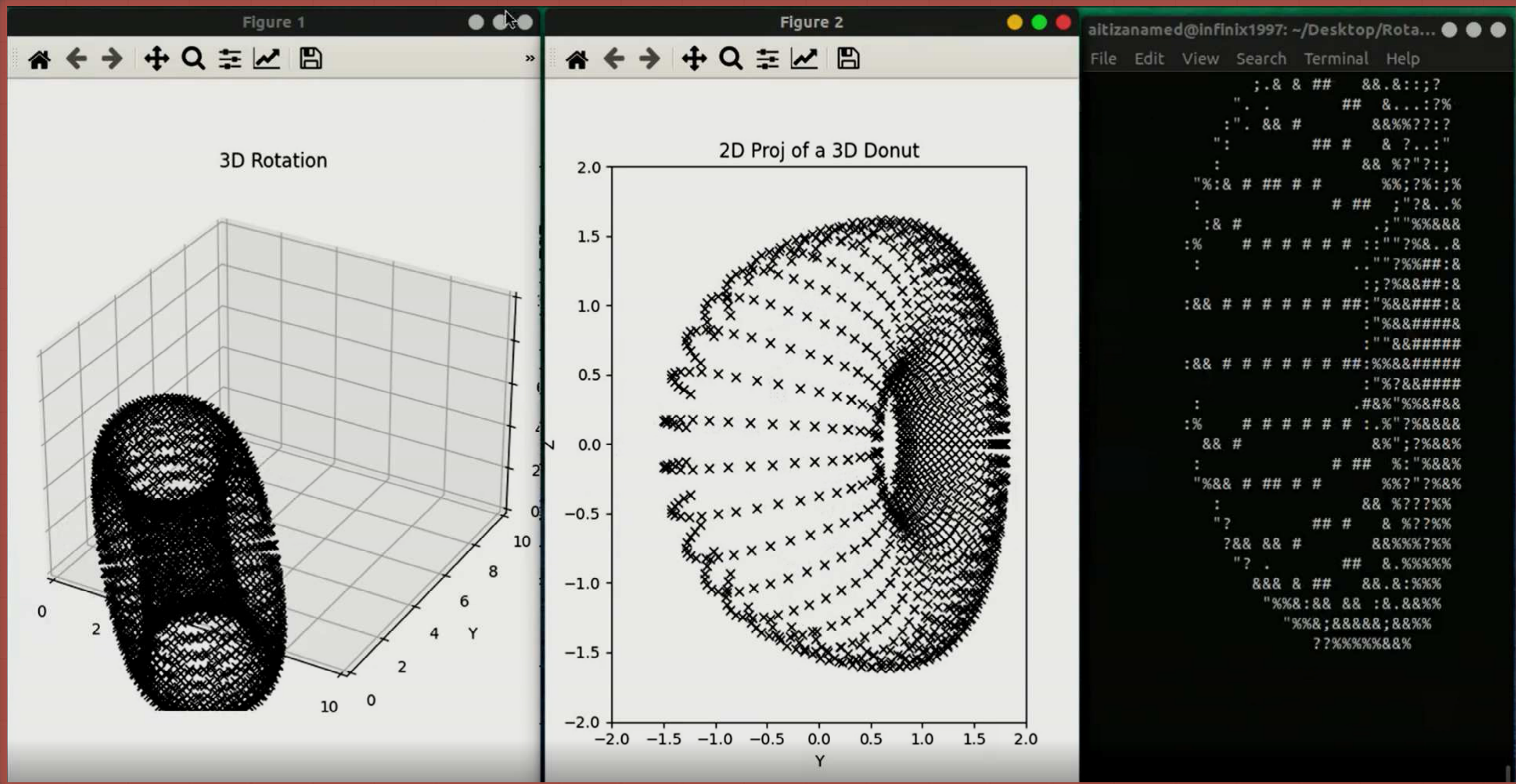
# 3D Donut Animation on terminal using python



