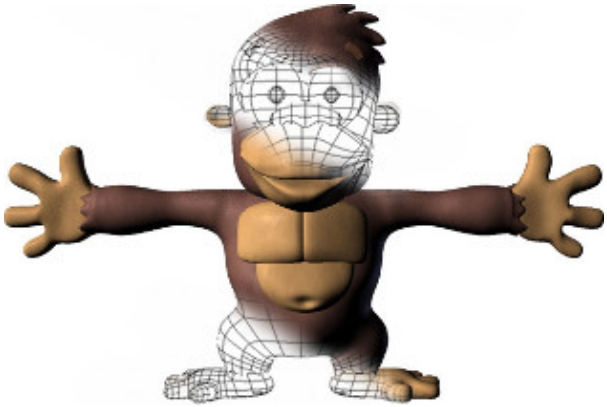


B2- C Graphical Programming

B-MUL-151

Raytracer 1

3D Layout Engine

v1.3.2

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Raytracer 1

3D Layout Engine

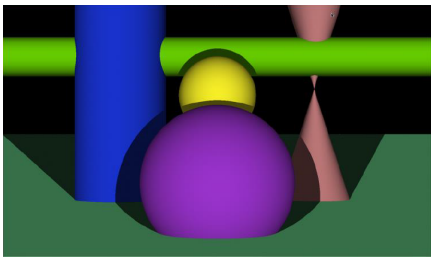
binary name:	raytracer1
repository name:	raytracer1
repository rights:	ramassage-tek
language:	C
group size:	1
compilation:	via Makefile, including re, clean and fclean rules

Mandatory Part

Raytracer 1 consists of programming the mandatory section of the next project, Raytracer 2. It requires completing a program that generates 3D images.

Your program must include the following features, in the following order:

- 3D objects' drawing:
  - Sphere,
  - Plane,
  - Cylinder,
  - Cone,
- the possibility of moving and turning the objects in every directions,
- the possiblity of moving and turning the camera in every directions,
- lighting effect from one point source,
- shadowing produced from light blocking.



Bonuses

Below is an **exhaustive** list of bonuses:

- scene described in a configuration file (format is up to you),
- handling multiple light sources,
- objects' brightness,
- plug-in system with the help of a dynamic library.

If you choose to render your scenes with configuration files, think about your format so that you can use it again in your Raytracer 2!

EPITECH. L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Autograder

In order to enable your work to be automatically graded, the implementation of the following functions is required:

- ***calc_dir_vector.c***

```
sfVector3f  calc_dir_vector(float dist_to_plane, sfVector2i screen_size, sfVector2i
    screen_pos);
```

dist_to_plane is the distance along the x-coordinate between the eye and the screen and with the eye looking at the center of the screen. The function returns the direction vector of the ray going from the eye toward the pixel at *screen_pos* on a *screen_size* screen.

- ***translate.c***, ***rotate.c***

```
sfVector3f  translate(sfVector3f to_translate, sfVector3f translations);
sfVector3f  rotate_xyz(sfVector3f to_rotate, sfVector3f angles);
sfVector3f  rotate_zyx(sfVector3f to_rotate, sfVector3f angles);
```

translate returns the new coordinates of a point at *to_translate* after a translation by the vector *translations*.
*rotate_** return the new coordinates of a point at *to_rotate* after a rotation by the given *angles* around the x, y and z axis. The angles are in degrees.
rotated_xyz and *rotated_zyx* respectively apply the rotations in the x-y-z and z-y-x order.

- ***sphere.c***, ***plane.c***, ***cylinder.c***, ***cone.c***

```
float       intersect_sphere(sfVector3f eye_pos, sfVector3f dir_vector, float radius);
float       intersect_plane(sfVector3f eye_pos, sfVector3f dir_vector);
float       intersect_cylinder(sfVector3f eye_pos, sfVector3f dir_vector, float radius);
float       intersect_cone(sfVector3f eye_pos, sfVector3f dir_vector, float semiangle);
sfVector3f  get_normal_sphere(sfVector3f intersection_point);
sfVector3f  get_normal_plane(int upward); // 'upward' is either 1 or 0
sfVector3f  get_normal_cylinder(sfVector3f intersection_point);
sfVector3f  get_normal_cone(sfVector3f intersection_point, float semiangle);
```

*intersect_** return the k so that we can calculate the distance between the eye at *eye_pos* and the nearest point on the object regarding the direction vector *dir_vector*. The function returns *-1.0f* if there is no frontal intersection point.
*get_normal_** return the normal vector on the object using the given parameters.
All the above functions reckon on the objects to be at $(0, 0, 0)$.
semiangle is given in degrees.

- ***light.c***

```
float       get_light_coef(sfVector3f light_vector, sfVector3f normal_vector);
```

The function returns the coefficient between 0 and 1 corresponding to how strong the light is on a point, using the *light_vector* between that point and the light source and the *normal_vector* of the object at the point.
0 means that the light has no effect on the point.

These functions must be located in an *src/* folder at the root of your repository.
If header files (.h) are included, make sure to place them in the root of your repository, either in an *include/* or an *inc* folder.

You can use an additional file named **utils.c** which can contain some functions called from your autograded functions.

EPITECH. L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Authorized Functions

- C Math library (-lm)
- C Libdl (-ldl)
- Pthread library (-lpthread)
- open
- close
- read
- write
- malloc
- free

CSFML functions:

- sfRenderWindow_isOpen
- sfRenderWindow_pollEvent
- sfRenderWindow_waitEvent
- sfRenderWindow_clear
- sfRenderWindow_drawSprite
- sfRenderWindow_display
- sfRenderWindow_create
- sfRenderWindow_destroy
- sfRenderWindow_close
- sfTexture_create
- sfTexture_updateFromPixels
- sfTexture_destroy
- sfSprite_create
- sfSprite_setTexture
- sfSprite_destroy
- all of System module's functions
- all of Window module's functions
- all of Audio module's functions

EPITECH. L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE