Devoir 1

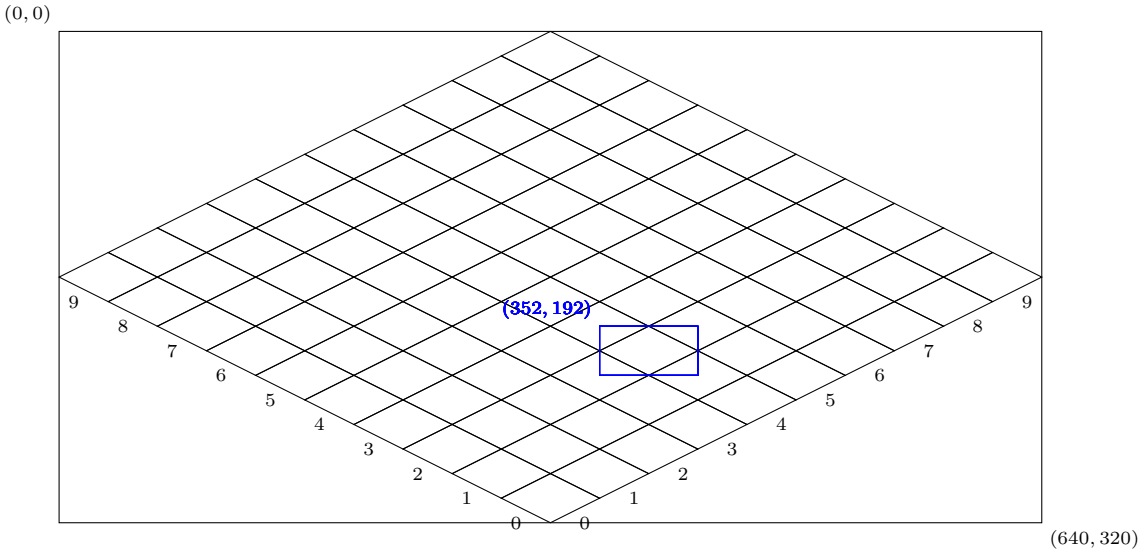
(à remettre au plus tard le 19 octobre, à 16h00)
(en classe ou dans la chute du département d'informatique, située au PK-4150)

Le devoir doit être rédigé **individuellement** et à l'**ordinateur** (il est cependant permis de remettre des annexes avec des **schémas dessinés** manuellement). Vous devez **justifier** chacune de vos réponses. Tout retard entraînera une pénalité de **20%** par jour ouvrable.

Question	1	2	3	4	5	Total
Sur	25	20	20	15	20	100
Note						

1. Supposons qu'une application graphique utilise des images de dimensions 64×32 pixels^2 pour représenter des tuiles en *projection isométrique* (dans le dessin ci-bas, un exemple d'image contenant une tuile est identifié par un rectangle bleu).

   (a) (15 points) Expliquez comment calculer la position du coin supérieur gauche d'une tuile 10×10 qu'on place dans une grille indexée comme l'illustre la grille ci-bas :



   Autrement dit, donnez le pseudocode d'une fonction

   fonction CoinSupGauche(i, j : indices) : point

   qui retourne la position que devrait occuper le coin supérieur gauche du rectangle contenant la tuile aux indices i et j. Par exemple, on s'attend à ce que l'appel CoinSupGauche$(4, 2)$ retourne la valeur $(352, 192)$ (voir le rectangle bleu dans l'image ci-haut).

(b) (10 points) Dans la question ci-bas, nous avons supposé que toutes les tuiles se trouvaient à la même hauteur. Supposons que nous souhaitions également supporter la possibilité d'afficher des tuiles plus hautes que d'autres. En utilisant des notions d'algèbres linéaire et vectorielle, expliquez de quelle façon vous traiteriez cette situation et montrez en particulier que certaines tuiles à des hauteurs différentes pourraient apparaître exactement au même endroit.

