# ISIM
# TP2 : Les Blobs

Jonathan Fabrizio
LRDE - EPITA

## Objectif

L'objectif de ce T.P. est d'enrichir le *raytracer* en ajoutant le rendu de *blobs*. Il y a plusieurs techniques pour faire ce rendu. On s'appuiera sur l'algorithme des *marching cubes*.

## 1 Blobs V1.0

Dans un premier temps nous allons implémenter une version simpliste de l'algorithme.

### 1.1 Travail préliminaire

#### Triangles

Ajoutez un objet `Triangle` à votre *raytracer*. Vous dériverez cette classe de votre classe abstraite qui représente les objets. Vous respecterez donc l'interface que vous avez pour vos objets, ainsi il pourra s'intégrer au moteur de rendu du *raytracer* sans aucune modification. Vous devrez donc implémenter la méthode qui calcule l'intersection d'un rayon avec le triangle, la méthode qui calcul la normale en tout point du triangle (il n'y a pas de *backface culling*, votre triangle sera donc visible des deux cotés ; il faudra donc faire attention lors du calcul de la normale de bien renvoyer la normale dans le bon demi-espace) et la méthode qui renvoie la texture au point donné.

### 1.2 Rendu du *blob*

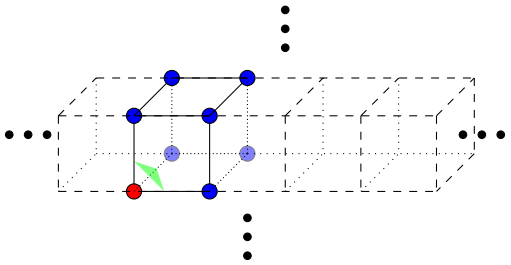
#### Création du *blob*

Créez une classe `Blob`. Cette classe doit pouvoir gérer un nombre illimité de points de potentiel (en revanche la fonction de décroissance du potentiel sera la même pour tous les points). Elle doit aussi :

- définir une zone de l'espace ou les potentiels seront évalués,
- définir un pas de discrétisassion de l'espace,
- fixer le niveau de potentiel de l'isosurface recherchée.

#### Calcul du maillage

Ajouter une méthode qui applique l'algorithme des *marching cubes* dans la zone de l'espace spécifiée pour générer un maillage de triangles - approximation de l'isosurface recherchée dans la zone de l'espace considérée. Pour cela, il faut parcourir l'espace à l'aide d'un cube et évaluer le potentiel en ses 8 sommets. En fonction du niveau de potentiel des huit sommets, vous pouvez en déduire la frontière dans ce cube comme dans la figure suivante (en rouge, un potentiel en dessous du seuil, en bleu les potentiels au dessus du seuil et en vert la frontière - triangle - déduite) :



Pour faire simple dans cette première partie, lorsque l'on identifiera l'isosurface, on placera les sommets des triangles au milieu des arrêtes du cube et les normales des triangles seront les normales "réelles" des triangles.

### Rendu

Ajoutez le maillage ainsi obtenu à une scène et lancer le calcul du rendu par le raytracer que vous avez développé lors de la précédente séance.

# 2 Blobs V2.0

Maintenant que notre *blob* - version simpliste - fonctionne, nous pouvons l'améliorer.

## 2.1 Travail préliminaire

#### Smooth Triangles

Ajoutez un objet `Smooth_Triangle` à votre *raytracer*. Dans cet objet, les normales devront être fournies en chaque sommet et elles seront interpolées en tout points du triangle de manière à donner une impression de lissage de la surface (voire, vous pourriez avoir une option pour activer ou non le lissage).

## 2.2 Rendu du *blob*

#### Calcul du maillage

Corrigez votre algorithme des *marching cubes* de manière à :

— affiner la position des sommets des triangles ; jusqu'à présent les sommets des triangles sont placés au milieu des arrêtes du cube, il est conseillé de faire une interpolation linéaire des potentiels pour essayer de trouver une position plus proche de l'isosurface recherchée,

— ne plus utiliser un maillage de triangles simple mais d'utiliser votre objet *smooth triangle* afin d'avoir un lissage de la surface finale.