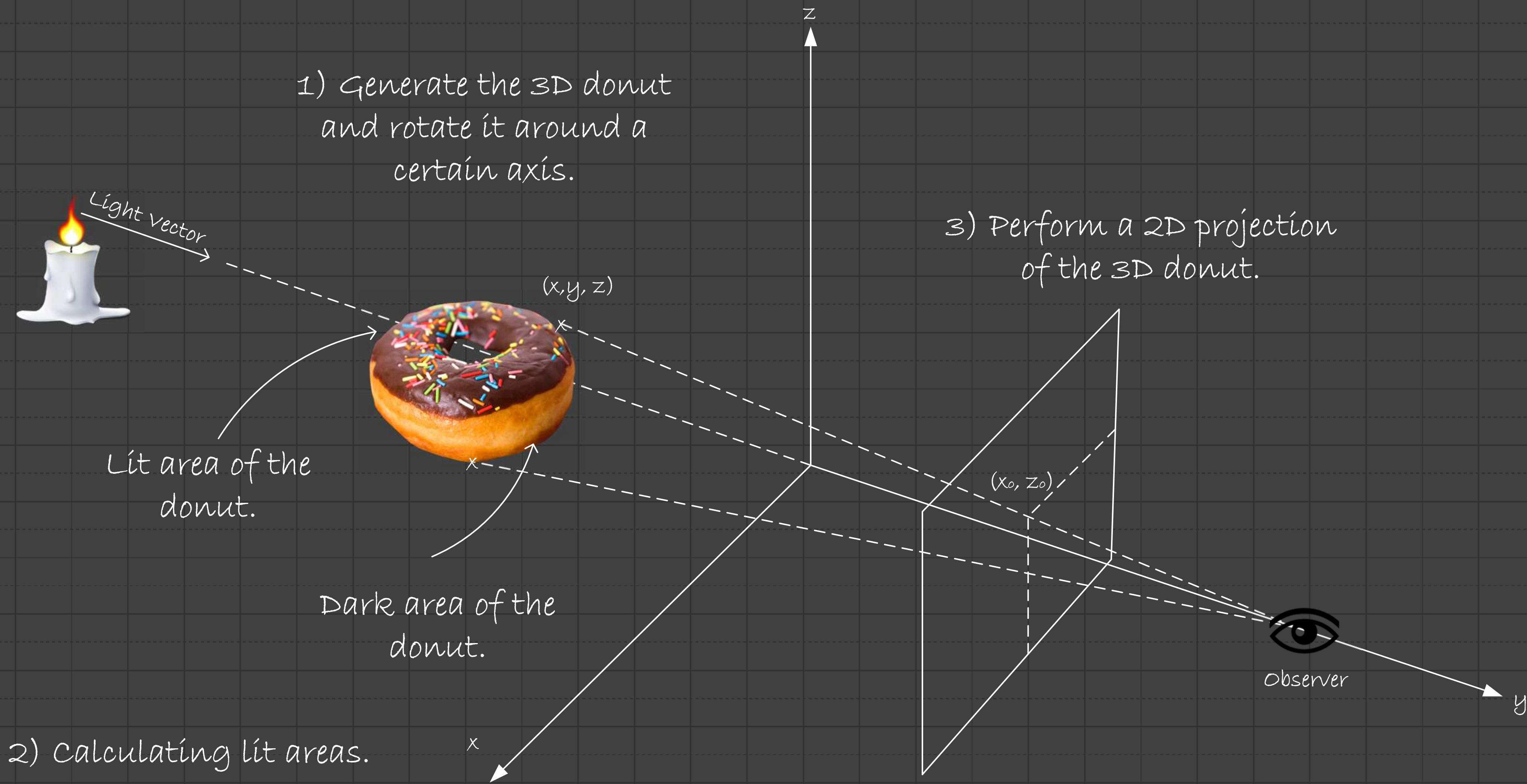

**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

# Plan :




**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

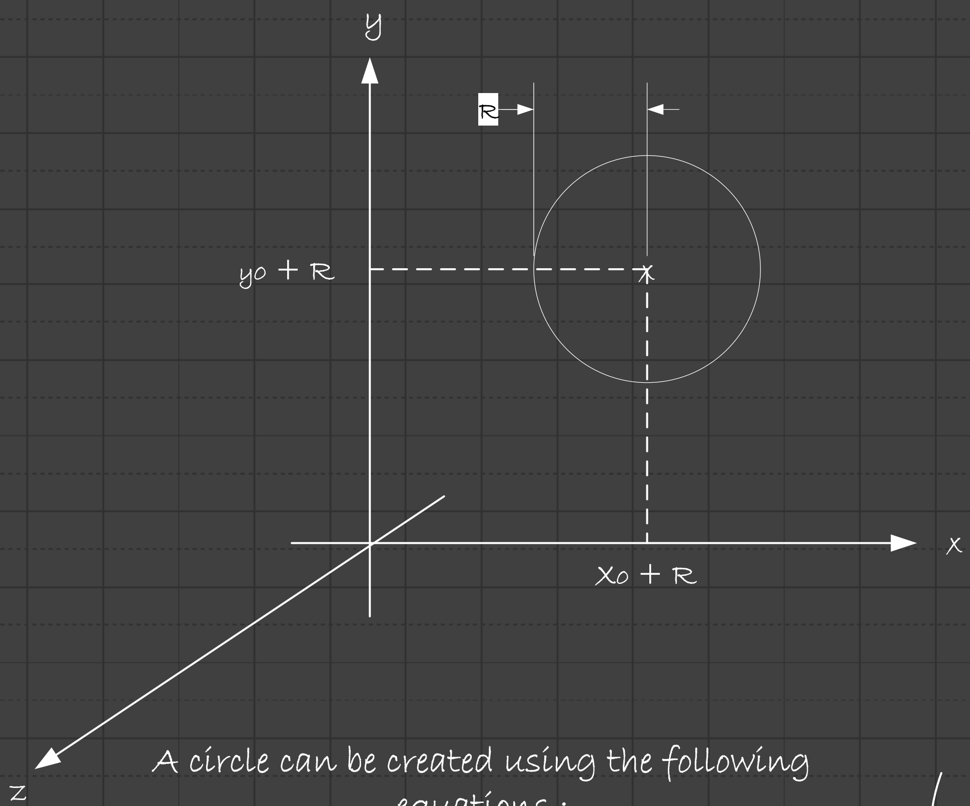
