Le composeur d'image par tuiles

Yves Mathieu, Tarik Graba

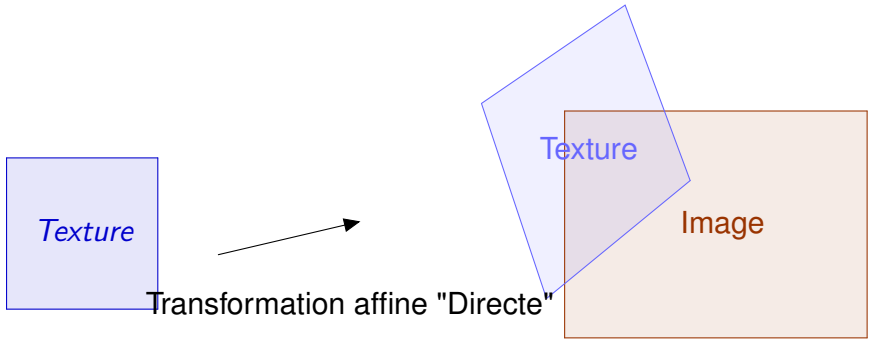
ELEC342

1 Fonctionalités du composeur d'images

Le composeur à pour rôle de générer un flux vidéo temps réel à partir de la composition de différentes textures. Les textures peuvent être des images fixes (images, logos, curseurs...) ou animées (vidéos entrantes).

Les textures sont des images rectangulaires codées en niveau de gris sur 8 bits. Elles sont affectées de transformées géométriques (**transformées affines**), de masques binaires (pour faire le découpage d'un contour d'objet) et de coefficients de opacité. Les transformées géométriques et les coefficients d'opacité doivent pouvoir eux-mêmes être modifiés en temps réel.

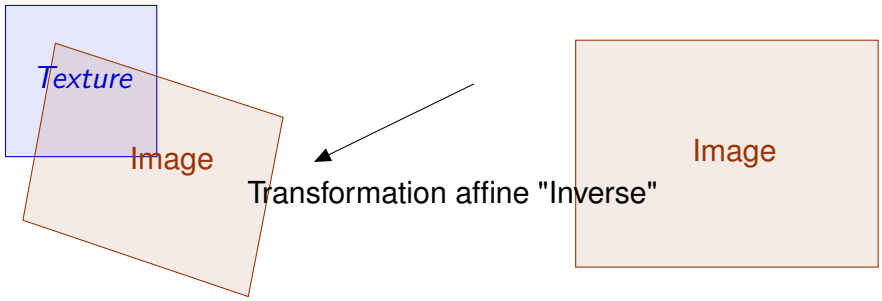
2 Transformations directes, Transformation inverses



Cette figure présente une possible transformation affine d'une texture vers l'image à afficher. La texture peut se retrouver partiellement ou complètement en dehors de l'image. Nous utiliserons le vocable "**Transformation directe**" pour représenter la transformation depuis l'espace de la texture vers l'espace de l'image.

La transformation directe est la transformation qui sera spécifiée au niveau de la composition. Elle sera utilisée, de plus, pour déterminer les zones de l'image recouvertes par une texture.

La seconde figure présente la "**Transformation inverse**", c'est a dire la transformation depuis l'espace de l'image vers l'espace de texture.



La connaissance des transformations inverses est nécessaire au composeur. En effet le composeur effectue un parcours régulier (lignes/colonnes) de l'image à générer et "va chercher" les pixels des textures à l'aide des transformations inverses.

3 Traitement par "tuiles"

L'algorithme de traitement de base utilisé de manière classique pour faire la composition d'une image peut être présenté par le pseudo-code suivant :

```
Initialiser l'image a la couleur du fond dans la mémoire externe
Pour chaque texture ayant une intersection non vide avec l'image:
   Lire l'image courante
   Lire la texture
   Composer la texture avec l'image courante
   Ecrire l'image courante
FinPour
```

Les textures sont traitées dans l'ordre de l'algorithme du "peintre" (de la plus éloignée de l'observateur à la plus proche). Ce type de méthode a l'inconvénient de nécessiter 3 entrées/sorties dans la mémoire d'image pour chaque pixel de chaque texture interceptant l'image. Une façon de contourner le problème est de découper l'image en tuiles (blocs rectangulaires) de taille fixe, et de traiter les tuiles dans une mémoire locale.

