Algorithme 1: Rastérisation d'un tétraèdre

Données : un tétraèdre et une image 3D

Résultat : ensemble de voxels de l'image dont le centre est dans le tétraèdre

pour chaque *nœud du tétraèdre* **faire**
- **si** $Z < Z_{min}$ **alors** $Z_{min} = Z$;
- **si** $Z > Z_{max}$ **alors** $Z_{max} = Z$;

pour $Z = Z_{min}$ **à** Z_{max} **faire**
- initialisation Intersection = liste vide de points 2D (X,Y) dans le plan Z;
- **pour chaque** *arête du tétraèdre* **faire**
  - $\{X,Y,Z\}_0 = noeud(0).coordonnee(\{X,Y,Z\})$;
  - $\{X,Y,Z\}_1 = noeud(1).coordonnee(\{X,Y,Z\})$;
  - **si** $Z \leq max(Z_0,Z_1)$ **et** $Z \geq min(Z_0,Z_1)$ **alors**
    - **si** $Z_0 = Z_1$ **alors**
      - $Y_{min} = \min(Y_{min}, \min(Y_0,Y_1))$;
      - $Y_{max} = \max(Y_{max}, \max(Y_0,Y_1))$;
      - Intersection.ajoute$[(X_0,Y_0),(X_1,Y_1)]$;
    - **sinon**
      - $X_i = X_0 + \frac{Z-Z_0}{Z_1-Z_0}(X_1 - X_0)$;
      - $Y_i = Y_0 + \frac{Z-Z_0}{Z_1-Z_0}(Y_1 - Y_0)$;
      - $Y_{min} = \min(Y_{min},Y_i)$;
      - $Y_{max} = \max(Y_{max},Y_i)$;
      - Intersection.ajoute$[(X_i,Y_i)]$;
- **pour** $Y = Y_{min}$ **à** Y_{max} **faire**
  - **pour** $it_0 = Intersection.debut$ **à** $it_0 = Intersection.fin$ **faire**
    - **pour** $it_1 = it_0 + 1$ **à** $it_1 = Intersection.fin$ **faire**
      - $\{X,Y\}_{I0} = it_0.coordonnee\{X,Y\}$;
      - $\{X,Y\}_{I1} = it_1.coordonnee\{X,Y\}$;
      - **si** $Y \leq max(Y_{I0},Y_{I1})$ **et** $Y \geq min(Y_{I0},Y_{I1})$ **alors**
        - **si** $Y_{I0} = Y_{I1}$ **alors**
          - $X_{min} = \min(X_{min}, \min(X_{I0},X_{I1}))$;
          - $X_{max} = \max(X_{max}, \max(X_{I0},X_{I1}))$;
        - **sinon**
          - $X_i = X_{I0} + \frac{Y-Y_{I0}}{Y_{I1}-Y_{I0}}(X_{I1} - X_{I0})$;
          - $X_{min} = \min(X_{min},X_i)$;
          - $X_{max} = \max(X_{max},X_i)$;
  - **pour** $X = X_{min}$ **à** X_{max} **faire**
    - Resultat.ajoute(VoxelImage(X,Y,Z));