) {
1161             value = tokenValue.image;
1162         }
1163         Declaration d = new Declaration(type.image, id.image, value);
1164         script.add(d);
1165     }
1166 }
```

```
1167
1168 void DeclareColor() :
1169 {
1170    Token type;
1171    Token id;
1172 }
1173 {
1174    type = <COLOR> id = <IDENTIFIER> (<ASSIGN> (DeclareStandardColor(type, id) | (DeclareRGB(type, id)) ) )?
1175 }
1176
1177 void DeclareStandardColor(Token type, Token id) :
1178 {
1179    Token value;
1180 }
1181 {
1182    ( value = <BLACK>
1183       | value = <BLUE>
1184       | value = <DARK_GRAY>
1185       | value = <GRAY>
1186       | value = <GREEN>
1187       | value = <LIGHT_GRAY>
1188       | value = <MAGENTA>
1189       | value = <ORANGE>
1190       | value = <PINK>
1191       | value = <RED>
1192       | value = <WHITE>
1193       | value = <YELLOW>
1194    )
1195    {
1196       Declaration d = new Declaration(type.image, id.image, value.image);
1197       script.add(d);
1198    }
1199 }
1200
1201 void DeclareRGB(Token type, Token id) :
1202 {
1203    Token r;
1204    Token g;
1205    Token b;
1206 }
1207 {
1208    ( r = <INTEGER_LITERAL> <COMMA> g = <INTEGER_LITERAL> <COMMA> b = <INTEGER_LITERAL>
1209       | r = <STRING_LITERAL> <COMMA> g = <STRING_LITERAL> <COMMA> b = <STRING_LITERAL>
1210    )
1211    {
1212       String rgb = r.image + "," + g.image + "," + b.image;
1213       Declaration d = new Declaration(type.image, id.image, rgb);
1214       script.add(d);
1215    }
1216 }
1217
```

```
1218 void Assignment() :
1219 {
1220    Token lhs;
1221    Token rhs;
1222 }
1223 {
1224    lhs = <IDENTIFIER> <ASSIGN> (rhs = <IDENTIFIER> | rhs = <INTEGER_LITERAL> | rhs =
<STRING_LITERAL>)
1225    {
1226       Assignment a = new Assignment(lhs.image, rhs.image);
1227       script.add(a);
1228    }
1229     <SEMICOLON>
1230 }
1231
1232
```