The Plotter Language INTerpretor

Sebastian Glita[1]

22nd July 2002

black
[0] gseba@users.sourforge.net

Contents

17Hfootnote.1

Chapter 1

Final

There's an **Open Source** project opened at SourceForge, named *plint*:

- web: http://plint.sourceforge.net
- admin: myself
- CVS access: :pserver:cvs.plint.sourforge.net:/cvsroot/plint
- mailing list

Chapter 2

Appendix

2.1 BNF Syntax

2.1.1 Core syntax

A *plint* source program is made up from two parts: header and main (top) procedure.
The header facilitates shell autorun feature and to settles the version of *plint* that should be used when interpretating as well as the plotters for which it was written; these last two can miss.

Main program, blocks, sections

⟨*interpret*⟩ ::= ⟨*header*⟩ ⟨*block*⟩

⟨*header*⟩ ::= ⟨*autorun*⟩ ⟨*version*⟩ ⟨*plotters*⟩ `plint` `STRING`

⟨*autorun*⟩ ::= ⟨*empty*⟩ | `autorun`

⟨*version*⟩ ::= ⟨*empty*⟩ | `version` `INTEGER` '`,`' `INTEGER`

⟨*plotters*⟩ ::= ⟨*empty*⟩ | `plotter` pltlist

⟨*pltlist*⟩ ::= `STRING` | pltlist '`,`' `STRING`

⟨*block*⟩ ::= '`{`' blackhole '`}`'

⟨*blackhole*⟩ ::= ⟨*empty*⟩ | ⟨*blackhole*⟩ ⟨*section*⟩

⟨*section*⟩ ::= ⟨*structure*⟩ | ⟨*data*⟩ | ⟨*flow*⟩ | ⟨*io*⟩ | ⟨*plugins_section*⟩

Structure

⟨*structure*⟩ ::= ⟨*files*⟩ | ⟨*blocks*⟩ | ⟨*plugins_structure*⟩

⟨*files*⟩ ::= ⟨*embed*⟩

⟨*blocks*⟩ ::= ⟨*named*⟩ | ⟨*anonymous*⟩

⟨*named*⟩ ::= ⟨*define*⟩

⟨*anonymous*⟩ ::= ⟨*block*⟩

⟨*embed*⟩ ::= ⟨*insinc*⟩ `STRING` ⟨*blackhole*⟩

⟨*define*⟩ ::= ⟨*procmet*⟩ `IDENTIFIER` ‘`(`’ ⟨*namelist*⟩ ‘`)`’ ⟨*hook*⟩ ⟨*block*⟩

⟨*procmet*⟩ ::= `procedure` | `method`

⟨*namelist*⟩ ::= ⟨*empty*⟩ | namelist1

⟨*namelist1*⟩ ::= ⟨*nameitem*⟩ | ⟨*namelist*⟩ ‘`,`’ ⟨*nameitem*⟩

⟨*nameitem*⟩ ::= `IDENTIFIER`

⟨*hook*⟩ ::= ⟨*plugins_hook*⟩

Data

⟨*data*⟩ ::= ⟨*variables*⟩ | ⟨*plugins_data*⟩

⟨*variables*⟩ ::= ⟨*new*⟩ | ⟨*assignment*⟩ | ⟨*operasg*⟩

Creation

⟨*new*⟩ ::= ‘‘’ ⟨*type*⟩ ⟨*newlist*⟩ ‘`;`’ ⟨*iterate*⟩

⟨*newlist*⟩ ::= ⟨*newitem*⟩ | ⟨*newlist*⟩ ‘`,`’ ⟨*newitem*⟩

⟨*newitem*⟩ ::= ⟨*newididxs*⟩ | ⟨*newididxs*⟩ ⟨*args*⟩

⟨*newididxs*⟩ ::= ⟨*identifier*⟩ ⟨*idxlist*⟩

Assignment

⟨*assignment*⟩ ::= ⟨*asgnlist*⟩ ⟨*assignment1*⟩

⟨*assignment1*⟩ ::= ‘`=`’ ⟨*asgnsrc*⟩ ‘`;`’ ⟨*iterate*⟩
| ⟨*asgoper*⟩ ⟨*rvalue*⟩ ‘`;`’ ⟨*iterate*⟩

⟨*asgnlist*⟩ ::= ⟨*asgnitem*⟩ | ⟨*asgnlist*⟩ ‘`,`’ ⟨*asgnitem*⟩

⟨*asgnitem*⟩ ::= ⟨*asgndst*⟩

⟨*asgndst*⟩ ::= ⟨*lvalue*⟩
| ‘=’ ⟨*lvalue*⟩

⟨*asgnsrc*⟩ ::= ⟨*nulvar*⟩
| ⟨*rvalue*⟩

⟨*operasg*⟩ ::= ⟨*asgoper*⟩ ‘(’ ⟨*reference*⟩ ‘,’ ⟨*tabvar*⟩ ‘)’ ⟨*iterate*⟩

Flow

⟨*flow*⟩ ::= ‘;’ | ⟨*primitives*⟩ | ⟨*control*⟩ | ⟨*sideeffect*⟩ | ⟨*plugins_flow*⟩

Primitives

⟨*primitives*⟩ ::= `reset` ‘(’ ‘)’
| `flush` ‘(’ ‘)’
| `select` ‘(’ ⟨*numvar*⟩ ‘)’
| `moveto` ‘(’ ⟨*pntarg*⟩ ‘)’
| `moverel` ‘(’ ⟨*pntarg*⟩ ‘)’
| `line` ‘(’ ⟨*zptarg*⟩ ‘)’
| `lineto` ‘(’ ⟨*pntarg*⟩ ‘)’
| `linerel` ‘(’ ⟨*pntarg*⟩ ‘)’
| `square` ‘(’ ⟨*pntarg*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `squarerel` ‘(’ ⟨*numvar*⟩ ‘)’
| `rectangle` (’ ⟨*zptarg*⟩ ‘)’
| `rectanglerel` ‘(’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `circle` ‘(’ ⟨*pntarg*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `circle` (’ ⟨*pntarg*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `circlerel` ‘(’ ⟨*numvar*⟩ ‘)’
| `circlerel` ‘(’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `ellipse` ‘(’ ⟨*pntarg*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `ellipse` ‘(’ ⟨*pntarg*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `ellipserel` ‘(’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `ellipserel` ‘(’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `polyline` ‘(’ ⟨*tabvar*⟩ ‘)’
| `polylinerel` ‘(’ ⟨*tabvar*⟩ ‘)’
| ⟨*plugins_primitives*⟩

