Enigma Level API II
Syntax Sheet

On syntax descriptions of datatype operators or methods we need to list allowed argument types. Often several types are possible and you are allowed to choose any of a list. In these cases we enlist the types enclosed by `<` and `>` and separated by `|`. These characters are not part of the operator or method itself and should thus not be typed into the level code. Note that we keep square braces `[`, `]` and curly braces `{`, `}` as literal Lua symbols. When these braces appear in the syntax you need to type them in the code.

Types

position A position within the world that can be described by an x and y coordinate.

positions The singleton type of the repository of all named positions.

object An Enigma object like a stone, item, floor, other. Any object is a position, too.

group A list of objects.

namedobjects The singleton type of the repository of all named objects.

default The singleton type of default values that can be used instead of Luas nil in anonymous table tile definitions.

tile A description of one or several objects for a common grid position (floor, item, stone, actor)

tiles The singleton type of the repository of all tile instances.

world The singleton type of the world that contains all objects.

position list A list of positions.

Position

Addition/Subtraction

```
result = pos <+|-> <pos | obj | cpos | polist>
result = <pos | obj | cpos | polist> <+|-> pos
```

Multiplication/Division

```
result = pos <*|/> number
result = number * pos
```

Sign

```
result = -pos
```

Center

```
result = #pos
```

Comparison

```
result = pos1 <==|~=> pos2
```

Concatenation

```
result = pos1 .. <pos2 | polist>
result = <pos1 | polist> .. pos2
```

Coordinate Access

```
result = pos["x"]
result = pos["y"]
result1, result2 = pos:xy()
```

Grid Rounding

```
result = pos:grid()
```

Existence

```
result = pos:exists()
```

Object

Attribute Access

```
result = obj["attributename"]
obj["attributename"] = value
obj:set({attributename1=value1,
         attributename2=value2, ...})
```

Messaging

```
result = obj:message("msg", value)
result = obj:msg(value)
```

Comparison

```
result = obj1 <==|~=> obj2
```

Existence

```
result = -obj
result = obj:exists()
```

Kill

```
obj:kill()
```

Kind Checks

```
result = obj:is("kind")
result = obj:kind()
```

Coordinate Access

```
result = obj["x"]
result = obj["y"]
result1, result2 = obj:xy()
```

Addition/Subtraction

```
result = obj <+|-> <pos | obj | cpos | polist>
result = <pos | obj | cpos | polist> <+|-> obj
```

Center

```
result = #obj
```

Join

```
result = obj + group
result = group + obj
```

Intersection

```
result = obj * group
result = group * obj
```

Difference

```
result = obj - group
result = group - obj
```

Sound

```
result = obj:sound("name", volume)
```

Group

Messaging

```
result = group:message("msg", value)
result = group:msg(value)
```

Attribute Write

```
group["attributename"] = value
group:set({attributename1=value1,
           attributename2=value2, ...})
```

Comparison

```
result = group1 <==|~=> group2
```

Length/Size

```
result = #group
```

Member Access

```
result = group[index]
result = group[obj]
```

Loop

```
for obj in group do ... end
```

Join

```
result = group + <obj | group>
result = <obj | group> + group
```

Intersection

```
result = <obj | group> * group
result = group * <obj | group>
```

Difference

```
result = <obj | group> - group
result = group - <obj | group>
```

Shuffle

```
result = group:shuffle()
```

Sorting
```
result = group:sort("circular")
result = group:sort("linear" <, direction>)
result = group:sort()
```

Subset
```
result = group:sub(number)
result = group:sub(start, end)
result = group:sub(start, -number)
```

Nearest Object
```
result = group:nearest(obj)
```

NamedObjects
Repository Request
```
result = no["name"]
```

Object Naming
```
no["name"] = obj
```

PositionList
Comparison
```
result = polist1 <==|~=> polist2
```

Length
```
result = #polist
```

Member Access
```
result = group[index]
```

Concatenation
```
result = polist1 .. <pos | polist2>
result = <pos | polist1> .. polist2
```

Translation
```
result = polist <+|-> <pos | obj | cpos>
result = <pos | obj | cpos> <+|-> polist
```

Stretching
```
result = polist * number
result = number * polist
```

Positions Repository
Repository Request
```
result = po["name"]
```

Repository Storage
```
po["name"] = obj
```

Position Convertion
```
result = po(<obj | pos | {x, y} | x,y>)
```

PositionList Convertion
```
result = po(group | {pos1, pos2, pos3 })
```

Tile and Object Declaration
Tile concat
```
result = tile .. <tile | odecl>
result = <tile | odecl> .. tile
```

Tiles Repository
Tiles Storage
```
ti["key"] = <tile | odecl>
```

Tiles Request
```
result = ti["key"]
```

Tile Convertion
```
result = ti(odecl)
```

World
World Creation
```
width, height = wo(topresolver, defaultkey, map)
width, height = wo(topresolver, libmap)
width, height = wo(topresolver, defaultkey, width, height)
```

World Tile Set
```
wo[<object | position | table |
    group | polist>] = tile_declarations
```

Global Attribute Set
```
wo["attritbutename"] = value
```

Global Attribute Get
```
var = wo["attritbutename"]
```

add
```
wo:add(tile_declarations)
wo:add(target, tile_declarations)
```

drawBorder
```
wo:drawBorder(upperleft_edge, lowerright_edge,
              <tile | key, resolver>)
wo:drawBorder(upperleft_edge, width, height,
              <tile | key, resolver>)
```

drawMap
```
wo:drawMap(resolver, anchor, ignore, map, [readdir])
wo:drawMap(resolver, anchor, libmap_map, [readdir])
```

drawRect
```
wo:drawRect(upperleft_edge, lowerright_edge,
            <tile | key, resolver>)
wo:drawRect(upperleft_edge, width, height,
            <tile | key, resolver>)
```

world floor
```
result = wo:fl(<pos | {x, y} | x,y | obj | group | polist>)
```

world item
```
result = wo:it(<pos | {x, y} | x,y | obj | group | polist>)
```

shuffleOxyd
```
wo:shuffleOxyd(rules)
```

world stone
```
result = wo:st(<pos| {x, y} | x,y | obj | group | polist>)
```

Functions
cond
```
cond(condition, iftrue, iffalse)
```

fl
```
result = fl(<pos | {x, y} | x,y | obj | group | polist>)
```

grp
```
grp(<{obj1,obj2, ...} | obj1,obj2, ... | group>)
```

it
```
result = it(<pos | {x, y} | x,y | obj | group | polist>)
```

ORI2DIR
```
result = ORI2DIR[orientation]
```

random
```
result = random(<| n | 1,u>)
```

st
```
result = st(<pos | {x, y} | x,y | obj | group | polist>)
```