(c) (5 points *boni*) Le concepteur principal d'un jeu appelé *Super Hexagon* est auteur d'un autre jeu qui exploite le fait que différentes tuiles à des hauteurs différentes apparaissent au même endroit. Quel est le nom de ce jeu et qui en est l'auteur ?

2. (20 points) Supposez que les services suivants sont disponibles pour manipuler des maillages :

- $M \leftarrow$ MaillageVide() construit un maillage vide et le stocke dans la variable M.
- M.AjouterSommet(p) ajoute un sommet dans le maillage M et dont les coordonnées sont données par le point p.
- M.AjouterArete(p_1, p_2) ajoute une arête entre les sommets p_1 et p_2 dans le maillage M. Si un ou les deux sommets n'existent pas dans la structure, une erreur est signalée.
- M.AjouterFace(P) ajoute une face dont les sommets sont donnés par la liste P. Ces sommets doivent être énumérés dans l'ordre antihoraire lorsqu'on regarde vers l'*extérieur* de la face. De plus, s'il n'y a pas d'arêtes entre deux sommets consécutifs, une erreur est signalée.

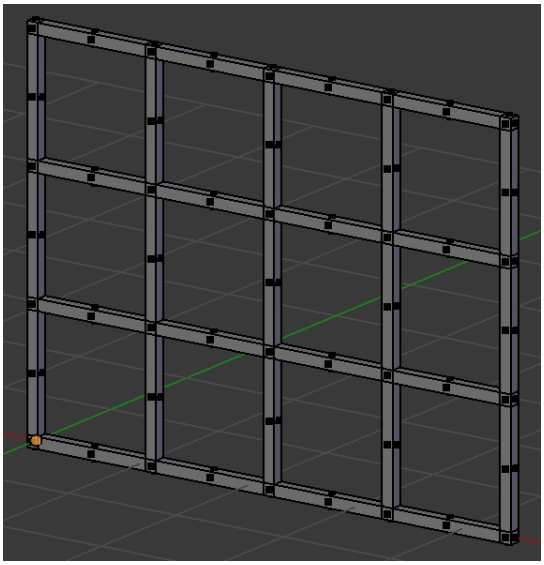
Donnez le **pseudocode** d'une fonction dont l'en-tête est

fonction Grille(m, n : entiers positifs, d, e : réels) : maillage

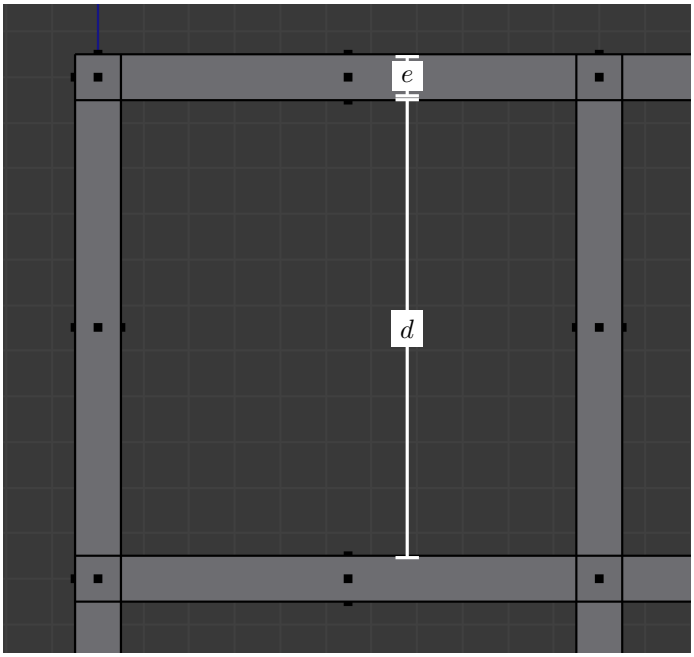
et qui retourne un maillage modélisant une grille de m carrés par n carrés, dont l'épaisseur est donnée par e et la distance entre les carrés de la grille est donnée par d. Autrement dit, si vous implémentiez votre fonction dans un script Blender et que vous utilisiez les paramètres $m = 3$, $n = 4$, $e = 0.1$ et $d = 1.0$, alors vous vous attendriez à obtenir un maillage similaire à celui illustré dans l'image ci-bas.