#### Rendu

Ajoutez le maillage ainsi obtenu à une scène et lancer le calcul du rendu par le raytracer que vous avez développé lors de la précédente séance.

# 3 Pour aller plus loin (Optionnel)

#### Volume englobant

Lorsque le maillage de triangles devient dense, le temps de rendu augmente énormément. Pour réduire un peu ce temps, il est possible de définir un volume englobant. Si un rayon n'intersecte pas ce volume englobant, il n'a aucune chance de toucher un triangle du blob.

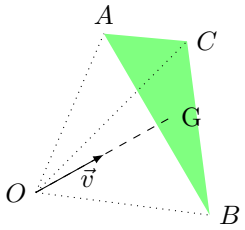
Utiliser donc un volume englobant pour réduire le temps de calcul du rendu. Dans ce cas vous pouvez prendre comme volume englobant la zone où nous avons appliqué les *marching cubes*. Cette zone est délimitée par un cube. Toutefois, ce n'est pas une obligation, vous pouvez choisir librement d'autres stratégies.

# 4 Un peu d'aide

#### Calcul de l'intersection rayon/triangle

Il y a plusieurs manières de calculer cette intersection. C'est un point critique pour la vitesse du *raytracer*. Vous pouvez utiliser le calcul que vous voulez. D'une manière générale il faut calculer l'intersection du rayon avec le plan qui porte le triangle, puis vérifier que le point d'intersection appartient au triangle.

Si vous voulez gagner un peu de temps sur cette partie, je vous propose une solution simple (on ne se préoccupe pas des cas dégénérés) : Si le rayon a pour origin $O$ et a pour vecteur directeur $\vec{v}$ et si on veut savoir si le rayon intersecte le triangle $ABC$ :




On peut exprimer $\vec{v}$ comme combinaison linéaire de $\overrightarrow{OA}$, $\overrightarrow{OB}$ et $\overrightarrow{OC}$. On a donc $\vec{v} = \alpha\overrightarrow{OA} + \beta\overrightarrow{OB} + \gamma\overrightarrow{OC}$. On peut poser G comme le barycentre de $(\alpha, \vec{A})$, $(\beta, \vec{B})$ et $(\gamma, \vec{B})$. On a donc $(\alpha+\beta+\gamma)\overrightarrow{OG} = \alpha\overrightarrow{OA} + \beta\overrightarrow{OB} + \gamma\overrightarrow{OC}$. Du coup, on peut en conclure que :

$$\overrightarrow{OG} = \frac{(\alpha\overrightarrow{OA} + \beta\overrightarrow{OB} + \gamma\overrightarrow{OC})}{(\alpha + \beta + \gamma)} \tag{1}$$

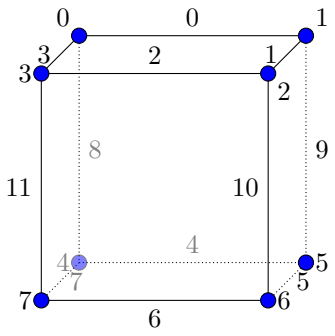
On calcule les coefficients par une inversion matricielle (tirée de $\vec{v} = \alpha\overrightarrow{OA} + \beta\overrightarrow{OB} + \gamma\overrightarrow{OC}$). Pour savoir enfin si le point appartient au triangle il faut que les coefficients ($\alpha$, $\beta$ et $\gamma$) soient de même signe.

Ce n'est qu'une solution parmi beaucoup d'autres. Vous pouvez utiliser une autre solution plus rapide si vous le souhaitez.

### Calcul de la frontière dans un cube en fonction des potentiels

Vous avez évalué le potentiel en 8 points et en fonction de ces 8 points vous devez déduire où se trouve la frontière. Cela fait en tout 256 combinaisons possibles. C'est donc un peu fastidieux à lister même s'il y a peu de cas de base et que les autres sont déduits par symétrie. Je vous propose de vous lister les triangles des frontières en fonction des configurations pour vous faire gagner un peu de temps.

