
Tiny*3dengineDocumentation*

Team COOP

Feb 22, 2022

Contents

1	Tiny 3D Engine	1
1.1	Loading a model	1
1.2	The SCENE and the ENGINE	2
1.3	What if I already have my vertices and polygons?	2
1.4	Command line	3
1.5	Performances	3
1.6	Requirements	3
1.7	Origins	4
2	tiny_3d_engine package	5
2.1	Submodules	5
2.2	tiny_3d_engine.cli module	5
2.3	tiny_3d_engine.color module	5
2.4	tiny_3d_engine.engine module	5
2.5	tiny_3d_engine.geo_loader module	5
2.6	tiny_3d_engine.part3d module	5
2.7	tiny_3d_engine.ply_loader module	5
2.8	tiny_3d_engine.scene3d module	5
2.9	tiny_3d_engine.screen module	5
2.10	tiny_3d_engine.viewfield module	5
3	Indices and tables	7

logo

This package is a python 3D engine based on the Tkinter Canvas. It uses Numpy for math handling.

It is available on [pipy](#), documentation is on [readthedocs](#), sources are mirrored on [gitlab.com](#)

Install this using

```
pip install tiny_3d_engine
```

It create simle 3D rendering for bars, tri and quad elements and store the scenes in Ensight's ASCII .geoformat. A trivial grid would look like:

trivial_example

1.1 Loading a model

In this simple example, a .geo file is loaded `load_file_as_scene` into a 3D scene. This scene is given to a new engine object `Engine3D`. We apply a rotation `.rotate()` before rendering `.render()` the scene on the screen, then leave the interaction to the user `.mainloop()`.

```
scene = load_file_as_scene("myfile.geo")
test = Engine3D(scene)
test.rotate('y', 45)
test.render()
test.mainloop()
```

1.2 The SCENE and the ENGINE

The **SCENE** is an object storing the 3D model. A void scene is simply `None`. You can update a scene with the method `.update()`. Each scene handle several parts identified by a name, a string looking either as `-tag-` (e.g. "ceiling") or as `-family-.-tag-` (e.g. "house.ceiling").

The **ENGINE** is used to project the scene on the 2D screen. Once started, the view point can be controlled by methods such as `.translate()` or `.rotate()`, then refreshed with `.render()`. The scene can be updated with `.update()`. If you want user interaction with the result, finish with the typical TK `.mainloop()`.

1.3 What if I already have my vertices and polygons?

In the following example, two squares are appended to an initially void **Scene3D** object, using the method `scene.add_or_update_part`.

- The first, in blue, is made of edges (2 vertices connectivity)
- The second, in red, is made of squares (4 vertices connectivity)

This scene is passed to the **Engine3D** object, triggering a window.

```
from tiny_3d_engine import (Scene3D, Engine3D)

scene = Scene3D()

SIZE = 2
LENGTH= 200.
points = list()
conn = list()
dx = LENGTH/

for i in range(SIZE):
    for j in range(SIZE):
        index = len(points)
        points.append([i*dx, j*dx, 0])
        points.append([(i+1)*dx, j*dx, 0])
        points.append([i*dx, (j+1)*dx, 0])
        points.append([(i+1)*dx, (j+1)*dx, 0])
        conn.append([index, index+1])
        conn.append([index+3, index+1])

scene.update("square1", points, conn, color="#0000ff")

points = list()
conn = list()
for i in range(SIZE):
    for j in range(SIZE):
        index = len(points)
        points.append([i*dx, j*dx, LENGTH])
        points.append([(i+1)*dx, j*dx, LENGTH])
        points.append([i*dx, (j+1)*dx, LENGTH])
        points.append([(i+1)*dx, (j+1)*dx, LENGTH])
        conn.append([index, index+1, index+3, index+2])
scene.update("square2", points, conn, color="#ff0000")

test = Engine3D(scene)
```

(continues on next page)

(continued from previous page)

```
test.rotate("x", 45)
test.rotate("y", 45)
test.render()
test.mainloop()
```

(It would have been easier in numpy, but I wanted to keep this readable for non-numpy programmers)

1.4 Command line

A small command line interface is available:

```
Usage: tiny_3d_engine [OPTIONS] COMMAND [ARGS]...

----- TINY_3D_ENGINE -----

You are now using the Command line interface of Tiny 3D engine, a Python3
Tkinter lightweight 3D engine, created at CERFACS (https://cerfacs.fr).

This package is likely as a dependency of other packages, to
provide a light 3D feedback for small 3D scenes <100 000 polygons. This
CLI is given here for developers perusal and demonstrations. Find the
script of these small tools in the /examples folder of the package.

This is a python package currently installed in your python environment.
See the full documentation at :
https://tiny-3d-engine.readthedocs.io/en/latest/.

DISCLAIMER: Tiny 3D engine is a brute force flat renderer. As it is NOT
using your graphical card, do not expect anything fancier than a 1980
video game.

Options:
  --help  Show this message and exit.

Commands:
  bench  Run a short benchmark on your machine.
  load   Load a 3D scene from FILENAME.
  rabbit Run a demo with the Stanford Rabbit.
```

1.5 Performances

Do not expect more than an early 90s videogame. During mouse interactions, the frames per second is roughly 30 000 / nb. of polygons (i.e. 15 fps for 2000 polygons).

The engine is by default limited 100 000 polygons in a static view and 2 000 during mouse interactions. If the model goes beyond these limits, the engine randomly remove polygons at the loading time, to keep the window responsive.

1.6 Requirements

The present library require Numpy and Tkinter. The Tk aspects are limited to the **screen** object. In the future I might write extensions for PyQt4 Canvas or Matplotlib... or not.

1.7 Origins

This work stems from a mix between a Pure Tcl/Tk Engine of mine [pure TK 3d engine](#) and the the [pyEngine3D-master](#) of [henry Haefliger pyEngine3D](#) , because I really liked the API.

The present one allow several parts to be loaded, and uses numpy. Scenes can be dumped or read from the Ensignt `.case/.geo` files.

2.1 Submodules

2.2 tiny_3d_engine.cli module

2.3 tiny_3d_engine.color module

2.4 tiny_3d_engine.engine module

2.5 tiny_3d_engine.geo_loader module

2.6 tiny_3d_engine.part3d module

2.7 tiny_3d_engine.ply_loader module

2.8 tiny_3d_engine.scene3d module

2.9 tiny_3d_engine.screen module

2.10 tiny_3d_engine.viewfield module

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`