Side effects

⟨*sideeffect*⟩ ::= ⟨*call*⟩ ‘;’ | ⟨*idsevar*⟩ ‘;’

⟨*call*⟩ ::= ⟨*procname*⟩ ⟨*args*⟩
| ⟨*methname*⟩ ⟨*args*⟩
| ⟨*methcall*⟩ ⟨*args*⟩

⟨*procname*⟩ ::= ⟨*identifier*⟩ ⟨*methname*⟩ ::= ⟨*xxxvar*⟩ ‘‘’ ⟨*identifier*⟩ ⟨*action*⟩

⟨*methcall*⟩ ::= ⟨*xxxvar*⟩ ‘‘’

Flow control

⟨*ctlcnt*⟩ ::= ⟨*empty*⟩ | ⟨*numvar*⟩

⟨*control*⟩ ::= ⟨*blockctrl*⟩ | ⟨*procctrl*⟩

⟨*procctrl*⟩ ::= ⟨*return*⟩ | ⟨*escape*⟩ | ⟨*exit*⟩

⟨*exit*⟩ ::= `exit`

⟨*return*⟩ ::= `return` ⟨*ctlcnt*⟩ ‘`;`’

⟨*escape*⟩ ::= `escape` ⟨*ctlcnt*⟩ ‘`;`’

⟨*blockctrl*⟩ ::= ⟨*break*⟩ | ⟨*continue*⟩ | ⟨*contloop*⟩

⟨*break*⟩ ::= `break` ⟨*ctlcnt*⟩ ‘`;`’

⟨*continue*⟩ ::= `continue` ⟨*ctlcnt*⟩ ‘`;`’

⟨*contloop*⟩ ::= `continue loop` ⟨*ctlcnt*⟩ ‘`;`’

Input/Ouptut

⟨*io*⟩ ::= `echo version` ‘`;`’ | ⟨*eco*⟩ | ⟨*sync*⟩

⟨*eco*⟩ ::= `echo` ⟨*ecolist*⟩ ‘`;`’

⟨*ecolist*⟩ ::= ⟨*ecoelist*⟩ | ⟨*ecolist*⟩ ‘`,`’ ⟨*ecoelist*⟩

⟨*ecoelist*⟩ ::= ⟨*nulvar*⟩ | ⟨*rvalue*⟩

⟨*sync*⟩ ::= `sync` ⟨*synclist*⟩ ‘`;`’

⟨*synclist*⟩ ::= ⟨*syncelist*⟩ | ⟨*synclist*⟩ ‘`,`’ ⟨*syncelist*⟩

⟨*syncelist*⟩ ::= ⟨*rvalue*⟩

Helper rules

⟨*args*⟩ ::= ‘(’ ⟨*tablist*⟩ ⟨*maketab*⟩ ‘)’
  | ‘!’ ‘(’ ⟨*tabvar*⟩ ‘)’

⟨*pntarg*⟩ ::= ⟨*point*⟩ | ⟨*number*⟩ ‘,’ ⟨*number*⟩

⟨*zptarg*⟩ ::= ⟨*zpoint*⟩ | ⟨*pntarg*⟩ ‘,’ ⟨*pntarg*⟩

⟨*asgoper*⟩ ::= `+=` | `-=` | `*=` | `/=` | `%=` | `^=` | `&=` | `|=` | `~=`

⟨*fncoper*⟩ ::= ‘+’ | ‘-’ | ‘*’ | ‘/’ | ‘%’ | ‘^’ | ‘&’ | ‘|’ | ‘~’

⟨*ppmmoper*⟩ ::= `++` | `--`

⟨*cmpoper*⟩ ::= ‘<’ | ‘>’ | `<=` | `>=` | `==` | `=|=`

⟨*cnvoper*⟩ ::= ‘$’ | ‘@’ | ‘"’ | ‘?’ | ‘!’ | ‘#’

⟨*rmoper*⟩ ::= ‘,’ | ‘:’

⟨*metaoper*⟩ ::= `new` | `echo` | `read` | `sync` | ‘<’ ‘>’ | ‘(’ ‘)’

⟨*lvalue*⟩ ::= ⟨*reference*⟩
  | ⟨*plugins_lvalues*⟩

⟨*rvalue*⟩ ::= ⟨*blkvar*⟩
  | ⟨*rnew*⟩ ⟨*identifier*⟩ ⟨*args*⟩ ⟨*action*⟩ ⟨*idxlist*⟩
  | ⟨*plugins_rvalues*⟩

⟨*rnew*⟩ ::= `new` | ‘`’

⟨*identifier*⟩ ::= `IDENTIFIER`
  | `this` | `self`
  | ‘"’ ⟨*strconv*⟩ ‘"’
  | ‘<’ ⟨*string*⟩ ‘>’
  | ⟨*plugins_identifiers*⟩

⟨*valtype*⟩ ::= `nul` | `number` | `integer` | `double` | `point` | `string` | `boolean` | `table` | `hash`
  | `type` ‘"’ ⟨*strconv*⟩ ‘"’
  | `type` ‘<’ ⟨*string*⟩ ‘>’
  | `type` ‘(’ ⟨*blkvar*⟩ ‘)’