L'algorithme devient alors :

```
Pour chaque tuile de l'image
  Initialiser une tuile avec la couleur de fond en mémoire locale
  Pour chaque texture interceptant la tuile courante
      Lire la portion de texture interceptant la tuile
      Composer la portion de texture avec la tuile
  FinPour
  Sauver la tuile en mémoire externe
FinPour
```

Ainsi il ne reste, dans la mémoire principale, plus qu'une opération de lecture par pixel de texture et une opération d'écriture finale pour chaque pixel de l'image générée.

4 Organisation de l'image affichée

4.1 Coordonnées

- Les pixels des images ont des coordonnées entières.
- Les pixels des textures ont des coordonnées entières.
- Le coin en haut à gauche est de coordonnées (0,0).
- L'axe horizontal est orienté de gauche à droite.
- L'axe vertical est orienté de haut en bas.

4.2 Taille des textures et des images

Les images et textures ont des largeurs et hauteurs multiples de 32 pixels.

4.3 Traitement d'une scène

L'**image** affichée sur l'écran est appelée "**scène**". Une scène est obtenue en composant plusieurs "**objets**" affectés de différents attributs de transformation affine et de transparence.

Une scène est décomposée en "**tuiles**" de taille 32×32 pixels. L'accès aux tuiles d'une scène se fait par l'intermédiaire d'une liste châinée.

La scène est définie par une structure "**LGScene_t**" comportant les paramètres globaux de la scène :

Variable	Utilisation
p_liste_de_tuiles	Pointeur vers la liste des tuiles de la scène
p_image	Pointeur vers l'image à générer
largeur	Largeur de l'image (en pixels, multiple de 4) Une ligne de l'image est alignée aux frontières de mots
ng_fond	couleur du fond (niveau de gris)

Remarque : tous les pointeurs sont alignés sur des adresses de 32 bits.

Le traitement d'une scène consiste à enchainer le traitement de toutes les tuiles qui la composent.

4.4 Traitement d'une tuile

A chaque tuile de la scène est associée une liste de "**surfaces**". Une surface contient les informations nécessaires au traitement d'une texture faisant intersection avec la tuile concernée. Ainsi, tenant compte de la taille des textures composées, et de leur transformation géométrique, chaque tuile dispose d'une liste de surfaces pouvant comporter de zero (aucune texture n'intercepte la tuile) au nombre maximum de textures composant la scène.

Le traitement d'une tuile consiste d'abord à enchaîner le traitement de toutes les surfaces qui la composent, en respectant l'algorithme du peintre : la première surface traitée est la plus éloignée de l'observateur, la dernière surface traitée est la plus proche de l'observateur. Le contenu de la tuile est ensuite sauvé dans la scène.

Chaque tuile est définie par une structure "**LGTuile_t**" comportant les paramètres spécifiques de la tuile :

Variable	**Utilisation**
p_tuile_suivante	Pointeur vers la tuile suivante de la scène. Si ce pointeur est nul alors la tuile courante est la dernière tuile.
p_liste_de_surfaces	Pointeur vers la listes des surfaces concernées par la tuile
row	Numéro de ligne de la tuile (min = 0)
col	Numéro de colonne de la tuile (min = 0)

4.5 Traitement d'une surface

Pour chaque surface, le composeur doit :

1. Lire le fragment de la texture correspondante et stocker ce fragment de texture dans une "**mémoire de texture locale**". Cette mémoire locale doit pouvoir stocker un fragment de texture de 64×64 pixels
2. Parcourir la tuile en cours, ligne par ligne, colonne par colonne et déterminer les coordonnées correspondantes dans la texture locale via la transformée géométrique inverse correspondant à l'objet considéré.
3. Pour chacun des couples de coordonnées dans la texture locale récupérer le voisinage 2×2 du pixel correspondant et créer le pixel résultant par

interpolation bilinéaire, en tenant compte éventuellement du masque de la texture.

