BACCALAURÉAT

SESSION 2026

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°9

DURÉE DE L'ÉPREUVE : 1 heure

Le sujet comporte 5 pages numérotées de 1 / 5 à 5 / 5
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Cette situation d'évaluation comporte ce document ainsi que des fichiers de codes et de données présents sur l'ordinateur à la disposition du candidat. Le candidat doit restituer ce document avant de sortir de la salle d'examen. Le candidat doit agir en autonomie et faire preuve d'initiative tout au long de l'épreuve.
En cas de difficulté, le candidat peut solliciter l'examinateur afin de lui permettre de continuer la tâche. Des moments privilégiés pour solliciter l'examinateur sont indiqués dans le document sous la forme d'appels professeur.
L'examinateur peut intervenir à tout moment, s'il le juge utile.

Les fichiers OBJ constituent l'un des formats les plus utilisés dans le domaine de la modélisation 3D.

Ils servent à décrire une forme géométrique en listant ses sommets (les points dans l'espace) et ses faces (les surfaces reliant ces points). Grâce à ce principe, il est possible de représenter aussi bien des objets simples, comme un cube, que des modèles complexes issus de logiciels de modélisation.

Ce format est particulièrement apprécié parce qu'il est simple à lire, facile à manipuler et compatible avec la majorité des outils 3D. Cette simplicité vient du fait qu'un fichier OBJ n'est rien d'autre qu'un fichier texte, dans lequel on décrit ligne par ligne les différents éléments : le nom de l'objet (o), les sommets (v), les faces (f)…

Voici un exemple du contenu d'un fichier OBJ permettant de construire une seule face d'un cube.

```
o cube
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 1.0 0.0
v 1.0 0.0 0.0
f 1 2 3 4
```

Figure 1 – Contenu d'un fichier OBJ

Représentation de l'information

Pour manipuler ces fichiers OBJ, nous avons à disposition trois classes Python :

- La classe `Sommet` : elle représente un sommet, défini par trois entiers précisant les coordonnées x, y et z du sommet.
- La classe `Face` : elle représente une face d'un élément 3D, définie par une liste d'indices. Cet indice identifie un sommet présent dans la liste des sommets d'un `Objet3D`.
- La classe `Objet3D` : elle représente un élément 3D, défini par une liste d'objets de type `Sommet` et une liste d'objets de type Face.

Calcul du volume d'un élément 3D

On s'intéresse au calcul du volume d'un cube afin d'obtenir une estimation du temps d'impression à l'aide d'une imprimante 3D. Pour cela, il faut récupérer la longueur d'une arête du cube, à l'aide de ses différents sommets.

Pour calculer la longueur d'une arête, il suffit de calculer la distance entre deux sommets adjacents, c'est-à-dire deux sommets reliés par une arête.

Question 1

Écrire la méthode `distance` de la classe `Sommet`. Elle prend en paramètre un objet de type `Sommet`. La méthode renvoie la distance entre l'objet courant et l'objet de type `Sommet` passé en paramètre.
On rappelle que la distance entre deux points A et B dans un repère choisi s'obtient grâce à la formule

$$\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$$

avec $(x_A; y_A; z_A)$ les coordonnées du point A et $(x_B; y_B; z_B)$ les coordonnées du point B.
Pour calculer le résultat d'une racine carrée en Python, on peut utiliser la fonction `sqrt` du module `math`.

Il faut maintenant trouver deux sommets adjacents au sein de la liste de sommets de notre objet 3D. Cette liste de sommets est présente au sein de la classe `Objet3D` grâce à l'attribut sommets. Pour savoir si deux sommets sont adjacents, on dispose d'une méthode `est_adjacent` dans la classe `Sommet`.

Question 2

Écrire la méthode `trouver_sommets_adjacents` de la classe `Objet3D`. Elle renvoie un tuple composé de deux sommets adjacents, `None` si aucun sommet n'est adjacent.
Cette méthode doit parcourir l'ensemble des couples possibles entre deux sommets à l'aide de deux boucles imbriquées et tester si les sommets du couple sont adjacents. Il faudra donc utiliser la méthode est_adjacent présente dans la classe Sommet. La méthode renvoie le premier couple de sommets adjacents trouvé.

