

Lua Tiny 3D

Introduction

This is a 3D engine which you can use to render 3D objects in Lua.
There are 4 functions that you are supposed to use :

```
rotate(vertices, angles, x, y, z)
translate(vertices, x, y, z)
scale(vertices, x, y, z)
render(gc, vertices, faces, mode, color, offset)
```

3D objects

3D objects are represented by their vertices, and their faces.

A vertex is a table of coordinates (x,y,z).

```
{0, 0, 0}
```

All the object's vertices are stored in a table.

```
{{0, 0, 0}, {0, 0, 1}, {0, 1, 0}}
```

A face is a table of vertex indices.

```
{1, 2, 3} --This is a triangle defined with 3 vertices : the first, the second and the third one
```

```
{1, 2, 3, 4, 5, 6, 7} --A face can be defined with any number of vertices
```

All the object's faces are stored in a table.

```
{{1, 2, 3}, {1, 2, 3, 4, 5, 6, 7}, {4, 2, 7, 8}}
```

Rendering mode

There are 6 rendering modes.

- 1 : Only vertices
- 2 : Vertices and edges
- 3 : Only edges
- 4 : Edges and faces, without lighting
- 5 : Edges and faces, with lighting
- 6 : Faces only, with lighting

Render

The function used to render an object is `render(gc, vertices, faces, mode, color, offset)`. It should be called in `on.paint(gc)`

-gc is the graphic context.

-vertices is the object's vertices.

-faces is the object faces.

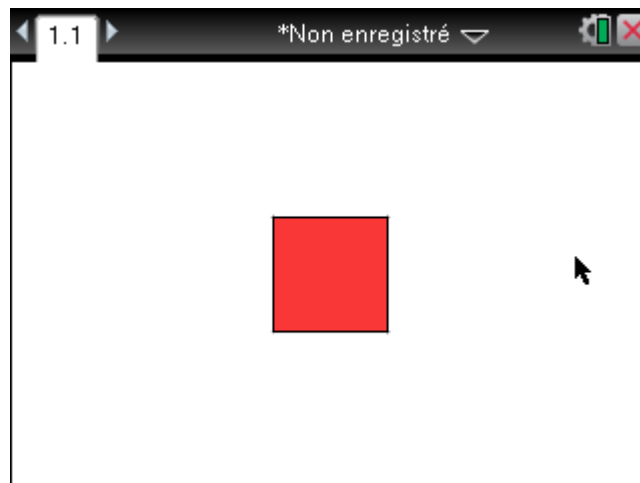
-mode is the rendering mode.

-color is the object's color. It is a table (R,G,B) e.g. : {150,150,100}.

-offset is the eye distance. Use 8 for optimal results.

```
cubeVertices={{-1,-1,-1},{1,-1,-1},{-1,1,-1},{-1,-1,1},{1,1,-1},{-1,1,1},{1,-1,1},{1,1,1}}
cubeFaces={{1,2,5,3},{1,4,6,3},{1,4,7,2},{8,7,4,6},{8,5,3,6},{8,5,2,7}}

function on.paint(gc)
  render(gc, cubeVertices, cubeFaces, 5, {200,0,0}, 8)
end
```



It looks like a 2D square... It would be more beautiful if it were rotated. Let's use transformations !

Transformations

There are 3 transformations :

`translate(vertices, x, y, z)`

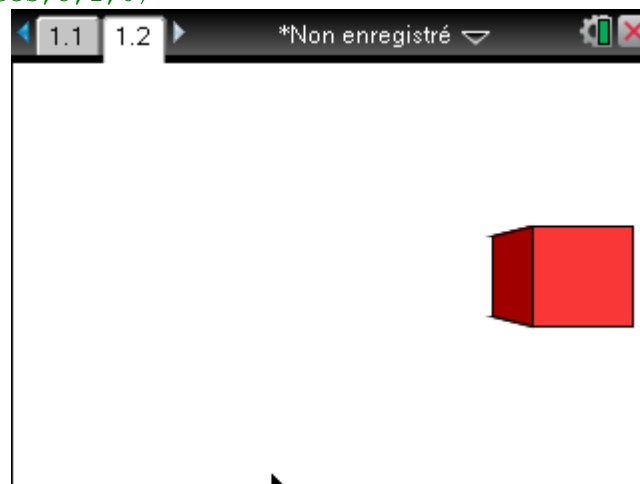
`scale(vertices, x, y, z)`

`rotate(vertices, angles, x, y, z)`

Translate

You can move vertices with this function.