4. Composer ce pixel résultant avec le pixel correspondant dans une "**mémoire locale de tuile**", en tenant compte éventuellement de l'**opacité** du nouveau pixel.
5. Si la surface traitée est la dernière, sauvegarder la tuile dans l'image résultante.

Chaque surface est définie par une structure "**LGSurface_t**" comportant les paramètres spécifiques de la surface :

Variable	Utilisation
p_surface_suivante	Pointeur vers la surface suivante de la tuile. Si ce pointeur est nul alors la surface courante est la dernière surface.
p_obj	Pointeur vers l'objet concerné par la surface
p_attributs	Pointeur vers les attributs de l'objet
ox	abcisse dans la texture locale de la transformée inverse du point (0,0) de la tuile
oy	ordonnée dans la texture locale de la transformée inverse du point (0,0) de la tuile
x_texture	Abcisse dans la texture du coin haut/gauche de la boite entourante de la zone de texture à charger localement (unité : pixel)
y_texture	Ordonnée dans la texture du coin haut/gauche de la boite entourante de la zone de texture à charger localement (unité : pixel)
w_texture	Largeur de la boîte entourante dans la texture
h_texture	Hauteur de la boite enthourante dans la texture

Contraintes : Pour simplifier les calculs d'adresses et chargements, les paramètres x_texture, w_texture, y_texture, h_texture seront arrondis à des frontières de 4 pixels.

4.6 Objet

Un objet décrit une texture et ses caractéristiques. Deux types d'objets peuvent être créés :

1. Une image en niveaux de gris codés sur 8 bits (4 pixels par mots)
2. Une image en niveaux de gris codés sur 8 bits (4 pixels par mots, associée à un masque codé sur 1 bit (32 pixels par mots)

Chaque objet est défini par une structure "**LGObj_t**" comportant les paramètres spécifiques de l'objet :

Variable	**Utilisation**
p_texture	Pointeur vers la texture associée
p_masque	Pointeur vers le masque associé (si nul, pas de masque)
largeur	largeur de la texture (unités : pixels)
hauteur	hauteur de la texture (unités : pixels)

Contraintes : pour simplifier les traitements, les textures ont des largeurs multiples de 32 pixels. Ainsi la longueur d'une ligne de pixel ou d'une ligne de masque est un multiple entier de mots de la mémoire.

4.7 Attributs

Les attributs permettent de déterminer la transformée affine à réaliser, ainsi que les paramètres de composition (opacité, interpolation,...).

Les attributs d'un objet sont définis par une structure "**LGAttributs_t**" comportant les paramètres suivants :

Variable	Utilisation
x0	abcisse dans la texture globale de la transformée inverse du point (0,0) de l'image
y0	ordonnée dans la texture globale de la transformée inverse du point (0,0) de l'image
a_x	coefficient de la transformée inverse pour l'abcisse selon l'indice i
b_x	coefficient de la transformée inverse pour l'abcisse selon l'indice j
a_y	coefficient de la transformée inverse pour l'ordonnée selon l'indice i
b_y	coefficient de la transformée inverse pour l'ordonnée selon l'indice j
alpha	Opacité de l'objet
interpolation	Choix de l'interpolation (1 :interpolation 0 : sans interpolation)

L'usage et le calcul des coefficients sont détaillés dans les sections suivantes.

5 Calcul des attributs :

Cette section vise à expliquer la manière dont sont calculés les attributs de surfaces. Cette section n'est pas utile à la réalisation du composeur mais peut servir de référence pour modifier le logiciel.

D'un point de vue utilisateur, la transformation affine de référence est celle permettant de "placer" une texture dans l'image : L'espace de départ est la texture, l'espace d'arrivée est l'image. Si nous appelons (x_t, y_t) les coordonnées dans la texture et (x_i, y_i) les coordonnées dans l'image via la transformée affine f nous avons alors :

$$(x_i, y_i) = f(x_t, y_t)$$

Le composeur réalise un parcours régulier dans l'image, il accède donc aux pixels des textures sur la base de la transformée inverse f^{-1}de f :

$$(x_t, y_t) = f^{-1}(x_i, y_i)$$

La fonction f^{-1} étant une transformée affine, nous pouvons écrire :

$$\begin{aligned} x_t &= x_0 + a_x \cdot x_i + b_x \cdot y_i \\ y_t &= y_0 + a_y \cdot x_i + b_y \cdot y_i \end{aligned}$$