Imaginons que l'on a le cube suivant et que les sommets et les arrêtes sont numéroté comme suit :




Pour chaque sommet, on peut savoir s'il est en dessous ou au dessus du potentiel recherché $S$. Vu qu'il y a 2 états par sommet et 8 sommets, il y a 256 frontières possibles. On va faire un adressage indéxé pour se simplifier la vie.

```
int index = 0;
if (potentiel(0) < S) index |= 1;
if (potentiel(1) < S) index |= 2;
if (potentiel(2) < S) index |= 4;
if (potentiel(3) < S) index |= 8;
if (potentiel(4) < S) index |= 16;
if (potentiel(5) < S) index |= 32;
if (potentiel(6) < S) index |= 64;
if (potentiel(7) < S) index |= 128;
```

On obtient un code unique pour chaque configuration.

Voilà la liste des frontières à construire pour chacune des 256 configurations. Chaque ligne représente une configuration et chaque numéro indique sur quelle arrête le sommet du triangle doit être ajouté ; il y a au plus 5 triangles par frontière donc $3 \times 5 = 15$ valeurs par ligne :

```
{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 8, 3, 9, 8, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 1, 2, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1},
```

```
{9, 2, 10, 0, 2, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 8, 3, 2, 10, 8, 10, 9, 8, -1, -1, -1, -1, -1, -1},
{3, 11, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 11, 2, 8, 11, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 9, 0, 2, 3, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 11, 2, 1, 9, 11, 9, 8, 11, -1, -1, -1, -1, -1, -1},
{3, 10, 1, 11, 10, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 10, 1, 0, 8, 10, 8, 11, 10, -1, -1, -1, -1, -1, -1},
{3, 9, 0, 3, 11, 9, 11, 10, 9, -1, -1, -1, -1, -1, -1},
{9, 8, 10, 10, 8, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 7, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 3, 0, 7, 3, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 9, 8, 4, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 1, 9, 4, 7, 1, 7, 3, 1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 8, 4, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 4, 7, 3, 0, 4, 1, 2, 10, -1, -1, -1, -1, -1, -1},
{9, 2, 10, 9, 0, 2, 8, 4, 7, -1, -1, -1, -1, -1, -1},
{2, 10, 9, 2, 9, 7, 2, 7, 3, 7, 9, 4, -1, -1, -1},
{8, 4, 7, 3, 11, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 4, 7, 11, 2, 4, 2, 0, 4, -1, -1, -1, -1, -1, -1},
{9, 0, 1, 8, 4, 7, 2, 3, 11, -1, -1, -1, -1, -1, -1},
{4, 7, 11, 9, 4, 11, 9, 11, 2, 9, 2, 1, -1, -1, -1},
{3, 10, 1, 3, 11, 10, 7, 8, 4, -1, -1, -1, -1, -1, -1},
{1, 11, 10, 1, 4, 11, 1, 0, 4, 7, 11, 4, -1, -1, -1},
{4, 7, 8, 9, 0, 11, 9, 11, 10, 11, 0, 3, -1, -1, -1},
{4, 7, 11, 4, 11, 9, 9, 11, 10, -1, -1, -1, -1, -1, -1},