⟨*vartype*⟩ ::= ‘(’ `type` ‘)’ ⟨*variable*⟩

⟨*type*⟩ ::= ⟨*identifier*⟩
  | ⟨*vartype*⟩

⟨*cast*⟩ ::= ‘<’ ⟨*type*⟩ ‘>’

⟨*at*⟩ ::= ⟨*empty*⟩
| `at` ⟨*numvar*⟩
| `at last`
| `at last` ⟨*numvar*⟩

⟨*id*⟩ ::= ⟨*identifier*⟩ | `argv`
| ‘_’ | `0_` | `1_` | `2_` | `3_` | `4_` | `5_` | `6_` | `7_` | `8_` | `9_`

⟨*idxlist*⟩ ::= ⟨*empty*⟩ | ⟨*idxlist*⟩ ⟨*index*⟩

⟨*index*⟩ ::= ‘$’ ⟨*id*⟩
| ‘[’ ⟨*numvar*⟩ ‘]’
| ‘.’ ⟨*identifier*⟩

⟨*rangelist*⟩ ::= ⟨*rangeitem*⟩ | ⟨*rangelist*⟩ ‘:’ ⟨*rangeitem*⟩

⟨*rangeitem*⟩ ::= ⟨*empty*⟩
| ⟨*number*⟩
| ⟨*number*⟩ ‘,’
| ‘,’ ⟨*number*⟩
| ⟨*number*⟩ ‘,’ ⟨*number*⟩
| ‘,’

References

⟨*reference*⟩ ::= ⟨*idref*⟩ | ⟨*callref*⟩

⟨*idref*⟩ ::= ⟨*id*⟩ ⟨*refindex*⟩

⟨*callref*⟩ ::= ⟨*call*⟩ ⟨*refindex*⟩

⟨*refindex*⟩ ::= ⟨*empty*⟩ | ⟨*refindex*⟩ ⟨*indexref*⟩

⟨*indexref*⟩ ::= ’$’ ⟨*id*⟩
| ‘[’ ⟨*number*⟩ ‘]’
| ‘.’ ⟨*identifier*⟩
| ‘[’ ⟨*string*⟩ ‘]’

Variabiles

⟨*refvar*⟩ ::= ⟨*reference*⟩ | ⟨*castvar*⟩

⟨*castvar*⟩ ::= ⟨*cast*⟩ ⟨*idref*⟩ ⟨*action*⟩
| ⟨*cast*⟩ ‘(’ ⟨*blkvar*⟩ ‘)’ ⟨*action*⟩
| ⟨*cast*⟩ ⟨*castvar*⟩ ⟨*action*⟩

⟨*idsevar*⟩ ::= ⟨*incdecvar*⟩ ⟨*action*⟩

⟨*incdecvar*⟩ ::= ⟨*ppmmoper*⟩ ⟨*reference*⟩

⟨*bltinvar*⟩ ::= ⟨*refvar*⟩ | ⟨*idsevar*⟩ | '(' ')'

⟨*opervar*⟩ ::= ⟨*variable*⟩ ⟨*fncoper*⟩ ⟨*variable*⟩
| ⟨*fncoper*⟩ ⟨*variable*⟩
| ⟨*variable*⟩ ⟨*ppmmoper*⟩
| ⟨*variable*⟩ '{' ⟨*rangelist*⟩ '}' ⟨*iterate*⟩

⟨*basicvar*⟩ ::= ⟨*opervar*⟩ ⟨*action*⟩

⟨*mixtvar*⟩ ::= `read` ⟨*type*⟩ ⟨*action*⟩
| '[' ⟨*tabvar*⟩ ']' ⟨*number*⟩

⟨*textvar*⟩ ::= '(' ⟨*blkvar*⟩ ')'
| ⟨*bltinvar*⟩
| ⟨*basicvar*⟩
| ⟨*mixtvar*⟩
| ⟨*plugins_var*⟩

⟨*variable*⟩ ::= ⟨*textvar*⟩

⟨*xxxvar*⟩ ::= ⟨*refvar*⟩ | ⟨*idsevar*⟩

⟨*blkvar*⟩ ::= ⟨*block*⟩ | ⟨*variable*⟩

⟨*nulvar*⟩ ::= ⟨*empty*⟩

⟨*cmpvar*⟩ ::= ⟨*variable*⟩ ⟨*cmpoper*⟩ ⟨*variable*⟩

Integer/Real numbers

⟨*bltinnum*⟩ ::= `INTEGER` | `DOUBLE`
| `len` '(' ⟨*blkvar*⟩ ')'
| `len` '(' ⟨*number*⟩ ')'
| `len` '(' ⟨*point*⟩ ')'
| `len` '(' ⟨*string*⟩ ')'
| `len` '(' ⟨*boolean*⟩ ')'
| `len` '(' ⟨*table*⟩ ')'
| `len` '(' ⟨*hash*⟩ ')'
| `rnd` '(' ⟨*numvar*⟩ ')'
| `min` '(' ⟨*numvar*⟩ ',' ⟨*numvar*⟩ ')'
| `max` '(' ⟨*numvar*⟩ ',' ⟨*numvar*⟩ ')'
| `fork` '(' ')'

| lev ‘(’ ‘)’
| lev ‘(’ ⟨*numvar*⟩ ‘)’

⟨*basicnum*⟩ ::= ⟨*number*⟩ ‘+’ ⟨*number*⟩
| ⟨*number*⟩ ‘-’ ⟨*number*⟩
| ⟨*number*⟩ ‘*’ ⟨*number*⟩
| ⟨*number*⟩ ‘/’ ⟨*number*⟩
| ⟨*number*⟩ ‘%’ ⟨*number*⟩
| ⟨*number*⟩ ‘&’ ⟨*number*⟩
| ⟨*number*⟩ ‘|’ ⟨*number*⟩
| ⟨*number*⟩ ‘^’ ⟨*number*⟩
| ‘+’ ⟨*number*⟩
| ‘-’ ⟨*number*⟩
| ‘/’ ⟨*number*⟩
| ‘~’ ⟨*number*⟩