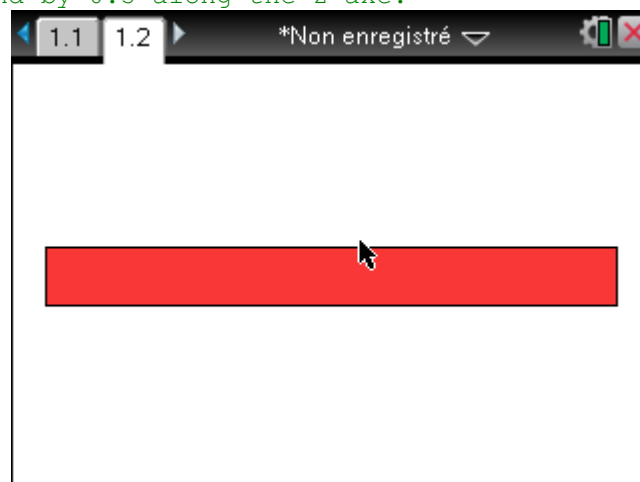
`translate(cubeVertices, 5, 1, 0)`



Scale

Increases or decreases the object size.

`scale(cubeVertices, 5, 1, 0.5)` --The size is multiplied by 5 along the x axe, by 1 along the y axe and by 0.5 along the z axe.

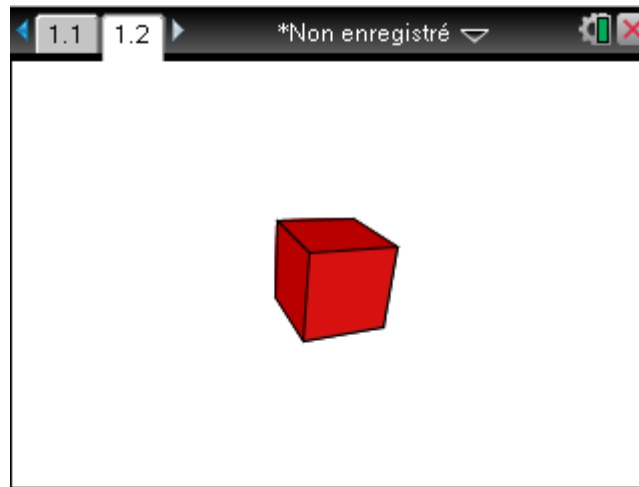


Rotate

You can rotate vertices.

`rotate(cubeVertices,3.14/4,1,0,1)` -A rotation of $\pi/4$ (rad) is applied around x axis and z axis.

NOTE : There is a bug with the rotate function. If you use the rotation around several axis, the object shrinks a little bit. So you should use `rotate()` several times



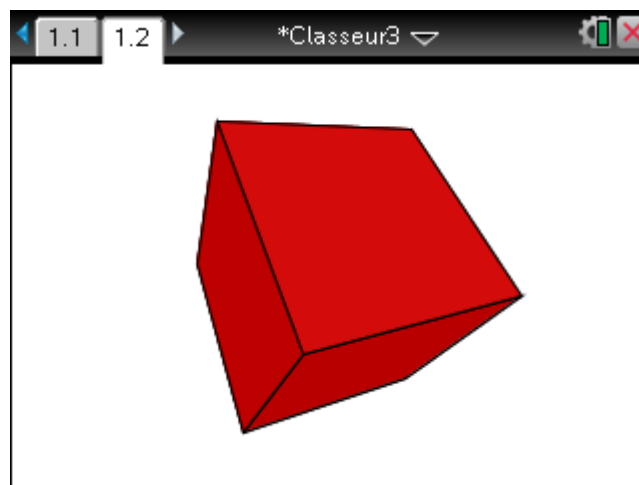
A nice rotating cube

```
cubeVertices={{-1,-1,-1},{1,-1,-1},{-1,1,-1},{-1,-1,1},{1,1,-1},{-1,1,1},{1,-1,1},{1,1,1}}
cubeFaces={{1,2,5,3},{1,4,6,3},{1,4,7,2},{8,7,4,6},{8,5,3,6},{8,5,2,7}}

scale(cubeVertices,2,2,2)
rotate(cubeVertices,3.14/8,1,0,0)

function on.paint(gc)
  render(gc,cubeVertices,cubeFaces,5,{200,0,0},8)
  timer.start(0.01)
end

function on.timer()
  timer.stop()
  rotate(cubeVertices,0.01,0,0,1)
  platform.window:invalidate()
end
```



Have fun !