SiGMAL Specifications

Contents

[Syntax 2](#_Toc428701942)

[General 2](#_Toc428701943)

[Graphics 2](#_Toc428701944)

[Models 2](#_Toc428701945)

[Animations 2](#_Toc428701946)

[Statements and Keywords 2](#_Toc428701947)

[Declarations 2](#_Toc428701948)

[Loops 2](#_Toc428701949)

[Objects 2](#_Toc428701950)

[Model Compilation 3](#_Toc428701951)

[Native Model Library (excerpt) 3](#_Toc428701952)

[Source Code 3](#_Toc428701953)

[Compilation Step 1 4](#_Toc428701954)

[Compilation Step 2 5](#_Toc428701955)

[Compilation Step 3 5](#_Toc428701956)

[Conversion into ASCII STL 6](#_Toc428701957)

# Syntax

## General

SiGMAL is semi-whitespace-oriented. Each statement must be in a single line or end in an ellipsis (...) to denote continuation. Multiple statements may be on a single line if they are separated by a semicolon (;). Any amount and type of whitespace may be on the leading or trailing end of a line, as well as between statements, but words in statements must be separated by spaces. A line with no non-whitespace characters indicates the end of a definition or procedure.

## Graphics

Not within the scope of this project

## Models

Since model definitions are compiled non-sequentially, statements can be in any order. However, when there are conflicting statements, the first one is used. Each definition must begin with a name followed by "Model:" and contain point declarations as well as line and/or face declarations. They may also contain any number and combination of other valid declarations.

Valid statements:

* declarations
* generalizations

## Animations

Not within the scope of this project

# Statements and Keywords

## Declarations

Variable declarations and assignments begin with the variable name, while Point, Line, and Face declarations begin with the class and name. All declarations then have the keyword *is* (*are* in the case of arrays) followed by a value, a list of values, or undefined.

## Loops

Not within the scope of this project

## Objects

Not within the scope of this project

# Model Compilation

## Native Model Library (excerpt)

Hexahedron ***Model***:

Point A *is* undefined; Point B *is* undefined

Point C *is* undefined; Point D *is* undefined

Point E *is* undefined; Point F *is* undefined

Point G *is* undefined; Point H *is* undefined

Face top's vertices *are* A,B,C,D

Face bottom's vertices *are* E,F,G,H

Face front's vertices *are* A,D,H,E

Face right's vertices *are* A,B,F,E

Face left's vertices *are* D,C,G,H

Face back's vertices *are* B,C,G,F

Cuboid ***Model***:

TypeOf Hexahedron

height *is* undefined

width *is* undefined

depth *is* undefined

Point A *is* (height/2, width/2, depth/2)

Point B *is* (height/2, width/2, -depth/2)

Point C *is* (height/2, -width/2, -depth/2)

Point D *is* (height/2, -width/2, depth/2)

Point E *is* (-height/2, width/2, depth/2)

Point F *is* (-height/2, width/2, -depth/2)

Point G *is* (-height/2, -width/2, -depth/2)

Point H *is* (-height/2, -width/2, depth/2)

Cube ***Model***:

TypeOf Cuboid

length *is* undefined

height *is* length

width *is* length

depth *is* length

## Source Code

UnitCube ***Model*:**

TypeOf Cube

length *is* 1

## Compilation Step 1

The compiler begins by appending the code of any parent model, replacing the inheritance statement with the one in the parent code, if any. It then removes any redundant declarations and repeats.

UnitCube ***Model*:**

TypeOf Cuboid

length *is* 1

height *is* length

width *is* length

depth *is* length

UnitCube ***Model*:**

TypeOf Hexahedron

length *is* 1

height *is* length

width *is* length

depth *is* length

Point A *is* (height/2, width/2, depth/2)

Point B *is* (height/2, width/2, -depth/2)

Point C *is* (height/2, -width/2, -depth/2)

Point D *is* (height/2, -width/2, depth/2)

Point E *is* (-height/2, width/2, depth/2)

Point F *is* (-height/2, width/2, -depth/2)

Point G *is* (-height/2, -width/2, -depth/2)

Point H *is* (-height/2, -width/2, depth/2)

UnitCube ***Model***:

length *is* 1

height *is* length

width *is* length

depth *is* length

Point A *is* (height/2, width/2, depth/2)

Point B *is* (height/2, width/2, -depth/2)

Point C *is* (height/2, -width/2, -depth/2)

Point D *is* (height/2, -width/2, depth/2)

Point E *is* (-height/2, width/2, depth/2)

Point F *is* (-height/2, width/2, -depth/2)

Point G *is* (-height/2, -width/2, -depth/2)

Point H *is* (-height/2, -width/2, depth/2)

top Face *is* boundedby A,B,C,D

bottom Face *is* boundedby E,F,G,H

front Face *is* boundedby A,D,H,E

right Face *is* boundedby A,B,F,E

left Face *is* boundedby D,C,G,H

back Face *is* boundedby B,C,G,F

## Compilation Step 2

Once the compiler reaches the top level model, it replaces variable names with their values.

UnitCube ***Model***:

height *is* 1

width *is* 1

depth *is* 1

Point A *is* (height/2, width/2, depth/2)

Point B *is* (height/2, width/2, -depth/2)

Point C *is* (height/2, -width/2, -depth/2)

Point D *is* (height/2, -width/2, depth/2)

Point E *is* (-height/2, width/2, depth/2)

Point F *is* (-height/2, width/2, -depth/2)

Point G *is* (-height/2, -width/2, -depth/2)

Point H *is* (-height/2, -width/2, depth/2)

top Face *is* boundedby A,B,C,D

bottom Face *is* boundedby E,F,G,H

front Face *is* boundedby A,D,H,E

right Face *is* boundedby A,B,F,E

left Face *is* boundedby D,C,G,H

back Face *is* boundedby B,C,G,F

UnitCube ***Model***:

Point A *is* (1/2, 1/2, 1/2)

Point B *is* (1/2, 1/2, -1/2)

Point C *is* (1/2, -1/2, -1/2)

Point D *is* (1/2, -1/2, 1/2)

Point E *is* (-1/2, 1/2, 1/2)

Point F *is* (-1/2, 1/2, -1/2)

Point G *is* (-1/2, -1/2, -1/2)

Point H *is* (-1/2, -1/2, 1/2)

top Face *is* boundedby A,B,C,D

bottom Face *is* boundedby E,F,G,H

front Face *is* boundedby A,D,H,E

right Face *is* boundedby A,B,F,E

left Face *is* boundedby D,C,G,H

back Face *is* boundedby B,C,G,F

## Compilation Step 3

Finally, the code is converted into the target format or immediately used in an animation.

## Conversion into ASCII STL