

# Open Command-oriented Geometric Graphics Generator

OpenCG<sup>3</sup> Spec Version 0.2.9

Dong Nai-Jia <sup>1</sup>    Lin Yong-Hsiang <sup>2</sup>

<sup>1</sup>National Chiao Tung University  
Department of Computer Science

<sup>2</sup>National Taiwan University  
Department of Agricultural Chemistry

August 17, 2017

# Perspective Projection

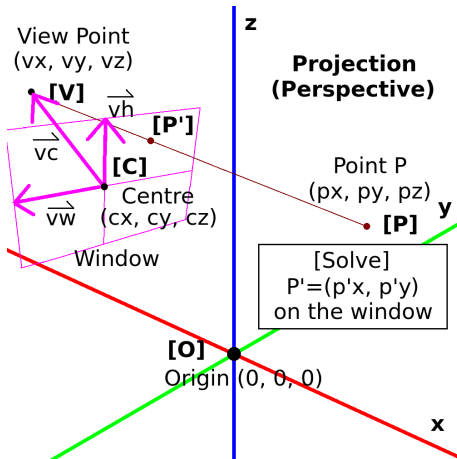


Figure: Projection in Euclidean  $\mathbb{R}^3$  Space

# Command Tokens

## Regular Expressions

$$\mathbb{N} := \{ \alpha \mid \alpha \in [0-9]^+ \}$$

$$\mathbb{R} := \{ \alpha \mid \alpha \in [+|-]?([0-9]*[.])?[0-9]^+ \}$$

$$\Rightarrow \mathbb{R} \supset \mathbb{N}$$

$$\mathbb{S} := \{ \alpha \mid \alpha \in '(.*)'|[.0-9A-Za-z+|-]^+ \}$$

$$\Rightarrow \mathbb{S} \supset \mathbb{R}$$

$$\mathbb{W} := \{ \alpha \mid \alpha \in [\backslash t] \}$$

$$\text{whitespace}$$

## Descriptions

- The matching mechanism abides by the maximal munch rule.
- Each command is whitespace-insensitive except being quoted by a pair of single quotation marks (').

# Command Grammars

## Context-Free Expansions

$\mathbf{C} \rightarrow \mathbf{AC} \mid ; \mid \text{EOL}$

$\mathbf{A} \rightarrow \mathbf{T(A)} \mid \mathbf{V(A)} \mid \mathbf{S(A)} \mid \mathbf{L(A)} \mid \mathbf{L(A,A,\dots,A)} \mid \mathbf{N} \mid \mathbf{R} \mid \mathbf{S}$

$$\begin{array}{l} \mathbf{T(\Pi)} \equiv \Pi : n \rangle \rightarrow ( \Sigma(\Pi, n) ) \\ \mathbf{V(\Pi)} \equiv \Pi : n \rangle \rightarrow < \Sigma(\Pi, n) > \\ \mathbf{S(\Pi)} \equiv \Pi : n \} \rightarrow \{ \Sigma(\Pi, n) \} \\ \mathbf{L(\Pi_1, \Pi_2, \dots, \Pi_{n-1}, \Pi_n)} \equiv \mathbb{L} [ \Pi_1 \Pi_2 \dots \Pi_{n-1} \Pi_n ] \rightarrow [ \Pi_1 \dots \Pi_n ] \end{array} \quad \left\| \quad \begin{array}{l} \Sigma(\Pi, n) \rightarrow \overbrace{\Pi \dots \Pi}^{n \text{ items}} \quad (\text{identical}) \\ \mathbf{L(\Pi)} \equiv \mathbb{L} [ \Pi : n ] \rightarrow [ \Sigma(\Pi, n) ] \end{array} \right.$$

## Descriptions

- Each command starts from  $\mathbf{C}$  and ends with a  $;$  or an EOL.
- Non-terminal symbol expansions are prior than function expansions except that symbols are used for describing arguments of a command.

# Command Parsing

## Escape Sequence

- `\x` is an escape sequence.
- If `x` is `\`, then it is treated as a single backslash.
- If `x` is EOL which may vary from platforms, then the sequence is omitted.
- Otherwise, the sequence is ignored and triggers a warning by default.

## Error Handling

- Physical lines are separated by an EOL.
- Logical lines are separated by either a semicolon or an unescaped EOL.
- If the command cannot be parsed by the grammar, then all the characters on the same logical line will be discarded.