⟨*mixtnum*⟩ ::= ‘*’ ⟨*string*⟩
| crtx | crty | maxx | maxy
| pntx ‘(’ ⟨*pntvar*⟩ ‘)’
| pnty ‘(’ ⟨*pntvar*⟩ ‘)’

⟨*textnum*⟩ ::= ‘(’ ⟨*number*⟩ ‘)’
| ⟨*bltinnum*⟩
| ⟨*basicnum*⟩
| ⟨*mixtnum*⟩
| ⟨*plugins_num*⟩

⟨*number*⟩ ::= ⟨*textnum*⟩
| ‘$’ ⟨*numconv*⟩

⟨*numconv*⟩ ::= ⟨*numconvar*⟩ ⟨*action*⟩

⟨*numconvar*⟩ ::= ⟨*numblkvar*⟩ | ⟨*numxxxvar*⟩

⟨*numblkvar*⟩ ::= ‘(’ ⟨*rvalue*⟩ ‘)’

⟨*numxxxvar*⟩ ::= ⟨*xxxvar*⟩

⟨*numvar*⟩ ::= ⟨*number*⟩ | ⟨*blkvar*⟩

2D Points

⟨*bltinpnt*⟩ ::= crtp
| ‘(’ ⟨*number*⟩ ‘,’ ⟨*number*⟩ ‘)’
| ‘[’ ⟨*number*⟩ ‘,’ ⟨*number*⟩ ‘]’

$\langle basicpnt \rangle$ ::= ‘`-`’ $\langle point \rangle$
| ‘`/`’ $\langle point \rangle$
| $\langle point \rangle$ ‘`+`’ $\langle point \rangle$
| $\langle point \rangle$ ‘`-`’ $\langle point \rangle$
| $\langle point \rangle$ ‘`*`’ $\langle point \rangle$

$\langle mixtpnt \rangle$ ::= $\langle zptpnt \rangle$
| $\langle number \rangle$ ‘`*`’ $\langle point \rangle$
| $\langle point \rangle$ ‘`+`’ $\langle number \rangle$
| $\langle point \rangle$ ‘`-`’ $\langle number \rangle$
| $\langle point \rangle$ ‘`*`’ $\langle number \rangle$
| $\langle point \rangle$ ‘`/`’ $\langle number \rangle$
| $\langle point \rangle$ ‘`%`’ $\langle number \rangle$

$\langle zptpnt \rangle$ ::= `P1` ‘`(`’ $\langle zpoint \rangle$ ‘`)`’ | `P2` ‘`(`’ $\langle zpoint \rangle$ ‘`)`’

$\langle textpnt \rangle$ ::= ‘`(`’ $\langle point \rangle$ ‘`)`’
| $\langle bltinpnt \rangle$
| $\langle basicpnt \rangle$
| $\langle mixtpnt \rangle$
| $\langle plugins\_pnt \rangle$

$\langle point \rangle$ ::= $\langle textpnt \rangle$
| ‘`@`’ $\langle pntconv \rangle$

$\langle pntconv \rangle$ ::= $\langle pntconvar \rangle$ $\langle action \rangle$

$\langle pntconvar \rangle$ ::= $\langle pntblkvar \rangle$ | $\langle pntxxxvar \rangle$

$\langle pntblkvar \rangle$ ::= ‘`(`’ $\langle rvalue \rangle$ ‘`)`’

$\langle pntxxxvar \rangle$ ::= $\langle xxxvar \rangle$

$\langle pntvar \rangle$ ::= $\langle point \rangle$ | $\langle blkvar \rangle$

$\langle bltin2pt \rangle$ ::= `frame` | ‘`[`’ $\langle point \rangle$ ‘`,`’ $\langle point \rangle$ ‘`]`’ | ‘`(`’ $\langle point \rangle$ ‘`,`’ $\langle point \rangle$ ‘`)`’

$\langle text2pt \rangle$ ::= $\langle bltin2pt \rangle$ | $\langle plugins\_2pt \rangle$

$\langle Zpoint \rangle$ ::= $\langle text2pt \rangle$

Strings

$\langle bltinstr \rangle$ ::= `STRING` | `plotter`

⟨*basicstr*⟩ ::= ⟨*string*⟩ ‘+’ ⟨*string*⟩
 | ⟨*string*⟩ ‘-’ ⟨*string*⟩
 | ⟨*string*⟩ ‘%’ ⟨*string*⟩
 | ⟨*string*⟩ ‘~’ ⟨*string*⟩
 | ‘+’ ⟨*string*⟩
 | ‘-’ ⟨*string*⟩
 | ‘/’ ⟨*string*⟩
 | ⟨*string*⟩ ‘{’ rangelist ‘}’
 | ⟨*string*⟩ ‘/’ ⟨*string*⟩ ‘/’ ⟨*string*⟩

⟨*mixtstr*⟩ ::= ⟨*string*⟩ ‘*’ ⟨*number*⟩
 | ‘"’ ⟨*identifier*⟩ ‘"’
 | ‘"’ `type` ‘(’ ⟨*variable*⟩ ‘)’

⟨*textstr*⟩ ::= ‘(’ ⟨*string*⟩ ‘)’
 | ⟨*basicstr*⟩
 | ⟨*bltinstr*⟩
 | ⟨*mixtstr*⟩
 | ⟨*plugins_str*⟩

⟨*string*⟩ ::= ⟨*textstr*⟩
 | ‘"’ ⟨*strconv*⟩

⟨*strconv*⟩ ::= ⟨*strconvar*⟩ ⟨*action*⟩

⟨*strconvar*⟩ ::= ⟨*strblkvar*⟩ | ⟨*strxxxvar*⟩

⟨*strblkvar*⟩ ::= ‘(’ ⟨*rvalue*⟩ ‘)’