# General concept:



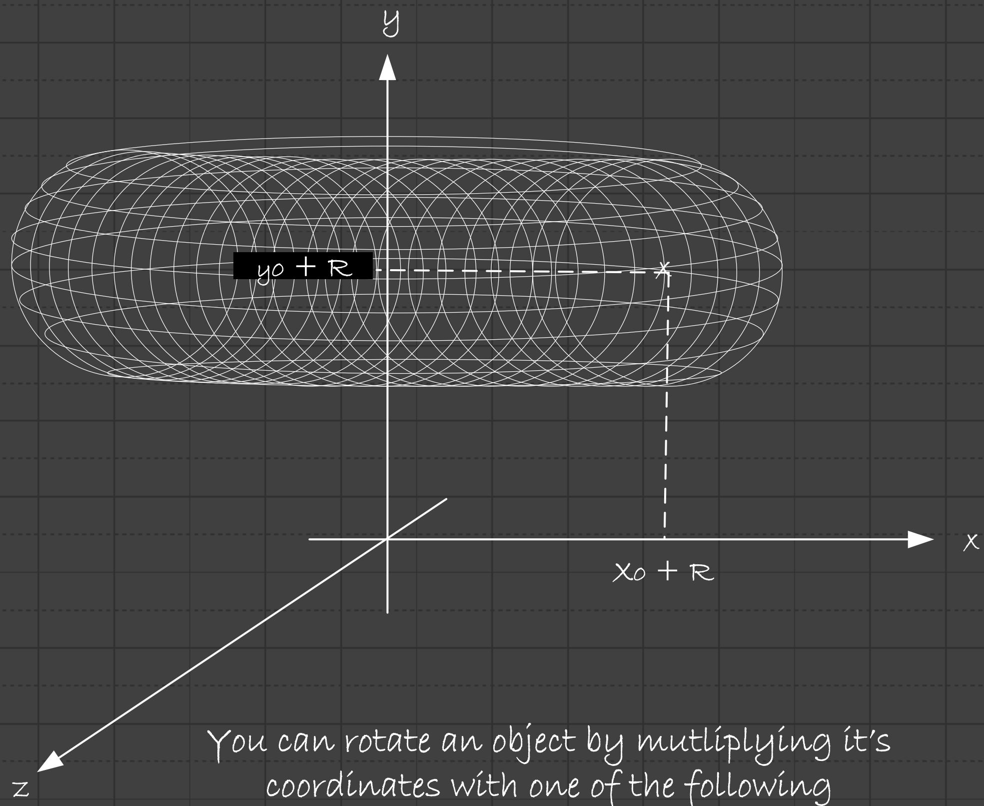

**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

# 3D Donut Generation Concept:

1) Create a circle, with a specific number of points.




