Open Command-oriented Geometric Graphics Generator

OpenCG[3] Specification Version 0.2.12

Dong Nai-Jia [1]   Lin Yong-Hsiang [2]

[1] National Chiao Tung University
Department of Computer Science

[2] National Taiwan University
Department of Agricultural Chemistry

August 19, 2017

Command Tokens

Regular Expressions

$$\mathbb{N} := \{\ \alpha\ |\ \alpha \in \text{[0-9]+}\ \}$$
$$\mathbb{R} := \{\ \alpha\ |\ \alpha \in \text{[+\textbackslash-]?([0-9]*[.])?[0-9]+}\ \} \qquad\qquad \Rightarrow \mathbb{R} \supset \mathbb{N}$$
$$\mathbb{S} := \{\ \alpha\ |\ \alpha \in \text{'(.*?)'|[.0-9A-Za-z+\textbackslash-]+}\ \} \qquad\qquad \Rightarrow \mathbb{S} \supset \mathbb{R}$$
$$\mathbb{W} := \{\ \alpha\ |\ \alpha \in \text{[\textbackslash t]}\ \} \qquad\qquad\qquad\qquad\qquad\qquad \text{whitespace}$$

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

$\mathbf{C} \to \mathbf{A}\,\mathbf{C} \mid \; ; \; \mid \; \text{EOL}$

$\mathbf{A} \to \mathbf{T}(\mathbf{A}) \mid \mathbf{V}(\mathbf{A}) \mid \mathbf{S}(\mathbf{A}) \mid \mathbf{L}(\mathbf{A}) \mid \mathbf{L}(\mathbf{A}, \mathbf{A}, \cdots, \mathbf{A}) \mid \mathbb{N} \mid \mathbb{R} \mid \mathbb{S}$

$\mathbf{T}(\Pi) \equiv \Pi : n\,) \to (\;\Sigma(\Pi, n)\;)$
$\qquad\qquad\qquad\qquad\qquad \overbrace{}^{n \text{ items}}$

$\mathbf{V}(\Pi) \equiv \Pi : n\,\rangle \to \texttt{<}\;\Sigma(\Pi, n)\;\texttt{>} \quad \Big\| \quad \Sigma(\Pi, n) \to \overbrace{\Pi \;\cdots\; \Pi}^{n \text{ items}} \qquad \text{(identical)}$

$\mathbf{S}(\Pi) \equiv \Pi : n\,\} \to \{\;\Sigma(\Pi, n)\;\} \quad \Big\| \quad \mathbf{L}(\Pi) \equiv \mathbb{L}\,[\,\Pi : n\,] \to [\;\Sigma(\Pi, n)\;]$

$\mathbf{L}(\Pi_1, \Pi_2, \cdots, \Pi_{n-1}, \Pi_n) \equiv \mathbb{L}\,[\,\Pi_1\,\Pi_2 \cdots \Pi_{n-1}\,\Pi_n\,] \to [\;\Pi_1 \cdots \Pi_n\;]$

Descriptions

- Each command starts from $\mathbf{C}$ and ends with a ; or an EOL.
- Non-terminal symbol expansions are prior than function expansions.

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.

Fields, Classes, Objects and References

Definitions

- The whole system are divided into four fields and several classes:
  1. field e-(nviron.):   includes class window and class camera.
  2. field p-(rimitive):   includes class point, class circle, etc.
  3. field c-(ompound): includes class line, class triangle, class polygon, etc.
  4. field a-(uxiliary):   includes class attrib and class group.

Notations

- $class^x$ denotes the name of a class in the field x.

- $label^x$ denotes the unique name of the object from a class in the field x.

Prototypes

- Argument prototypes are written in a mixture of types and names with underlines.

- Each type with an asterisk indicates that the brackets are used for cross-referencing.

- Cross-reference is a feature for manipulating multitple objects in a single command.

