SiGMAL v1.0 Specification

(in-dev version 1)

1 Introduction.................

2 Lexical Structure............

2.1 General.................

2.2 Points and Vectors......

2.3 Lines...................

2.4 Arrays and Hashes.......

2.5 Control Structures......

2.6 Objects.................

2.6.1 Member Objects...

2.7 Environments............

2.8 Actions.................

3 Keywords and Natives.........

3.1 Objects.................

3.2 Actions and Modifiers...

1 Introduction

Simple Graphics, Modeling, and Animation Language, or SiGMAL, is a Domain Specific Language focused on graphics. It was designed to be simple and easy to use without sacrificing functionality. This was done by making the syntax very English-like. SiGMAL was designed in 2012 and was inspired by Alice.

2 Lexical Structure

This section describes the syntax and structure of the various features of SiGMAL.

2.1 General

SiGMAL uses a line-by-line interpreter to force users to make easily read code. To aid programming further, it automatically ignores any words that do not belong, making commenting easier.

2.2 Points and Vectors

In an object definition, points are used to when defining the faces. Points are also used for drawing arcs in 2 dimensions. In actions, points can be used as the center of rotation or as the destination for moving. Vectors are used along with points to define a line.

To define a point, use coordinates in the format (x, y) for 2 dimensions and (x, y, z) for 3 dimensions. Vectors use the same format, but use angle brackets <> instead of parentheses (). The entire coordinate must be on the same line. The x, y, and z can be constants, variables, and/or equations. An increase in x represents forward, an increase in y represents right, and an increase in z represents up.

To refer to a single coordinate of a point (i.e. its x, y, or z) use the point's designation followed by the coordinate desired, e.g. the x coordinate of point A is referred to as Ax. Vectors use i for x, j for y, and k for z. Vectors can also use t if they are a velocity vector.

2.3 Lines

In an object definition, lines can be used as directions to extend a surface, or as axes of rotation when making a solid of revolution. In actions, they can be used for direction of motion or for set movement paths.

Lines can be defined 2 ways. One is with 2 endpoints, the other is with a starting point and a vector.

2.4 Arrays and Hashes

Arrays are used can are used when creating an instance of an object using the *withDimensions* modifier, in addition to traditional uses. Hashes are used in the traditional way- to have a set of key-value pairs. They are also used when defining the default of optional modifiers in action definitions.

Arrays use the format [item1, item2, ...]. They can be split on multiple lines, but each item must be on a single line. Hashes use the format {key1:val1, key2:val2, ...}. Like arrays, hashes can be on multiple lines, but the entire key-value pair must be on the same line. Keys without a value will default to null.

2.5 Control Structures

Control structures include conditionals (if statements) and loops. They can be used to determine how many times a block of code is to be executed, if at all.

If statements use the word *if* and then a test statement on one line, followed by a block of code. For additional cases, use the phrase *else if* and then a test statement, followed by a block of code. For the last case, you can use *else* without a test statement if you want to execute the following block of code if the scenario doesn't fit a case defined. All if statements end with the word *regardless* or *rgdls* for short.

For statements all start with *repeat*, have a block of code, and end with *while* or *until* followed by a test statement. *While* loops execute until the statement is false, while *until* loops execute until the statement is true.