2) Extrude the previous circle around an evolution axis by performing a rotation and saving the new coordinates.




A circle can be created using the following equations :

$$x = x_0 + R.\cos(\theta)$$

$$y = y_0 + R.\sin(\theta)$$

$$z = z_0$$

You can rotate an object by mutliplying it's coordinates with one of the following matrices :

$$Rx = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & \sin(\theta) & -\cos(\theta) \end{pmatrix} \quad Ry = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \quad Rz = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Example :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & \sin(\theta) & -\cos(\theta) \end{pmatrix} . \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

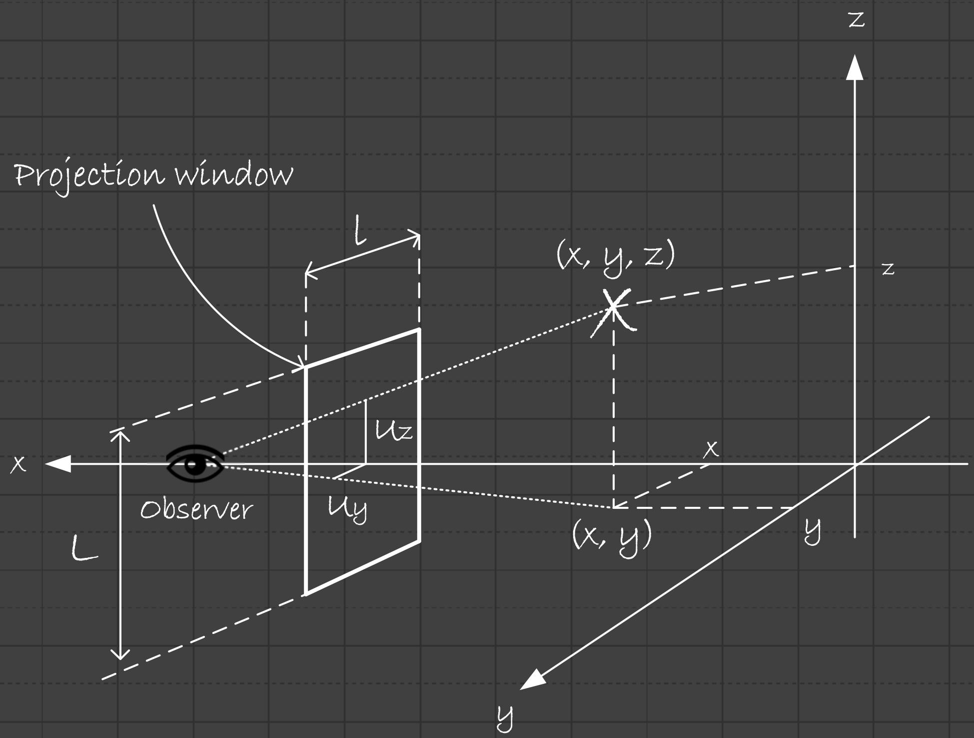


**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

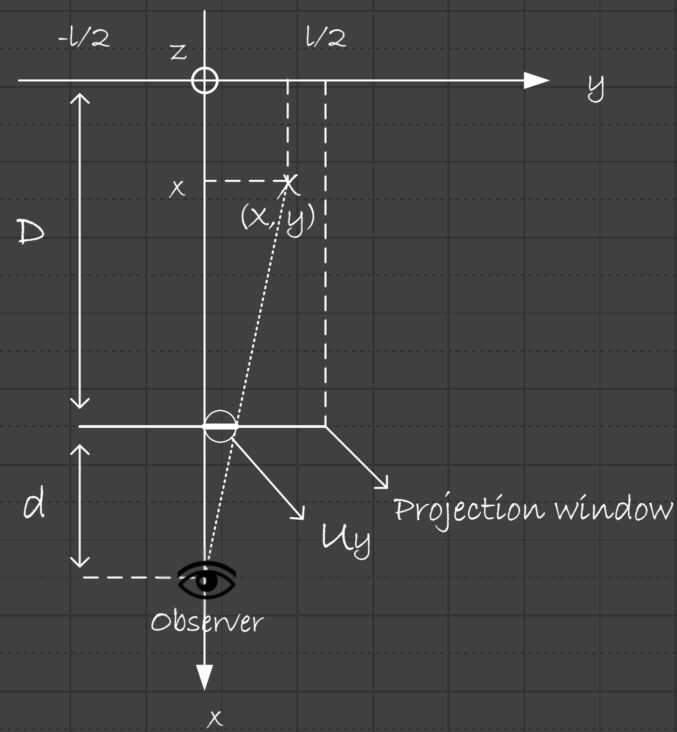
# 2D Projection Concept:

Assuming that the 2D projection is performed on the (y, z) plan and it is observed using an L by l window. Then, the projected coordinates Ud and Uz can be derived by applying thale's theorem following the views below:

Perspective view :




Upper view :




$$U_z = \sqrt{\frac{d^2 + U_y^2}{y^2 + (D - x + d)^2}}$$

$$U_y = \frac{d}{D - x} \cdot y$$

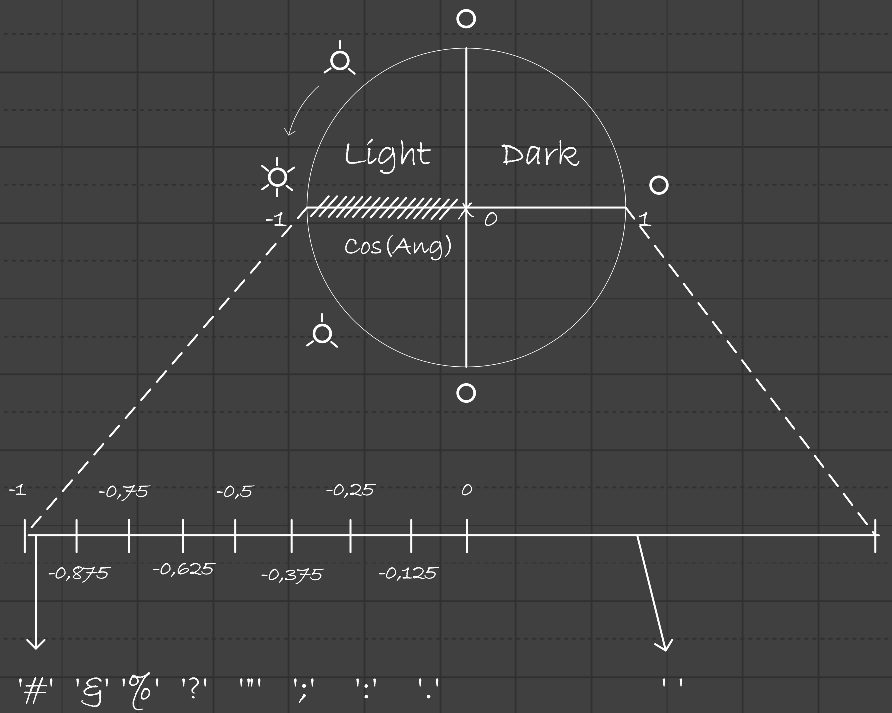


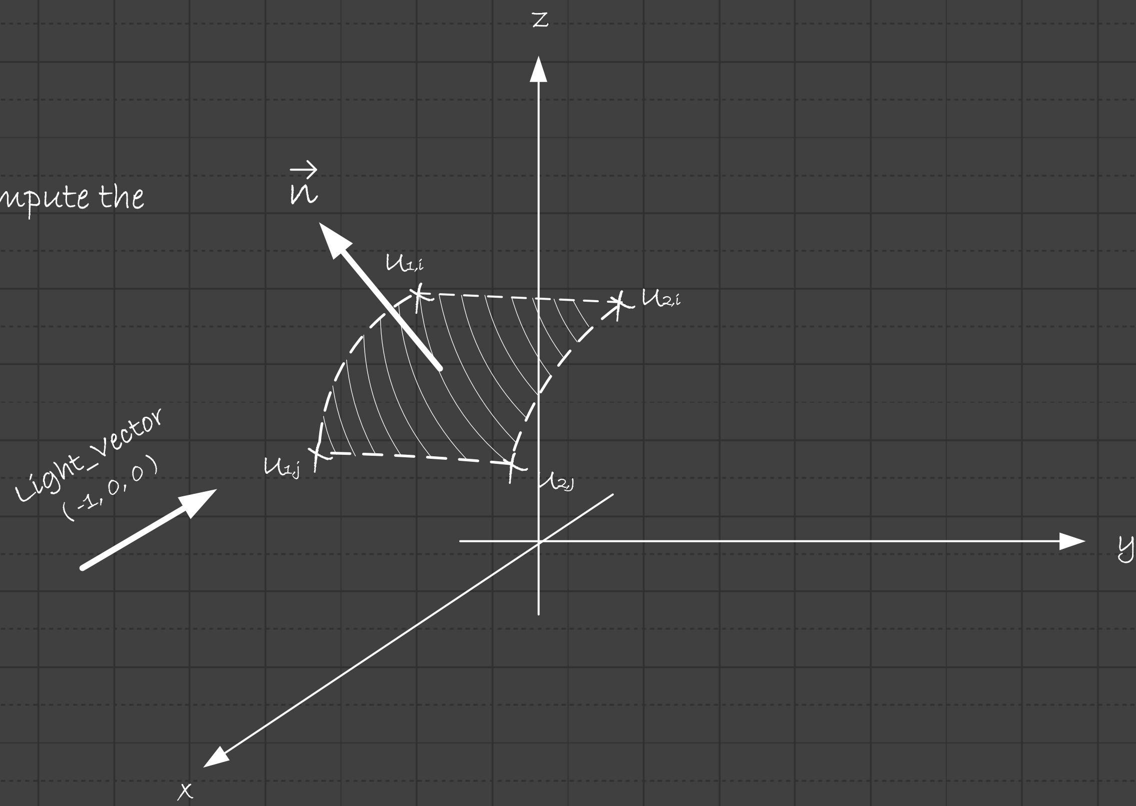
**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

# Lit Areas Calculation concept:

To tell whether a surface of the donut is lit or not, we can compute the scalar product of its normal vector and the light vector :

$$\vec{n} \cdot \vec{V_l} = \cos(\widehat{\vec{n}\ \vec{V_l}}) = \cos(Ang)$$




How it works ?

If, for instance, the computed cos(ang) is in the range [-1: -0,875], the a '#' will be printed according to the surface's projected coordinates. Except for cos(Ang) > 0 we abort plotting and printing it.

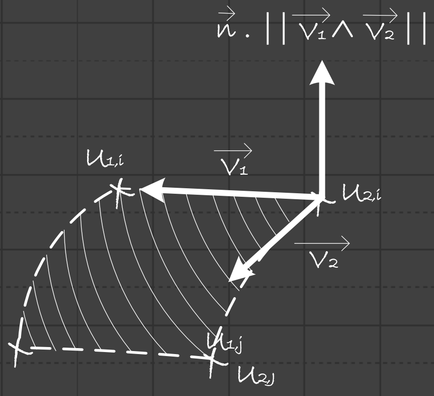
In the next slide, we will, learn how to compute $\vec{n}$



**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

# Surface normal vector, Calculation concept:

The surface normal vector is the cross product of the adjacent vectors, respectively : $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$