- Each name with a plus/minus/ampersand implies that the given name is used for creating new objects/deleting existed objects/cross-referencing among objects, etc.
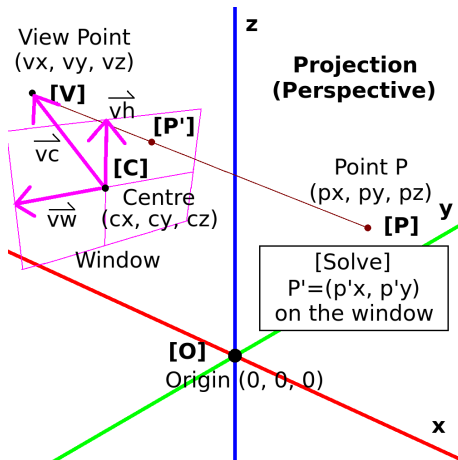
Figure: Projection in Euclidean $\mathbb{R}^3$ Space

Create a Window

Command

```
create window  𝕊  + labelᵉ                                              (1)
```

Parameters

- <u>label</u>ᵉ   : the name of the object from the class window

Examples

```
create window main
```

Delete a Window

Command

```
delete window  𝕊 – label^e   𝕊 string
```
(2)

Parameters

- <u>label</u>[e] : the name of the object from the class window
- <u>string</u> : the text printed right after exiting the current session

Examples

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

Create a Camera

Command

```
create camera  𝕊 +labelᵉ  ℝ : 3⟩ center  ℝ : 3⟩ : 2⟩ plane  ℝ : 3⟩ sight        (3)
```

Parameters

- $label^e$: the name of the object from the class camera
- center   : the world coordinate (c_x, c_y, c_z) of the center of the viewport
- plane   : the horizontal and the vertical vertors $(\vec{v_w}, \vec{v_h})$ of the viewport
- sight   : the reverse line of sight $\vec{v_c}$ from center to the camera

Examples

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

Select a Camera

Command

`select camera` $\underline{\mathbb{S}}$ $\underline{\text{\& label}_1^e}$ $\underline{\mathbb{S}}$ $\underline{\text{\& label}_2^e}$ (4)

Parameters

- $\underline{\text{label}_1^e}$: the name of the object from the class camera
- $\underline{\text{label}_2^e}$: the name of the object from the class window

Examples

```
select camera z-top main
```

Create Points

Command

```
create point   S + labelᵖ : }*        R : 3) coord              (5)
create point   S + labelᵖ : ⩾ n)*     R : 3) coord : n)*        (6)
```

Parameters

- $\underline{label^p}$: the name of the object from the class point
- $\underline{coord}$: the world coordinate (p_x, p_y, p_z) of the object named $\underline{label^p}$

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    S – label^p : }*
```
(7)

Parameters

- label^p : the name of the object from the class point

Examples

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

Create Attributes

Command

```
create attrib  S + labelᵃ : }⋆  𝕃 [ 𝕃 [ S classᵖᶜ  𝕃 [ S prop S value : ] ] : ]⋆   (8)
create attrib  S + labelᵃ : )⋆  𝕃 [ 𝕃 [ S classᵖᶜ  𝕃 [ S prop S value : ] ] : ]⋆   (9)
```

Parameters

- $\underline{label^a}$: the name of the object from the class attrib
- $\underline{class^{pc}}$: the name of a class in the field primitive or compound
- $\underline{prop}$: the property of the object from $\underline{class^{pc}}$
- $\underline{value}$: the value of $\underline{prop}$ in the designated format

Examples

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

Attach Attributes

Command

```
attach attrib  S & labelᵃ : )*  S & labelᵖᶜ : }*                    (10)
attach attrib  S & labelᵃ : )*  S & labelᵖᶜ : )*                    (11)
```

Parameters

- <u>labelᵃ</u>   : the name of the object from the class attrib
- <u>labelᵖᶜ</u> : the name of the object from a class in the field primitive or compound

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 Operations

Command

```
assign operat  S action  S class  N repeat [= ∞]                    (12)
```

Parameters

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

Examples

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