{9, 5, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 5, 4, 0, 8, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 5, 4, 1, 5, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 5, 4, 8, 3, 5, 3, 1, 5, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 9, 5, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 8, 1, 2, 10, 4, 9, 5, -1, -1, -1, -1, -1, -1},
{5, 2, 10, 5, 4, 2, 4, 0, 2, -1, -1, -1, -1, -1, -1},
{2, 10, 5, 3, 2, 5, 3, 5, 4, 3, 4, 8, -1, -1, -1},
{9, 5, 4, 2, 3, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 11, 2, 0, 8, 11, 4, 9, 5, -1, -1, -1, -1, -1, -1},
{0, 5, 4, 0, 1, 5, 2, 3, 11, -1, -1, -1, -1, -1, -1},
{2, 1, 5, 2, 5, 8, 2, 8, 11, 4, 8, 5, -1, -1, -1},
{10, 3, 11, 10, 1, 3, 9, 5, 4, -1, -1, -1, -1, -1, -1},
{4, 9, 5, 0, 8, 1, 8, 10, 1, 8, 11, 10, -1, -1, -1},
{5, 4, 0, 5, 0, 11, 5, 11, 10, 11, 0, 3, -1, -1, -1},
{5, 4, 8, 5, 8, 10, 10, 8, 11, -1, -1, -1, -1, -1, -1},
{9, 7, 8, 5, 7, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 3, 0, 9, 5, 3, 5, 7, 3, -1, -1, -1, -1, -1, -1},
{0, 7, 8, 0, 1, 7, 1, 5, 7, -1, -1, -1, -1, -1, -1},
{1, 5, 3, 3, 5, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 7, 8, 9, 5, 7, 10, 1, 2, -1, -1, -1, -1, -1, -1},
{10, 1, 2, 9, 5, 0, 5, 3, 0, 5, 7, 3, -1, -1, -1},
{8, 0, 2, 8, 2, 5, 8, 5, 7, 10, 5, 2, -1, -1, -1},
{2, 10, 5, 2, 5, 3, 3, 5, 7, -1, -1, -1, -1, -1, -1},
{7, 9, 5, 7, 8, 9, 3, 11, 2, -1, -1, -1, -1, -1, -1},
{9, 5, 7, 9, 7, 2, 9, 2, 0, 2, 7, 11, -1, -1, -1},
{2, 3, 11, 0, 1, 8, 1, 7, 8, 1, 5, 7, -1, -1, -1},
{11, 2, 1, 11, 1, 7, 7, 1, 5, -1, -1, -1, -1, -1, -1},
{9, 5, 8, 8, 5, 7, 10, 1, 3, 10, 3, 11, -1, -1, -1},
{5, 7, 0, 5, 0, 9, 7, 11, 0, 1, 0, 10, 11, 10, 0},
{11, 10, 0, 11, 0, 3, 10, 5, 0, 8, 0, 7, 5, 7, 0},
{11, 10, 5, 7, 11, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{10, 6, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 5, 10, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 0, 1, 5, 10, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 8, 3, 1, 9, 8, 5, 10, 6, -1, -1, -1, -1, -1, -1},
{1, 6, 5, 2, 6, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 6, 5, 1, 2, 6, 3, 0, 8, -1, -1, -1, -1, -1, -1},
{9, 6, 5, 9, 0, 6, 0, 2, 6, -1, -1, -1, -1, -1, -1},
{5, 9, 8, 5, 8, 2, 5, 2, 6, 3, 2, 8, -1, -1, -1},
{2, 3, 11, 10, 6, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 0, 8, 11, 2, 0, 10, 6, 5, -1, -1, -1, -1, -1, -1},
{0, 1, 9, 2, 3, 11, 5, 10, 6, -1, -1, -1, -1, -1, -1},
{5, 10, 6, 1, 9, 2, 9, 11, 2, 9, 8, 11, -1, -1, -1},
{6, 3, 11, 6, 5, 3, 5, 1, 3, -1, -1, -1, -1, -1, -1},
{0, 8, 11, 0, 11, 5, 0, 5, 1, 5, 11, 6, -1, -1, -1},
{3, 11, 6, 0, 3, 6, 0, 6, 5, 0, 5, 9, -1, -1, -1},
{6, 5, 9, 6, 9, 11, 11, 9, 8, -1, -1, -1, -1, -1, -1},
{5, 10, 6, 4, 7, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1},
```