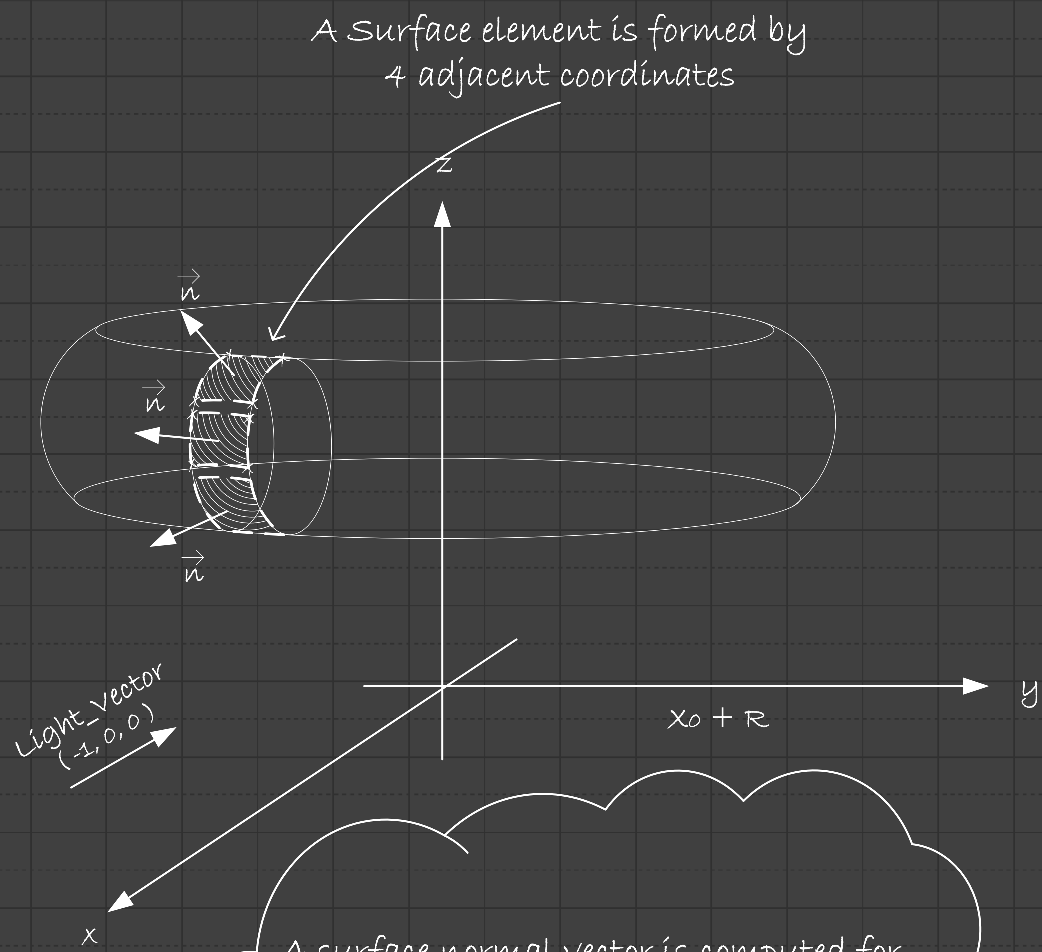

With :

$$\overrightarrow{V_1} = \begin{pmatrix} x_{1,i}-x_{2,i} \\ y_{1,i}-y_{2,i} \\ z_{1,i}-z_{2,i} \end{pmatrix} \quad \text{and} \quad \overrightarrow{V_2} = \begin{pmatrix} x_{2,j}-x_{2,i} \\ y_{2,j}-y_{2,i} \\ z_{2,j}-z_{2,i} \end{pmatrix}$$

$$\overrightarrow{V_1} \wedge \overrightarrow{V_2} = \begin{pmatrix} x_{1,i}-x_{2,i} \\ y_{1,i}-y_{2,i} \\ z_{1,i}-z_{2,i} \end{pmatrix} \wedge \begin{pmatrix} x_{2,j}-x_{2,i} \\ y_{2,j}-y_{2,i} \\ z_{2,j}-z_{2,i} \end{pmatrix}$$

$$= \begin{pmatrix} (y_{1,i}-y_{2,i})\cdot(z_{2,j}-z_{2,i}) - (z_{1,i}-z_{2,i})\cdot(y_{2,j}-y_{2,i}) \\ (z_{1,i}-z_{2,i})\cdot(x_{2,j}-x_{2,i}) - (z_{2,j}-z_{2,i})\cdot(x_{1,i}-x_{2,i}) \\ (x_{1,i}-x_{2,i})\cdot(y_{2,j}-y_{2,i}) - (x_{2,j}-x_{2,i})\cdot(y_{1,i}-y_{2,i}) \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$||\overrightarrow{V_1} \wedge \overrightarrow{V_2}|| = \sqrt{a^2 + b^2 + c^2} \qquad \text{Thus,} \quad \vec{n} = \frac{\overrightarrow{V_1} \wedge \overrightarrow{V_2}}{||\overrightarrow{V_1} \wedge \overrightarrow{V_2}||}$$