⟨*strxxxvar*⟩ ::= ⟨*xxxvar*⟩

⟨*strvar*⟩ ::= ⟨*string*⟩ | ⟨*blkvar*⟩

Booleens - true/false

⟨*bltinbol*⟩ ::= `true` | `false`
 | `last`
 | `var` ⟨*id*⟩ | `var` ‘(’ ⟨*id*⟩ ⟨*at*⟩ ‘)’

⟨*basicbol*⟩ ::= `~~` ⟨*boolean*⟩
 | ⟨*boolean*⟩ `&&` ⟨*boolean*⟩
 | ⟨*boolean*⟩ `||` ⟨*boolean*⟩
 | ⟨*boolean*⟩ `^^` ⟨*boolean*⟩
 | ⟨*boolean*⟩ `==` ⟨*boolean*⟩
 | ⟨*boolean*⟩ `=|=` ⟨*boolean*⟩

```
                | ⟨boolean⟩ <= ⟨boolean⟩
                | ⟨boolean⟩ >= ⟨boolean⟩

⟨mixtbol⟩     ::= ⟨cmpvar⟩ ⟨action⟩
                | ⟨valtype⟩ == ⟨valtype⟩
                | ⟨valtype⟩ =|= ⟨valtype⟩
                | ⟨cast⟩ == ⟨cast⟩
                | ⟨cast⟩ =|= ⟨cast⟩
                | ⟨number⟩ == ⟨number⟩
                | ⟨number⟩ =|= ⟨number⟩
                | ⟨number⟩ ‘<’ ⟨number⟩
                | ⟨number⟩ ‘>’ ⟨number⟩
                | ⟨number⟩ <= ⟨number⟩
                | ⟨number⟩ >= ⟨number⟩
                | ⟨point⟩ == ⟨point⟩
                | ⟨point⟩ =|= ⟨point⟩
                | ⟨point⟩ ‘<’ ⟨point⟩
                | ⟨point⟩ ‘>’ ⟨point⟩
                | ⟨point⟩ <= ⟨point⟩
                | ⟨point⟩ >= ⟨point⟩
                | ⟨string⟩ == ⟨string⟩
                | ⟨string⟩ =|= ⟨string⟩
                | ⟨string⟩ ‘<’ ⟨string⟩
                | ⟨string⟩ ‘>’ ⟨string⟩
                | ⟨string⟩ <= ⟨string⟩
                | ⟨string⟩ >= ⟨string⟩
                | ⟨table⟩ == ⟨table⟩
                | ⟨table⟩ =|= ⟨table⟩

⟨textbol⟩     ::= ‘(’ ⟨boolean⟩ ‘)’
                | ⟨bltinbol⟩
                | ⟨basicbol⟩
                | ⟨mixtbol⟩
                | ⟨plugins_bol⟩

⟨boolean⟩     ::= ⟨textbol⟩ | ‘?’ ⟨bolconv⟩

⟨bolconv⟩     ::= ⟨bolconvar⟩ ⟨action⟩

⟨bolconvar⟩   ::= ⟨bolblkvar⟩ | ⟨bolxxxvar⟩

⟨bolblkvar⟩   ::= ‘(’ ⟨rvalue⟩ ‘)’

⟨bolxxxvar⟩   ::= ⟨xxxvar⟩

⟨bolvar⟩      ::= ⟨boolean⟩ | ⟨blkvar⟩
```

Arrays

⟨*tablist*⟩ ::= ⟨*tabelist*⟩ | tablist ‘,’ ⟨*tabelist*⟩

⟨*tabelist*⟩ ::= ⟨*rvalue*⟩

⟨*tabfrm*⟩ ::= ‘{’ ⟨*tablist*⟩ ‘}’

⟨*bltintab*⟩ ::= ‘!’ ⟨*tabfrm*⟩
 | ⟨*table*⟩ ‘{’ rangelist ‘}’
 | `ARGS` | `ARGS` ‘(’ ⟨*at*⟩ ‘)’
 | `VAR` | `VAR` ‘(’ ⟨*at*⟩ ‘)’

⟨*basictab*⟩ ::= ⟨*table*⟩ ‘+’ ⟨*table*⟩ | ⟨*table*⟩ ‘*’ ⟨*number*⟩

⟨*mixttab*⟩ ::= ⟨*string*⟩ ‘/’ ⟨*string*⟩
 | ⟨*number*⟩ ‘+’ ⟨*table*⟩
 | ⟨*number*⟩ ‘*’ ⟨*table*⟩
 | ⟨*number*⟩ ‘-’ ⟨*table*⟩
 | ⟨*number*⟩ ‘/’ ⟨*table*⟩
 | ⟨*number*⟩ ‘%’ ⟨*table*⟩
 | ⟨*number*⟩ ‘&’ ⟨*table*⟩
 | ⟨*number*⟩ ‘|’ ⟨*table*⟩
 | ⟨*number*⟩ ‘^’ ⟨*table*⟩
 | ⟨*table*⟩ ‘+’ ⟨*number*⟩
 | ⟨*table*⟩ ‘-’ ⟨*number*⟩
 | ⟨*table*⟩ ‘/’ ⟨*number*⟩
 | ⟨*table*⟩ ‘%’ ⟨*number*⟩
 | ⟨*table*⟩ ‘&’ ⟨*number*⟩
 | ⟨*table*⟩ ‘|’ ⟨*number*⟩
 | ⟨*table*⟩ ‘^’ ⟨*number*⟩
 | ⟨*point*⟩ ‘+’ ⟨*table*⟩
 | ⟨*point*⟩ ‘-’ ⟨*table*⟩
 | ⟨*point*⟩ ‘*’ ⟨*table*⟩
 | ⟨*table*⟩ ‘-’ ⟨*point*⟩
 | ⟨*string*⟩ ‘+’ ⟨*table*⟩
 | ⟨*string*⟩ ‘-’ ⟨*table*⟩
 | ⟨*string*⟩ ‘%’ ⟨*table*⟩
 | ⟨*table*⟩ ‘+’ ⟨*string*⟩
 | ⟨*table*⟩ ‘-’ ⟨*string*⟩
 | ⟨*table*⟩ ‘%’ ⟨*string*⟩
 | ⟨*boolean*⟩ `&&` ⟨*table*⟩
 | ⟨*boolean*⟩ `||` ⟨*table*⟩
 | ⟨*boolean*⟩ `^^` ⟨*table*⟩
 | ⟨*table*⟩ `&&` ⟨*boolean*⟩