Appeler le professeur pour lui présenter votre réponse ou en cas de difficulté.

On dispose d'une fonction volume_cube présente dans le fichier main.py. Le dictionnaire `parametres_imprimante` contient des informations concernant une imprimante 3D.

- La valeur associée à la clé remplissage précise le taux de remplissage du plastique lors de l'impression. Il est représenté sous la forme d'un entier, précisant un pourcentage.
- La valeur associée à la clef vitesse_extrusion précise la vitesse d'impression de l'imprimante. Elle est représentée sous la forme d'un entier, précisant la vitesse en mm^3/s.

Question 3

Écrire une fonction `estimation_impression`. Elle prend en paramètre le volume réel d'un élément 3D sous la forme d'un entier et un dictionnaire contenant les informations d'une imprimante 3D. La fonction renvoie le temps total d'impression sous la forme d'un flottant, exprimé en seconde.

- Dans un premier temps, il faut calculer le volume d'impression de l'objet, c'est-à-dire en multipliant le volume réel par le taux de remplissage.
- Dans un second temps, il faut calculer le temps d'impression en secondes. On rappelle que le temps s'obtient en divisant le volume d'impression de l'objet par la vitesse d'extrusion de l'imprimante.

Manipulation d'éléments 3D

On propose un programme permettant de construire un objet de type Objet3D composé d'une seule face carrée (4 sommets, 1 face).

```
from Objet3D import Objet3D

objet = Objet3D()
objet.ajouter_sommet(0, 0, 0)
objet.ajouter_sommet(0, 1, 0)
objet.ajouter_sommet(1, 1, 0)
objet.ajouter_sommet(1, 0, 0)
objet.ajouter_face([1, 2, 3, 4])

objet.afficher()
```

L'appel à la méthode `ajouter_face` permet d'ajouter une face à notre élément 3D. Elle prend en paramètre une liste d'entiers représentant l'ordre d'ajout des sommets dans notre objet 3D. Par conséquent, l'entier 1 dans la liste [1, 2, 3, 4] correspond au premier sommet ajouté dans l'objet 3D, dont la position est (0, 0, 0).

Question 4

Modifier le programme présent dans le fichier main.py afin d'instancier et afficher la pyramide présentée ci-dessous.

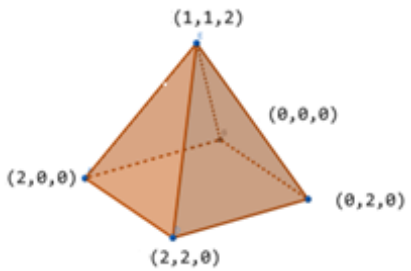

Figure 2 – Représentation d'une pyramide

Appeler le professeur pour lui présenter votre réponse ou en cas de difficulté.

On dispose d'une méthode exporter présente dans la classe `Objet3D`. Cette méthode exporte l'ensemble des sommets et des faces de l'élément 3D selon le format d'un fichier OBJ. Cette méthode prend en paramètre une chaîne de caractères correspondant au nouveau nom du fichier exporté.

On souhaite agrandir ou rétrécir un élément 3D selon un rapport (agrandissement ou réduction), et pouvoir ensuite exporter ce nouvel objet. On dispose dans la classe Objet3D de la méthode transformer, qui permet de transformer l'instance courante selon le rapport passé en paramètre.

Question 5

Analyser le fonctionnement de la méthode transformer. Proposer une amélioration afin qu'elle ne modifie plus l'instance courante. La méthode doit désormais renvoyer un nouvel objet résultant de la transformation, sans modifier l'instance d'origine.
Tester la nouvelle méthode sur la pyramide de la question 4 avec un rapport de 2, puis l'afficher pour observer l'agrandissement.

Appeler le professeur pour lui présenter votre réponse ou en cas de difficulté.

Description du dossier

Le dossier fourni au candidat sur l'ordinateur comporte les éléments suivants :

- une version PDF de l'énoncé ;
- Plusieurs fichiers de classe Python : `Objet3D.py`, `Face.py`, `Sommet.py`
- Un fichier `main.py`

Préparation de l'environnement

Pour faire fonctionner le code fourni dans le dossier, les bibliothèques suivantes doivent être présentes : `matplotlib`, `math`.