{4, 3, 0, 4, 7, 3, 6, 5, 10, -1, -1, -1, -1, -1, -1},
{1, 9, 0, 5, 10, 6, 8, 4, 7, -1, -1, -1, -1, -1, -1},
{10, 6, 5, 1, 9, 7, 1, 7, 3, 7, 9, 4, -1, -1, -1},
{6, 1, 2, 6, 5, 1, 4, 7, 8, -1, -1, -1, -1, -1, -1},
{1, 2, 5, 5, 2, 6, 3, 0, 4, 3, 4, 7, -1, -1, -1},
{8, 4, 7, 9, 0, 5, 0, 6, 5, 0, 2, 6, -1, -1, -1},
{7, 3, 9, 7, 9, 4, 3, 2, 9, 5, 9, 6, 2, 6, 9},
{3, 11, 2, 7, 8, 4, 10, 6, 5, -1, -1, -1, -1, -1, -1},
{5, 10, 6, 4, 7, 2, 4, 2, 0, 2, 7, 11, -1, -1, -1},
{0, 1, 9, 4, 7, 8, 2, 3, 11, 5, 10, 6, -1, -1, -1},
{9, 2, 1, 9, 11, 2, 9, 4, 11, 7, 11, 4, 5, 10, 6},
{8, 4, 7, 3, 11, 5, 3, 5, 1, 5, 11, 6, -1, -1, -1},
{5, 1, 11, 5, 11, 6, 1, 0, 11, 7, 11, 4, 0, 4, 11},
{0, 5, 9, 0, 6, 5, 0, 3, 6, 11, 6, 3, 8, 4, 7},
{6, 5, 9, 6, 9, 11, 4, 7, 9, 7, 11, 9, -1, -1, -1},
{10, 4, 9, 6, 4, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 10, 6, 4, 9, 10, 0, 8, 3, -1, -1, -1, -1, -1, -1},
{10, 0, 1, 10, 6, 0, 6, 4, 0, -1, -1, -1, -1, -1, -1},
{8, 3, 1, 8, 1, 6, 8, 6, 4, 6, 1, 10, -1, -1, -1},
{1, 4, 9, 1, 2, 4, 2, 6, 4, -1, -1, -1, -1, -1, -1},
{3, 0, 8, 1, 2, 9, 2, 4, 9, 2, 6, 4, -1, -1, -1},
{0, 2, 4, 4, 2, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 3, 2, 8, 2, 4, 4, 2, 6, -1, -1, -1, -1, -1, -1},
{10, 4, 9, 10, 6, 4, 11, 2, 3, -1, -1, -1, -1, -1, -1},
{0, 8, 2, 2, 8, 11, 4, 9, 10, 4, 10, 6, -1, -1, -1},
{3, 11, 2, 0, 1, 6, 0, 6, 4, 6, 1, 10, -1, -1, -1},
{6, 4, 1, 6, 1, 10, 4, 8, 1, 2, 1, 11, 8, 11, 1},
{9, 6, 4, 9, 3, 6, 9, 1, 3, 11, 6, 3, -1, -1, -1},
{8, 11, 1, 8, 1, 0, 11, 6, 1, 9, 1, 4, 6, 4, 1},
{3, 11, 6, 3, 6, 0, 0, 6, 4, -1, -1, -1, -1, -1, -1},
{6, 4, 8, 11, 6, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 10, 6, 7, 8, 10, 8, 9, 10, -1, -1, -1, -1, -1, -1},
{0, 7, 3, 0, 10, 7, 0, 9, 10, 6, 7, 10, -1, -1, -1},
{10, 6, 7, 1, 10, 7, 1, 7, 8, 1, 8, 0, -1, -1, -1},
{10, 6, 7, 10, 7, 1, 1, 7, 3, -1, -1, -1, -1, -1, -1},
{1, 2, 6, 1, 6, 8, 1, 8, 9, 8, 6, 7, -1, -1, -1},
{2, 6, 9, 2, 9, 1, 6, 7, 9, 0, 9, 3, 7, 3, 9},
{7, 8, 0, 7, 0, 6, 6, 0, 2, -1, -1, -1, -1, -1, -1},