| ⟨table⟩ || ⟨boolean⟩
| ⟨table⟩ ^^ ⟨boolean⟩

⟨opertab⟩ ::= ⟨fncoper⟩ ‘(’ ⟨table⟩ ‘,’ ⟨rvalue⟩ ‘)’ ⟨iterate⟩
| ⟨fncoper⟩ ‘(’ ⟨blkvar⟩ ‘,’ ⟨table⟩ ‘)’ ⟨iterate⟩
| ⟨fncoper⟩ ‘(’ ⟨tabvar⟩ ‘)’ ⟨iterate⟩
| ⟨ppmmoper⟩ ‘(’ ⟨tabvar⟩ ‘)’ ⟨iterate⟩

⟨texttab⟩ ::= ‘(’ ⟨table⟩ ‘)’
| ⟨basictab⟩
| ⟨bltintab⟩
| ⟨mixttab⟩
| ‘!’ ⟨opertab⟩
| ⟨plugins_tab⟩

⟨table⟩ ::= ⟨texttab⟩
| ‘!’ ⟨tabconv⟩

⟨tabconv⟩ ::= ⟨tabconvar⟩ ⟨action⟩

⟨tabconvar⟩ ::= ⟨tabblkvar⟩ | ⟨tabxxxvar⟩

⟨tabblkvar⟩ ::= ‘(’ ⟨rvalue⟩ ‘)’

⟨tabxxxvar⟩ ::= ⟨xxxvar⟩

⟨tabvar⟩ ::= ⟨table⟩ | ⟨blkvar⟩

Hashes

⟨hshlist⟩ ::= ⟨hshelist⟩ | ⟨hshlist⟩ ‘,’ ⟨hshelist⟩

⟨hshelist⟩ ::= ⟨strvar⟩ ‘:’ ⟨rvalue⟩

⟨hshfrm⟩ ::= ‘[’ ⟨hshlist⟩ ‘]’

⟨bltinhsh⟩ ::= ‘#’ ⟨hshfrm⟩

⟨basichsh⟩ ::= ⟨hash⟩ ‘+’ ⟨hash⟩
| ⟨hash⟩ ‘|’ ⟨hash⟩
| ⟨hash⟩ ‘*’ ⟨hash⟩
| ⟨hash⟩ ‘&’ ⟨hash⟩
| ⟨hash⟩ ‘-’ ⟨hash⟩
| ⟨hash⟩ ‘/’ ⟨hash⟩

⟨*texthsh*⟩ ::= '(' ⟨*hash*⟩ ')'
| ⟨*basichsh*⟩
| ⟨*bltinhsh*⟩
| ⟨*plugins_hsh*⟩

⟨*hash*⟩ ::= ⟨*texthsh*⟩
| '#' ⟨*hshconv*⟩

⟨*hshconv*⟩ ::= ⟨*hshconvar*⟩ ⟨*action*⟩

⟨*hshconvar*⟩ ::= ⟨*hshblkvar*⟩ | ⟨*hshxxxvar*⟩

⟨*hshblkvar*⟩ ::= '(' ⟨*rvalue*⟩ ')'

⟨*hshxxxvar*⟩ ::= ⟨*xxxvar*⟩

⟨*hshvar*⟩ ::= ⟨*hash*⟩ | ⟨*blkvar*⟩

2.1.2 Plugins' syntax[1]

IO extension

⟨*io_wait*⟩ ::= `wait` ⟨*strvar*⟩ ';'

⟨*io_format*⟩ ::= `format nul` '=' ⟨*strvar*⟩ ';'
| `format integer` '=' ⟨*strvar*⟩ ',' ⟨*bolvar*⟩ ';'
| `format double` '=' ⟨*strvar*⟩ ';'
| `format point` '=' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ';'
| `format string` '=' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ';'
| `format boolean` '=' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ';'
| `format table` '=' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ';'
| `format hash` '=' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ',' ⟨*strvar*⟩ ';'

arrays

⟨*io_formatab*⟩ ::= `format` ⟨*valtype*⟩

The builtin plugin: "bltin"

data

⟨*bltin_ret*⟩ ::= ⟨*bltin_rvalues*⟩ ⟨*at*⟩ ';'

⟨*bltin_new*⟩ ::= `new` ⟨*bltin_new1*⟩ ⟨*bltin_new1*⟩ ::= ⟨*newtype*⟩ ⟨*bltin_newlist*⟩ ⟨*at*⟩ ';' ⟨*newtype*⟩ ::= ⟨*valtype*⟩
| ⟨*empty*⟩ ⟨*bltin_newlist*⟩ ::= ⟨*newididxs*⟩ | ⟨*bltin_newlist*⟩ ',' ⟨*newididxs*⟩

[1] Paragraphs are for plugin names, and subparagraphs are for the sections where the following rules are plugged.

rvalues

⟨*bltin_rvalues*⟩ ::= `read` ⟨*valtype*⟩ | ⟨*number*⟩ | ⟨*point*⟩ | ⟨*string*⟩ | ⟨*boolean*⟩ | ⟨*table*⟩ | ⟨*hash*⟩

Mathematical addon functions

numbers

⟨*numrem*⟩ ::= `rem` ‘(’ ⟨*numvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’