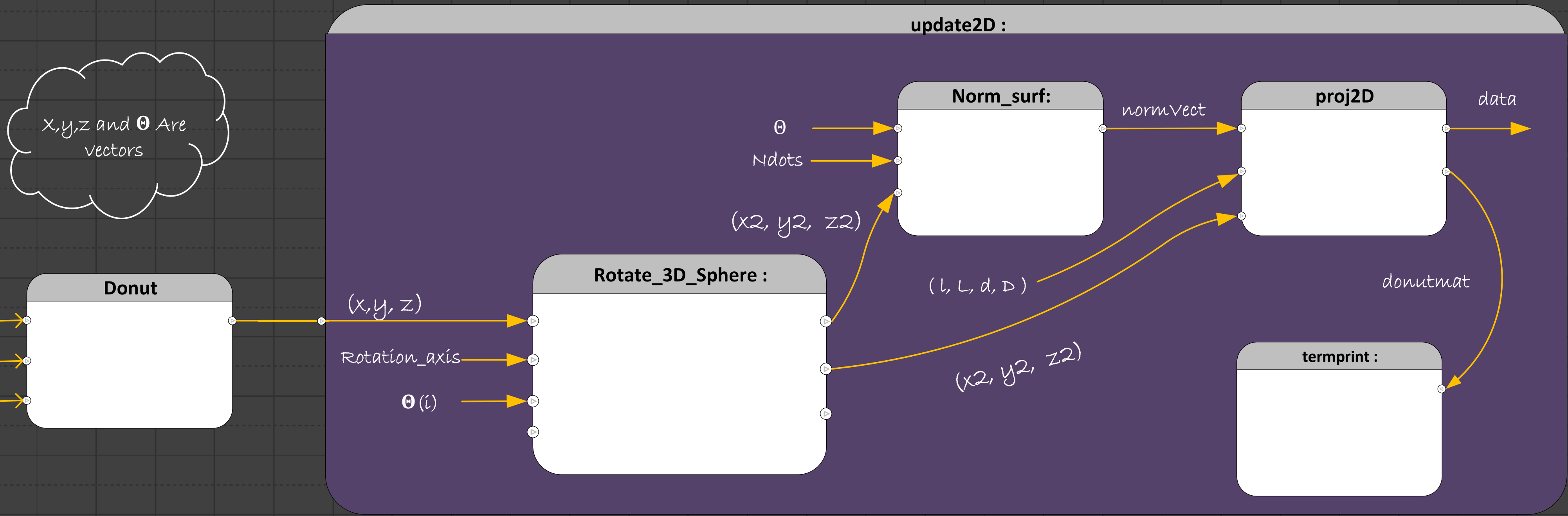

A surface normal vector is computed for every 4 adjacent coordinates of the donut. The donut is formed using *Ndots* circles, containing each : *Ndots* points.

Ndots = 50



**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

# Software Architecture :




## Functional description :

Donut : This function takes three arguments r1, r2 and Theta to generate a 3D donut (it also calls the Rotate_3D_Sphere to create the donut).

Rotate_3D_Sphere : This function takes 5 arguments x, y, z, rotation_axis and Theta (i) to rotate, with an angle of theta (i), the 3D donut along the rotation axis.

Norm_surf : This function takes 5 arguments x, y, z, Ndots (number of dots per circle = nombre of circles forming the 3Ddonut) and Theta to compute the surface normal vectors.

proj2D : This function takes 7 arguments x, y, z, l, L, d, D, normVect to perform a 2D projection over the (y, z) plan.

termprint : This function takes 1 argument donutmat and it prints it on terminal.

update2D : This function is in charge of animating effect in the terminal. It calls all the functions for each time step.



**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana

# Python Code :

**3D_PyDonut code link :**

You can clone it directly to your computer using the following link :
https://github.com/AitIzan/3D_PyDonut.git

That's All
^ ^

## Recommendations :

```
# 3D_PyDonut
3D donut's animation on terminal using python.
############################################
#  Author : Mohamed Ait-Izana              #
#  LinkedIn : Mohamed Ait-izana            #
#  Email : mohamed.aitizana@ensem.ac.ma    #
############################################
```

This project is an attempt to animate and plot a 3D donut both on terminal and using a pyplot figure.
The mathematical concepts used in the project are explained in the joint document.

How to use it ?
1 ) You will need Python3 installed on your computer as well as some libraries (matplotlib, numpy,mpl_toolkits..).
2 ) Run the 3D_PyDonut.py file and adjust the size of your terminal till you see the perfect shape of the donut ^ ^

NotaBene :
> The animation will be very slow because of the complexity of the program and the 3 animations that are running.
> No code optimization has been applied ! feel free to optimize it ^ ^.



**Author** : Mohamed Ait-Izana **Email** : mohamed.aitizana@ensem.ac.ma **LinkedIn** : Mohamed Ait-izana