{7, 3, 2, 6, 7, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 3, 11, 10, 6, 8, 10, 8, 9, 8, 6, 7, -1, -1, -1},
{2, 0, 7, 2, 7, 11, 0, 9, 7, 6, 7, 10, 9, 10, 7},
{1, 8, 0, 1, 7, 8, 1, 10, 7, 6, 7, 10, 2, 3, 11},
{11, 2, 1, 11, 1, 7, 10, 6, 1, 6, 7, 1, -1, -1, -1},
{8, 9, 6, 8, 6, 7, 9, 1, 6, 11, 6, 3, 1, 3, 6},
{0, 9, 1, 11, 6, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 8, 0, 7, 0, 6, 3, 11, 0, 11, 6, 0, -1, -1, -1},
{7, 11, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 6, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 8, 11, 7, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 9, 11, 7, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 1, 9, 8, 3, 1, 11, 7, 6, -1, -1, -1, -1, -1, -1},
{10, 1, 2, 6, 11, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 3, 0, 8, 6, 11, 7, -1, -1, -1, -1, -1, -1},
{2, 9, 0, 2, 10, 9, 6, 11, 7, -1, -1, -1, -1, -1, -1},
{6, 11, 7, 2, 10, 3, 10, 8, 3, 10, 9, 8, -1, -1, -1},
{7, 2, 3, 6, 2, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 0, 8, 7, 6, 0, 6, 2, 0, -1, -1, -1, -1, -1, -1},
{2, 7, 6, 2, 3, 7, 0, 1, 9, -1, -1, -1, -1, -1, -1},
{1, 6, 2, 1, 8, 6, 1, 9, 8, 8, 7, 6, -1, -1, -1},
{10, 7, 6, 10, 1, 7, 1, 3, 7, -1, -1, -1, -1, -1, -1},
{10, 7, 6, 1, 7, 10, 1, 8, 7, 1, 0, 8, -1, -1, -1},
{0, 3, 7, 0, 7, 10, 0, 10, 9, 6, 10, 7, -1, -1, -1},
{7, 6, 10, 7, 10, 8, 8, 10, 9, -1, -1, -1, -1, -1, -1},
{6, 8, 4, 11, 8, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 6, 11, 3, 0, 6, 0, 4, 6, -1, -1, -1, -1, -1, -1},
{8, 6, 11, 8, 4, 6, 9, 0, 1, -1, -1, -1, -1, -1, -1},
{9, 4, 6, 9, 6, 3, 9, 3, 1, 11, 3, 6, -1, -1, -1},
{6, 8, 4, 6, 11, 8, 2, 10, 1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 3, 0, 11, 0, 6, 11, 0, 4, 6, -1, -1, -1},
{4, 11, 8, 4, 6, 11, 0, 2, 9, 2, 10, 9, -1, -1, -1},
{10, 9, 3, 10, 3, 2, 9, 4, 3, 11, 3, 6, 4, 6, 3},
{8, 2, 3, 8, 4, 2, 4, 6, 2, -1, -1, -1, -1, -1, -1},
{0, 4, 2, 4, 6, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 9, 0, 2, 3, 4, 2, 4, 6, 4, 3, 8, -1, -1, -1},
{1, 9, 4, 1, 4, 2, 2, 4, 6, -1, -1, -1, -1, -1, -1},

