Infographie
Travaux dirigés
Feuille 4 (Dessin 3D)

Exercice 1. z-buffer

Nous allons ici simuler, en dimension 2, l'algorithme du z-buffer vu en cours. Étant donné (Ox) et (Oy) les deux axes repérant le plan, nous nous plaçons dans le cas d'une projection orthogonale (ou simple) sur la droite $y = 10$ (qui joue ici le rôle du plan de projection en 3D). On suppose que l'affichage est constitué de 10 pixels correspondant aux points $(0, 10)$, $(1, 10)$, $(2, 10)$,...,$(9, 10)$. Le maillage est constitué (voir Figure 1)

- des sommets: $P_0 = (3, 1)$, $P_1 = (3, 6)$, $P_2 = (7, 8)$, $P_3 = (5, 6)$, $P_4 = (8, 6)$.
- des mailles (ici des segments): $M_0 = (0, 1)$, $M_1 = (1, 2)$, $M_2 = (2, 3)$, $M_3 = (3, 4)$, $M_4 = (4, 0)$.

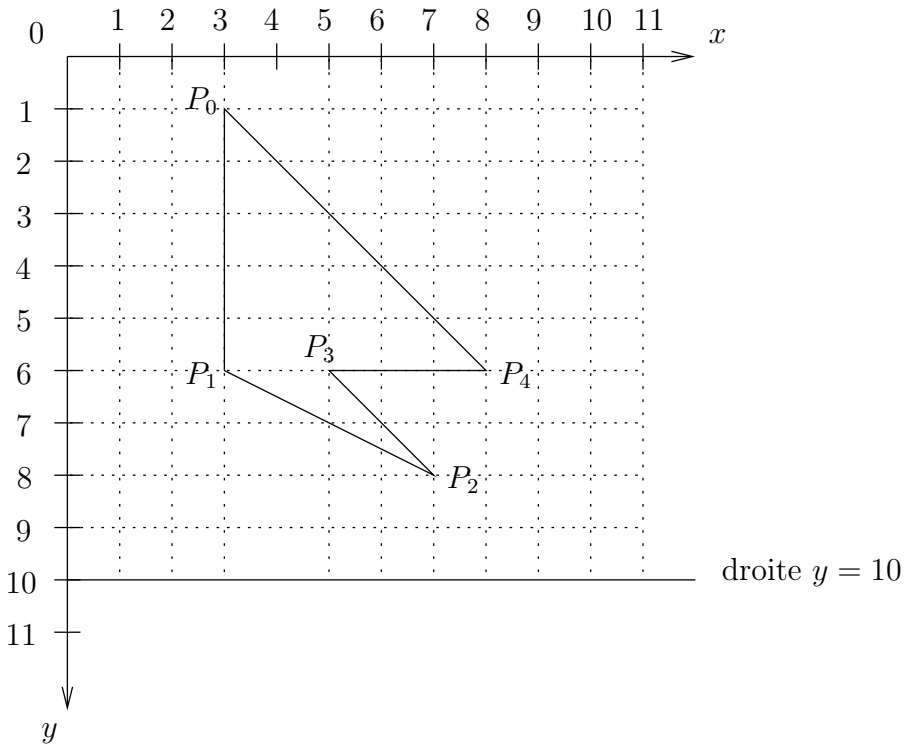

FIGURE 1. Exercice sur l'algorithme du z-buffer en dimension 2.

Donner le contenu des tableaux *altitude* et *maille Visible* après le traitement de chacune des mailles du maillage, au cours du déroulement de l'algorithme du z-buffer.

Exercice 2. Équation du plan de la maille

(1) Écrire une méthode *Point3D moins(Point3D P)*, de la classe *Point3D*, renvoyant le *Point3D* dont les coordonnées sont les différences entre les coordonnées du *Point3D* sur lequel est appelée la méthode et P.

(2) Écrire une méthode *Point3D vectoriel(Point3D P)*, de la classe *Point3D*, renvoyant le produit vectoriel entre le *Point3D* sur lequel est appelée la méthode et P.

(3) Écrire une méthode *Point4D maille2plan(int i)*, de la classe *Maillage*, renvoyant l'équation du plan de la maille d'indice i. Vous serez attentifs à ne pas diviser par 0.
(4) Écrire une méthode *Point4D lesMaille2lesPlans()*, de la classe *Maillage*, permettant de calculer les équations de plan de toutes les mailles. (Le résultat est mémorisé dans une variable *lesPlans* de la classe *Maillage*.)

Exercice 3. Projection inverse

(1) Écrire une méthode *Point3D plongePoint2D(float z)*, de la classe *Point2D*, qui ajoute une troisième coordonnée z au point2D auquel elle est appliquée.
(2) Écrire une méthode *float z_ intersection(Point3D A, Point3D B)*, de la classe *Point4D*, calculant la troisième coordonnée de l'intersection entre le plan défini par le *Point4D* et la droite passant par A et B.

Exercice 4. Lumière

(1) Écrire une méthode *normalise()*, de la classe *Point3D*, permettant de normaliser un *Point3D*. (Il a la même direction, mais est de longueur 1.)
(2) Écrire une méthode *float scalaire(Point3D P)*, de la classe *Point3D*, renvoyant le produit scalaire entre deux *Point3D*.
(3) Écrire une méthode *Point3D calculeLumiere(int x, int y, int i)*, de l'applet *dessin3D*, calculant la luminosité en un point du plan de projection de coordonnées (x, y), issue de la maille d'indice i. Vous supposerez que vous disposez :
- des caractéristiques de la lumière.
- des indices de réflexion. (Mémorisés dans des variables de la classe *Maillage*.)
- d'une méthode *Point3D projectionInverse(int x, int y, int i)*, de l'applet *dessin3D*, renvoyant le *Point3D* de la maille i se projetant en (x, y).
- d'une méthode *Point3D determineNormale(Point3D X, int i)*, de la classe *Maillage*, renvoyant la normale en X (X appartient à la maille d'indice i).

Exercice 5. Interpolation

Écrire une méthode *Point3D calculeCoefInterpolation(Point3D X,int i)*, de la classe *Maillage*, calculant les coefficients pour faire une interpolation au point X de valeurs que l'on connaît aux sommets de la maille d'indice i. (Ex.: Les coordonnées de la normale, les coordonnées de textures,...)