Attention ! La topologie de votre maillage doit être bien définie. Autrement dit, aucune erreur ne doit être signalée (vous devez donc d'abord ajouter les sommets, ensuite les arêtes, puis finalement les faces) et vous devez vous assurer que c'est le côté *visible* des faces (c'est-à-dire vers l'extérieur) qui est décrit en sens antihoraire.

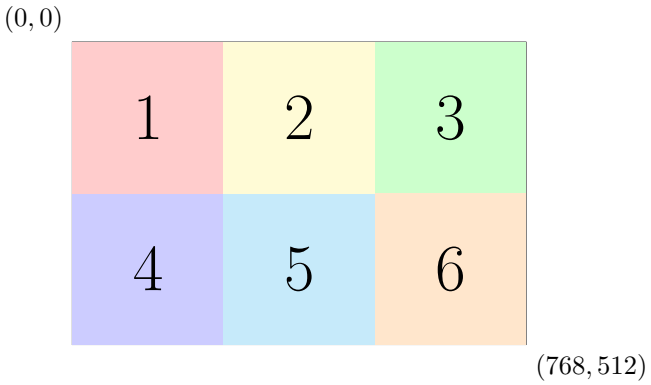
Note : Vous pouvez utiliser la notation $[p_1, p_2, p_3, p_4]$ pour dénoter une liste qui contient les quatre points p_1, p_2, p_3 et p_4 (autrement dit, vous pouvez utiliser des crochets pour dénoter des listes). Aussi, bien que ce ne soit pas demandé dans la question, n'hésitez pas à vérifier votre stratégie en l'implémentant comme un script Blender !



Voici une vue de face d'une portion de la grille avec les dimensions correspondantes :



3. (20 points) Considérez la texture suivante T décrite par une image de dimensions 512×768 pixels^2



Supposez que la couleur du pixel de la i-ième ligne et j-ième colonne est donné par $T[i][j]$ et supposez que vous avez un cube C dont le centre est donné par $C.\text{CENTRE}$ et la demi-longueur d'un côté est $C.\text{RAYON}$. Donnez le pseudocode d'une fonction dont l'en-tête est

fonction CouleurPixel(p : point, C : cube, T : texture) : couleur

qui indique pour chaque point p qui se trouve sur le cube C, la couleur du pixel qu'il devrait avoir. Il y a plusieurs solutions possibles à ce problème, mais vous devez respecter les contraintes suivantes :

- Chacun des 6 chiffres doit apparaître sur une face;
- Deux faces opposées doivent avoir une somme de 7.

Remarque : vous n'avez pas à valider si p est bien un point qui se trouve sur le cube, vous pouvez supposer que c'est toujours le cas.

4. (15 points) Deux objets décrivent une trajectoire rectiligne dans l'espace 3D selon les fonctions vectorielles suivantes :

$$\begin{aligned} \vec{r_1}(t) &= (t+5, 2t-2, 3t-1) \\ \vec{r_2}(t) &= (3t-3, 2t-2, 15-t), \end{aligned}$$

pour $t \geq 0$, où t représente le temps en heures. Est-ce que les trajectoires des objets s'intersectent ? Est-ce que les objets entreront en collision ? Si oui, à quel moment ? Justifiez.

5. (20 points) Considérez un cube dont les 8 sommets sont $(\pm 1, \pm 1, 1 \pm 1)$ et qui repose sur le plan $z = 0$. Une source de lumière se trouve en position $(4, 4, 2)$ et éclaire tout point de l'espace avec la même intensité, peu importe la distance (s'il existe un d'obstacle entre le point et la source de lumière, alors la lumière ne parvient pas au point). Décrivez la forme de l'ombre induite par le cube sur le plan et calculez son aire. *Remarque :* N'hésitez pas à joindre un schéma de la situation pour mieux expliquer votre réponse.