{8, 1, 3, 8, 6, 1, 8, 4, 6, 6, 10, 1, -1, -1, -1},
{10, 1, 0, 10, 0, 6, 6, 0, 4, -1, -1, -1, -1, -1, -1},
{4, 6, 3, 4, 3, 8, 6, 10, 3, 0, 3, 9, 10, 9, 3},
{10, 9, 4, 6, 10, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 5, 7, 6, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 4, 9, 5, 11, 7, 6, -1, -1, -1, -1, -1, -1},
{5, 0, 1, 5, 4, 0, 7, 6, 11, -1, -1, -1, -1, -1, -1},
{11, 7, 6, 8, 3, 4, 3, 5, 4, 3, 1, 5, -1, -1, -1},
{9, 5, 4, 10, 1, 2, 7, 6, 11, -1, -1, -1, -1, -1, -1},
{6, 11, 7, 1, 2, 10, 0, 8, 3, 4, 9, 5, -1, -1, -1},
{7, 6, 11, 5, 4, 10, 4, 2, 10, 4, 0, 2, -1, -1, -1},
{3, 4, 8, 3, 5, 4, 3, 2, 5, 10, 5, 2, 11, 7, 6},
{7, 2, 3, 7, 6, 2, 5, 4, 9, -1, -1, -1, -1, -1, -1},
{9, 5, 4, 0, 8, 6, 0, 6, 2, 6, 8, 7, -1, -1, -1},
{3, 6, 2, 3, 7, 6, 1, 5, 0, 5, 4, 0, -1, -1, -1},
{6, 2, 8, 6, 8, 7, 2, 1, 8, 4, 8, 5, 1, 5, 8},
{9, 5, 4, 10, 1, 6, 1, 7, 6, 1, 3, 7, -1, -1, -1},
{1, 6, 10, 1, 7, 6, 1, 0, 7, 8, 7, 0, 9, 5, 4},
{4, 0, 10, 4, 10, 5, 0, 3, 10, 6, 10, 7, 3, 7, 10},
{7, 6, 10, 7, 10, 8, 5, 4, 10, 4, 8, 10, -1, -1, -1},
{6, 9, 5, 6, 11, 9, 11, 8, 9, -1, -1, -1, -1, -1, -1},
{3, 6, 11, 0, 6, 3, 0, 5, 6, 0, 9, 5, -1, -1, -1},
{0, 11, 8, 0, 5, 11, 0, 1, 5, 5, 6, 11, -1, -1, -1},
{6, 11, 3, 6, 3, 5, 5, 3, 1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 9, 5, 11, 9, 11, 8, 11, 5, 6, -1, -1, -1},
{0, 11, 3, 0, 6, 11, 0, 9, 6, 5, 6, 9, 1, 2, 10},
{11, 8, 5, 11, 5, 6, 8, 0, 5, 10, 5, 2, 0, 2, 5},
{6, 11, 3, 6, 3, 5, 2, 10, 3, 10, 5, 3, -1, -1, -1},
{5, 8, 9, 5, 2, 8, 5, 6, 2, 3, 8, 2, -1, -1, -1},
{9, 5, 6, 9, 6, 0, 0, 6, 2, -1, -1, -1, -1, -1, -1},
{1, 5, 8, 1, 8, 0, 5, 6, 8, 3, 8, 2, 6, 2, 8},
{1, 5, 6, 2, 1, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 3, 6, 1, 6, 10, 3, 8, 6, 5, 6, 9, 8, 9, 6},
{10, 1, 0, 10, 0, 6, 9, 5, 0, 5, 6, 0, -1, -1, -1},
{0, 3, 8, 5, 6, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{10, 5, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 5, 10, 7, 5, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 5, 10, 11, 7, 5, 8, 3, 0, -1, -1, -1, -1, -1, -1},
{5, 11, 7, 5, 10, 11, 1, 9, 0, -1, -1, -1, -1, -1, -1},
{10, 7, 5, 10, 11, 7, 9, 8, 1, 8, 3, 1, -1, -1, -1},
{11, 1, 2, 11, 7, 1, 7, 5, 1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 1, 2, 7, 1, 7, 5, 7, 2, 11, -1, -1, -1},
{9, 7, 5, 9, 2, 7, 9, 0, 2, 2, 11, 7, -1, -1, -1},
{7, 5, 2, 7, 2, 11, 5, 9, 2, 3, 2, 8, 9, 8, 2},
{2, 5, 10, 2, 3, 5, 3, 7, 5, -1, -1, -1, -1, -1, -1},
{8, 2, 0, 8, 5, 2, 8, 7, 5, 10, 2, 5, -1, -1, -1},
{9, 0, 1, 5, 10, 3, 5, 3, 7, 3, 10, 2, -1, -1, -1},
{9, 8, 2, 9, 2, 1, 8, 7, 2, 10, 2, 5, 7, 5, 2},
{1, 3, 5, 3, 7, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 7, 0, 7, 1, 1, 7, 5, -1, -1, -1, -1, -1, -1},
{9, 0, 3, 9, 3, 5, 5, 3, 7, -1, -1, -1, -1, -1, -1},