⟨*math*⟩ ::= ⟨*math1*⟩ ⟨*pntdbl*⟩ ::= ⟨*pntdbl1*⟩

⟨*math1*⟩ ::= `abs` ‘(’ ⟨*numvar*⟩ ‘)’
| `sgn` ‘(’ ⟨*numvar*⟩ ‘)’
| `floor` ‘(’ ⟨*numvar*⟩ ‘)’
| `ceil` ‘(’ ⟨*numvar*⟩ ‘)’
| `sin` ‘(’ ⟨*numvar*⟩ ‘)’
| `cos` ‘(’ ⟨*numvar*⟩ ‘)’
| `tan` ‘(’ ⟨*numvar*⟩ ‘)’
| `asin` ‘(’ ⟨*numvar*⟩ ‘)’
| `acos` ‘(’ ⟨*numvar*⟩ ‘)’
| `atan` ‘(’ ⟨*numvar*⟩ ‘)’
| `sec` ‘(’ ⟨*numvar*⟩ ‘)’
| `csc` ‘(’ ⟨*numvar*⟩ ‘)’
| `cot` ‘(’ ⟨*numvar*⟩ ‘)’
| `r2d` ‘(’ ⟨*numvar*⟩ ‘)’
| `d2r` ‘(’ ⟨*numvar*⟩ ‘)’
| `exp` ‘(’ ⟨*numvar*⟩ ‘)’
| `log` ‘(’ ⟨*numvar*⟩ ‘)’
| `exp2` ‘(’ ⟨*numvar*⟩ ‘)’
| `log2` ‘(’ ⟨*numvar*⟩ ‘)’
| `exp10` ‘(’ ⟨*numvar*⟩ ‘)’
| `log10` ‘(’ ⟨*numvar*⟩ ‘)’
| `sqrt` ‘(’ ⟨*numvar*⟩ ‘)’
| `sinh` ‘(’ ⟨*numvar*⟩ ‘)’
| `cosh` ‘(’ ⟨*numvar*⟩ ‘)’
| `tanh` ‘(’ ⟨*numvar*⟩ ‘)’
| `asinh` ‘(’ ⟨*numvar*⟩ ‘)’
| `acosh` ‘(’ ⟨*numvar*⟩ ‘)’
| `atanh` ‘(’ ⟨*numvar*⟩ ‘)’

⟨*pntdbl1*⟩ ::= `mod` ‘(’ ⟨*pntvar*⟩ ‘)’ | `ang` ‘(’ ⟨*pntvar*⟩ ‘)’

points

⟨*pntdiv*⟩ ::= ⟨*point*⟩ ‘/’ ⟨*point*⟩

“freetype” library based text rendering

primitives

⟨*freetype_wff*⟩ ::= `font` ‘(’ ⟨*string*⟩ ‘)’
| `face` ‘(’ ⟨*string*⟩ ‘)’
| `face` ‘(’ ⟨*number*⟩ ‘)’
| `font` ‘(’ ⟨*number*⟩ ‘)’
| `font` ‘(’ ⟨*pntarg*⟩ ‘)’
| `align` ‘(’ ⟨*strvar*⟩ ‘)’
| `write` ‘(’ ⟨*strvar*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*pntarg*⟩ ‘)’
| `write` ‘(’ ⟨*strvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `write` ‘(’ ⟨*strvar*⟩ ‘)’

arrays

⟨*freetype_font*⟩ ::= `font` ‘(’ ‘)’

Zpoints

⟨*freetype_bbox*⟩ ::= `bbox` ‘(’ ⟨*strvar*⟩ ‘,’ ⟨*numvar*⟩ ‘,’ ⟨*pntarg*⟩ ‘)’
| `bbox` ‘(’ ⟨*strvar*⟩ ‘,’ ⟨*numvar*⟩ ‘)’
| `bbox` ‘(’ ⟨*strvar*⟩ ‘)’

System interface

flow

⟨*shell*⟩ ::= `shell` ⟨*strvar*⟩ ‘;’

numbers

⟨*sys_num*⟩ ::= `pid` | `ppid`

strings

⟨*sys_str*⟩ ::= `user` | `host`

arrays

⟨*sys_tab*⟩ ::= `uname` | `time`

Plot points at minimum 1 cm/mm/in/pt away

primitives

⟨*unit_use*⟩ ::= `unit` ‘(’ number ‘,’ string ‘)’ | `unit` ‘(’ point ‘)’

“if-else” block

flow

⟨*ifelse*⟩ ::= ⟨*if*⟩ | ⟨*if*⟩ `else` ⟨*block*⟩ | ⟨*if*⟩ `else` ⟨*ifelse*⟩

⟨*if*⟩ ::= `if` ‘`(`’ ⟨*bolvar*⟩ ‘`)`’ ⟨*block*⟩

“case” block

flow

⟨*case*⟩ ::= `case` ” ⟨*cases*⟩ ”

⟨*cases*⟩ ::= ⟨*casee_num*⟩ ⟨*cases_num*⟩
| ⟨*casee_str*⟩ ⟨*cases_str*⟩
| ⟨*casee_pnt*⟩ ⟨*cases_pnt*⟩

⟨*casee_num*⟩ ::= ‘`(`’ ⟨*number*⟩ ‘`)`’

⟨*cases_num*⟩ ::= ⟨*empty*⟩
| ⟨*case_num_list*⟩ ‘`;`’ ⟨*block*⟩ ⟨*cases_num*⟩
| `else` ⟨*block*⟩

⟨*case_num_list*⟩ ::= ⟨*case_num_item*⟩ | ⟨*case_num_list*⟩ ‘`:`’ ⟨*case_num_item*⟩

⟨*case_num_item*⟩ ::= ⟨*numvar*⟩ | ⟨*numvar*⟩ ‘`,`’ ⟨*numvar*⟩

⟨*casee_str*⟩ ::= ‘`(`’ ⟨*string*⟩ ‘`)`’