# Class and Object System

## Classes

- Classes are split into two categories, top and bottom.
- Top classes are class window, class camera, and data classes.
- Bottom classes are class attrib and class group.
- Data classes are split into primitive classes and compound classes.
- Primitive classes are class point, etc.
- Compound classes are class line, class polygon, etc.

## Objects

- An object is instantiated from a class aforementioned.
- An object has an unique name throughout the category of its class.

## Relations

- References are bidirectional and can be created or deleted via commands.

# Create a Window

## Command

```
create window $ labelw (1)
```

## Parametres

- label<sup>w</sup> : the name of the object instantiated from the class window

## Examples

```
create window main
```

# Delete a Window

## Command

```
delete window $ labelw $ string (2)
```

## Parametres

- label<sup>w</sup> : the name of the object instantiated from the class window
- string : the text printed right after exiting the session

## Examples

```
delete window main  
delete window main 'Have a nice day.'
```



# Create a Camera

## Command

```
create camera S labelc  $\mathbb{R} : 3$ ) centre  $\mathbb{R} : 3$   $\rangle : 2$ ) plane  $\mathbb{R} : 3$   $\rangle$  sight (3)
```

## Parametres

- label<sup>c</sup> : the name of the object instantiated from the class camera
- centre : the world coordinate  $(c_x, c_y, c_z)$  of the centre of the viewport
- plane : the horizontal and the vertical vectors  $(\vec{v}_w, \vec{v}_h)$  of the viewport
- sight : the reverse line of sight  $\vec{v}_c$  from centre to the camera

## Examples

```
create camera z-top (0 0 1) (<1 0 0> <0 1 0>) <0 0 1>
```

# Attach a Camera

## Command

```
attach camera $ labelc $ labelw (4)
```

## Parametres

- label<sup>c</sup> : the name of the object instantiated from the class camera
- label<sup>w</sup> : the name of the object instantiated from the class window

## Examples

```
attach camera z-top main
```

# Create Points

## Command

create point  $\S$  label<sup>P</sup> : }  $\mathbb{R} : 3$  ) coord (5)  
 create point  $\S$  label<sup>P</sup> :  $\geq n$  )  $\mathbb{R} : 3$  ) coord : n ) (6)

## Parametres

- label<sup>P</sup> : the name of the object instantiated from the class point
- coord : the world coordinate  $(p_x, p_y, p_z)$  of the object named label<sup>P</sup>

## Examples

```
create point 'origin'    (0 0 0)
create point {X-1 X-2}  (1 0 0)
create point (Y-1 Z-1) ((0 1 0)(0 0 1))
```

# Delete Points

## Command

```
delete point  § labelP : } (7)
```

## Parametres

- label<sup>P</sup> : the name of the object instantiated from the class point

## Examples

```
delete point  origin  
delete point  {origin 'random-point'}
```

# Create Attributes

## Command

create attrib S attrib : } L [ L [ S class<sup>t</sup> S property A value ] : ] (8)

create attrib S attrib : ) L [ L [ S class<sup>t</sup> S property A value ] : ] (9)

## Parametres

- attrib : the name of the object instantiated from the class attrib
- class<sup>t</sup> : the name of one of the top classes
- property : the property of the object of class<sup>t</sup>
- value : the appropriate value of the property

## Examples

```
create attrib (magenta dashed-and-translucent-green) \
[[point fill-hsv (300 1.0 1.0)] \
[line style dashed] [line fill-rgba [(0 255 0) 0.5]]]
```

# Attach Attributes

## Command

`attach attrib $ attrib : $ label : }` (10)

`attach attrib $ attrib : $ label : )` (11)

## Parametres

- attrib : the name of the object instantiated from the class `attrib`
- label : the name of the object instantiated from one of the top classes

## Examples

```
attach attrib red point-0
attach attrib (red large) point-1
attach attrib blue {point-2 rect-0}
attach attrib (5px black) {point-3 circ-0}
attach attrib (red thick) (point-4 line-0 triangle-0)
```

# Assign an Operation Name

## Command

`assign opname S action S class N repeat [=  $\infty$ ]` (12)

## Parametres

- action : the name of the corresponding action of class
- class : the name of one of the classes
- repeat : the amount of the commands emitting operation names

## Examples

```
assign instr create point 2
x-axis (1 0 0); y-axis (0 1 0)
// Back To Normal
```