{9, 8, 7, 5, 9, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{5, 8, 4, 5, 10, 8, 10, 11, 8, -1, -1, -1, -1, -1, -1},
{5, 0, 4, 5, 11, 0, 5, 10, 11, 11, 3, 0, -1, -1, -1},
{0, 1, 9, 8, 4, 10, 8, 10, 11, 10, 4, 5, -1, -1, -1},
{10, 11, 4, 10, 4, 5, 11, 3, 4, 9, 4, 1, 3, 1, 4},
{2, 5, 1, 2, 8, 5, 2, 11, 8, 4, 5, 8, -1, -1, -1},
{0, 4, 11, 0, 11, 3, 4, 5, 11, 2, 11, 1, 5, 1, 11},
{0, 2, 5, 0, 5, 9, 2, 11, 5, 4, 5, 8, 11, 8, 5},
{9, 4, 5, 2, 11, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 5, 10, 3, 5, 2, 3, 4, 5, 3, 8, 4, -1, -1, -1},
{5, 10, 2, 5, 2, 4, 4, 2, 0, -1, -1, -1, -1, -1, -1},
{3, 10, 2, 3, 5, 10, 3, 8, 5, 4, 5, 8, 0, 1, 9},
{5, 10, 2, 5, 2, 4, 1, 9, 2, 9, 4, 2, -1, -1, -1},
{8, 4, 5, 8, 5, 3, 3, 5, 1, -1, -1, -1, -1, -1, -1},
{0, 4, 5, 1, 0, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 4, 5, 8, 5, 3, 9, 0, 5, 0, 3, 5, -1, -1, -1},
{9, 4, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 11, 7, 4, 9, 11, 9, 10, 11, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 4, 9, 7, 9, 11, 7, 9, 10, 11, -1, -1, -1},
{1, 10, 11, 1, 11, 4, 1, 4, 0, 7, 4, 11, -1, -1, -1},
{3, 1, 4, 3, 4, 8, 1, 10, 4, 7, 4, 11, 10, 11, 4},
{4, 11, 7, 9, 11, 4, 9, 2, 11, 9, 1, 2, -1, -1, -1},
{9, 7, 4, 9, 11, 7, 9, 1, 11, 2, 11, 1, 0, 8, 3},
{11, 7, 4, 11, 4, 2, 2, 4, 0, -1, -1, -1, -1, -1, -1},

```
{11, 7, 4, 11, 4, 2, 8, 3, 4, 3, 2, 4, -1, -1, -1},
{2, 9, 10, 2, 7, 9, 2, 3, 7, 7, 4, 9, -1, -1, -1},
{9, 10, 7, 9, 7, 4, 10, 2, 7, 8, 7, 0, 2, 0, 7},
{3, 7, 10, 3, 10, 2, 7, 4, 10, 1, 10, 0, 4, 0, 10},
{1, 10, 2, 8, 7, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 1, 4, 1, 7, 7, 1, 3, -1, -1, -1, -1, -1, -1},
{4, 9, 1, 4, 1, 7, 0, 8, 1, 8, 7, 1, -1, -1, -1},
{4, 0, 3, 7, 4, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 8, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 10, 8, 10, 11, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 9, 3, 9, 11, 11, 9, 10, -1, -1, -1, -1, -1, -1},
{0, 1, 10, 0, 10, 8, 8, 10, 11, -1, -1, -1, -1, -1, -1},
{3, 1, 10, 11, 3, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 11, 1, 11, 9, 9, 11, 8, -1, -1, -1, -1, -1, -1},
{3, 0, 9, 3, 9, 11, 1, 2, 9, 2, 11, 9, -1, -1, -1},
{0, 2, 11, 8, 0, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 2, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 3, 8, 2, 8, 10, 10, 8, 9, -1, -1, -1, -1, -1, -1},
{9, 10, 2, 0, 9, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 3, 8, 2, 8, 10, 0, 1, 8, 1, 10, 8, -1, -1, -1},
{1, 10, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 3, 8, 9, 1, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 9, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 3, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}
```