⟨*cases_str*⟩ ::= ⟨*empty*⟩
| ⟨*case_str_list*⟩ ‘`;`’ ⟨*block*⟩ ⟨*cases_str*⟩
| `else` ⟨*block*⟩

⟨*case_str_list*⟩ ::= ⟨*case_str_item*⟩ | ⟨*case_str_list*⟩ ‘`:`’ ⟨*case_str_item*⟩

⟨*case_str_item*⟩ ::= ⟨*string*⟩

⟨*casee_pnt*⟩ ::= ‘`(`’ ⟨*point*⟩ ‘`)`’

⟨*cases_pnt*⟩ ::= ⟨*empty*⟩
| ⟨*case_pnt_list*⟩ ‘`;`’ ⟨*block*⟩ ⟨*cases_pnt*⟩
| `else` ⟨*block*⟩

⟨*case_pnt_list*⟩ ::= ⟨*case_pnt_item*⟩ | ⟨*case_pnt_list*⟩ ‘`:`’ ⟨*case_pnt_item*⟩

⟨*case_pnt_item*⟩ ::= ⟨*point*⟩

map between operator literals and their string “type”-representation

strings

⟨oper_str⟩ ::= operator ⟨asgoper⟩
| operator ⟨rmoper⟩
| operator ⟨metaoper⟩
| ‘<’ ‘>’ operator
| operator ⟨ppmmoper⟩
| ⟨ppmmoper⟩ operator
| operator ‘.’ ⟨fncoper⟩
| operator ⟨fncoper⟩
| operator ⟨cmpoper⟩
| operator ⟨cnvoper⟩
| operator ‘=’
| operator ‘ ’

OO programming based on the “class” paradigm

data

⟨class_new⟩ ::= class ⟨class_name⟩ ⟨class_inherit⟩ ‘{’ ⟨class_operators⟩ ⟨class_methods⟩ ⟨class_datas⟩ ⟨class_friends⟩ ‘}’

⟨class_name⟩ ::= ⟨identifier⟩

⟨class_inherit⟩ ::= ⟨empty⟩ | ⟨class_item⟩

⟨class_item⟩ ::= ‘:’ ⟨identifier⟩

⟨class_operators⟩ ::= ⟨empty⟩ | ⟨class_operator⟩ ‘;’ ⟨class_operators⟩

⟨class_operator⟩ ::= ⟨class_op_left⟩ operator ⟨class_item⟩
| operator ⟨class_op_right⟩ ⟨class_item⟩

⟨class_op_right⟩ ::= ⟨asgoper⟩
| ⟨fncoper⟩
| ⟨cmpoper⟩
| ‘ ’
| ‘=’
| ⟨rmoper⟩
| ⟨ppmmoper⟩
| operator
| ‘(’ ‘)’

⟨*class_op_left*⟩ ::= ⟨*fncoper*⟩
| ⟨*ppmmoper*⟩
| ⟨*cnvoper*⟩
| ⟨*metaoper*⟩

⟨*class_methods*⟩ ::= ⟨*empty*⟩ | ⟨*class_method*⟩ ‘`;`’ ⟨*class_methods*⟩

⟨*class_method*⟩ ::= `method` ⟨*identifier*⟩ ⟨*class_item*⟩

⟨*class_datas*⟩ ::= ⟨*empty*⟩ | `data` ⟨*class_datas1*⟩ ‘`;`’

⟨*class_datas1*⟩ ::= ⟨*class_data*⟩ | ⟨*class_datas1*⟩ ‘`,`’ ⟨*class_data*⟩

⟨*class_data*⟩ ::= ⟨*class_data1*⟩ ⟨*class_data_init*⟩

⟨*class_data1*⟩ ::= ⟨*identifier*⟩ | `const` ⟨*identifier*⟩

⟨*class_data_init*⟩ ::= ⟨*empty*⟩ | ‘`=`’ ⟨*rvalue*⟩

⟨*class_friends*⟩ ::= ⟨*empty*⟩ | `friend` ⟨*class_friends1*⟩ ‘`;`’

⟨*class_friends1*⟩ ::= ⟨*class_friend*⟩ | ⟨*class_friends1*⟩ ‘`,`’ ⟨*class_friend*⟩

⟨*class_friend*⟩ ::= ⟨*type*⟩

identifiers

⟨*class_base_id*⟩ ::= `base`

strings

⟨*class_class*⟩ ::= `class` ‘`(`’ ⟨*blkvar*⟩ ‘`)`’

hook

⟨*class_hook*⟩ ::= ‘`:`’ ⟨*class_base_self*⟩ ‘`(`’ ⟨*tablist*⟩ ‘`)`’

⟨*class_base_self*⟩ ::= `base` | `self`

flow

⟨*class_base_new*⟩ ::= `METACLASS` `base` ‘`(`’ ⟨*tablist*⟩ ⟨*maketab*⟩ ‘`)`’

⟨*class_self_new*⟩ ::= `METACLASS` `self` ‘`(`’ ⟨*tablist*⟩ ⟨*maketab*⟩ ‘`)`’

The fixed number of times “loop”

flow

$\langle loop \rangle$ ::= `loop` '`(`' $\langle numvar \rangle$ '`)`' $\langle block \rangle$

The "while" conditional loop

flow

$\langle while \rangle$ ::= `while` '`(`' $\langle blackhole \rangle$ '`)`' $\langle block \rangle$

"for" iteration

flow

$\langle for \rangle$ ::= `for` '`(`' $\langle blackhole \rangle$ '`:`' $\langle blackhole \rangle$ '`:`' $\langle blackhole \rangle$ '`)`' $\langle block \rangle$

for each element of an array or hash run a block

flow

$\langle foreach\_tab \rangle$ ::= `foreach` '`(`' $\langle table \rangle$ '`)`' $\langle block \rangle$

$\langle foreach\_hsh \rangle$ ::= `foreach` '`(`' $\langle hash \rangle$ '`)`' $\langle block \rangle$