Nous traitons l'image tuile par tuile, les tuiles sont de taille 32×32. Nous pouvons transposer la fonction f^{-1} dans le cadre d'une tuile de rangée "row" et de colonne "col". En appellant (i, j) les coordonnées dans la tuile nous obtenons :

$$\begin{aligned} x_t &= (x_0 + 32 \cdot (a_x \cdot \text{col} + b_x \cdot \text{row})) + a_x \cdot i + b_x \cdot j \\ y_t &= (y_0 + 32 \cdot (a_y \cdot \text{col} + b_y \cdot \text{row})) + a_y \cdot i + b_y \cdot j \end{aligned}$$

Le composeur ne peut pas accéder directement à toute la texture, mais simplement à une portion de celle ci placée dans une mémoire locale de 32x32 pixels. Il faut déterminer les coordonnées de la zone à charger. Pour cela on calcule les antécédants par la fonction f^{-1} des quatre sommets de la tuile, et on détermine les coordonnées de la boite entourante. Les valeurs obtenues permettent de déterminer les paramètres x_texture, y_texture, w_texture et h_texture de la surface (voir section 3.4).

Le point de coordonnée (x_texture, y_texture) dans la texture se retrouve en position $(0, 0)$ dans la mémoire locale 32×32. Dans l'espace de cette mémoire locale , la transformation f^{-1} devient la transformation inverse locale f_l^{-1} :

$$(x_{\text{tl}}, y_{\text{tl}}) = f^{-1}(i, j)$$

avec :

$$\begin{aligned} x_{\text{tl}} &= (x_0 - x_{\text{texture}} + 32 \cdot (a_x \cdot \text{col} + b_x \cdot \text{row})) + a_x \cdot i + b_x \cdot j \\ y_{\text{tl}} &= (y_0 - y_{\text{texture}} + 32 \cdot (a_y \cdot \text{col} + b_y \cdot \text{row})) + a_y \cdot i + b_y \cdot j \end{aligned}$$

En conclusion, les attributs définis en 3.6 sont :

$$\begin{aligned} \text{ox} &= x_0 - x_{\text{texture}} + 32 \cdot (a_x \cdot \text{col} + b_x \cdot \text{row}) \\ \text{oy} &= y_0 - y_{\text{texture}} + 32 \cdot (a_y \cdot \text{col} + b_y \cdot \text{row}) \end{aligned}$$

6 Composition des pixels dans une tuile

6.1 Interpolation

Le composeur dispose d'une mémoire de tuile locale 32×32. Pour un point de coordonnée (i,j) on récupère la coordonnée antécédante $(x_{\text{tl}}, y_{\text{tl}})$. Cette co-

ordonnée n'est pas entière. La partie fractionnaire permet de calculer un pixel résultant P_r par interpolation bilinéaire entre les 4 pixels voisins de (x_{tl}, y_{tl}) dans la mémoire locale de texture. Si la texture a un masque binaire associé, alors on interpole aussi le masque binaire pour obtenir une opacité O_r propre au pixel P_r.

6.2 « Clipping »

Au niveau « local »il n'est pas possible de déterminer si le pixel calculé est en dehors des limites de la texture. Il faut, parallèlement au calcul local, réaliser un calcul de position global et forcer le pixel à être totalement transparent.

6.3 Composition

Enfin, si on appelle P_c la valeur du pixel courant à la coordonnée (i, j), et P_f la valeur finale du pixel on aura :

$$P_f = O_r \cdot \text{alpha} \cdot P_r + (1 - O_r \cdot \text{alpha}) \cdot P_c$$

Le paramètre alpha est l'opacité générale de l'objet considéré. Les valeurs d'opacité sont considérées comme étant en virgule fixe avec un maximum égal à 1. Enfin, si le paramètre d'interpolation de l'objet est zero alors P_r est obtenu en cherchant le pixel le plus proche (arrondi des coordonnées) et O_r est l'opacité de ce pixel (0 ou 1).