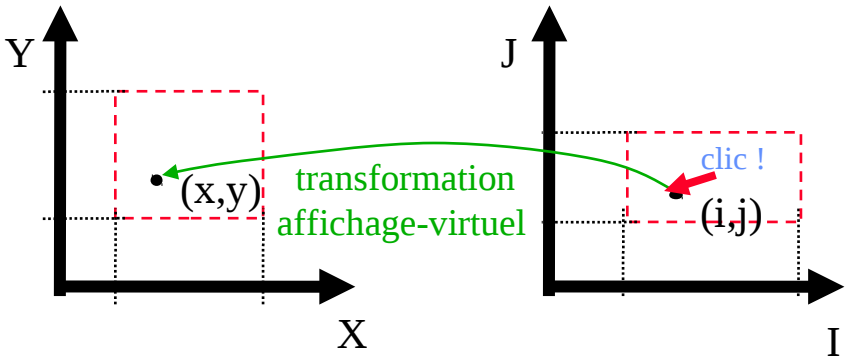
Sélection graphique

Sélection 2D

Rappel

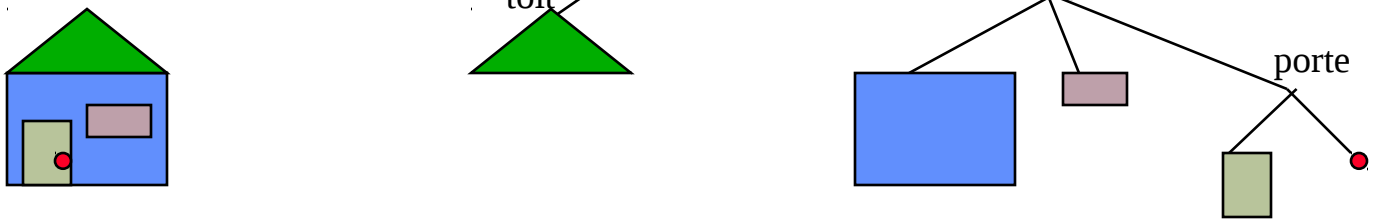


Sélection graphique d'un groupe d'objets: en général, une nouvelle sélection élimine l'ancienne. On procède donc:

- par une syntaxe différente
  - p.e. un autre bouton de souris, une touche de modification (« shift »)
- à l'aide d'un rectangle élastique
  - il faut alors différencier une simple sélection d'un début de rectangle
- à l'aide d'un « lasso » (traçage d'une ligne libre autour des objets désirés)
  - plus coûteux de déterminer quels objets ont été sélectionnés

Sélection 2D

Une fois qu'on a sélectionné plusieurs objets, on peut en faire un groupe et ainsi créer toute une hiérarchie d'objets



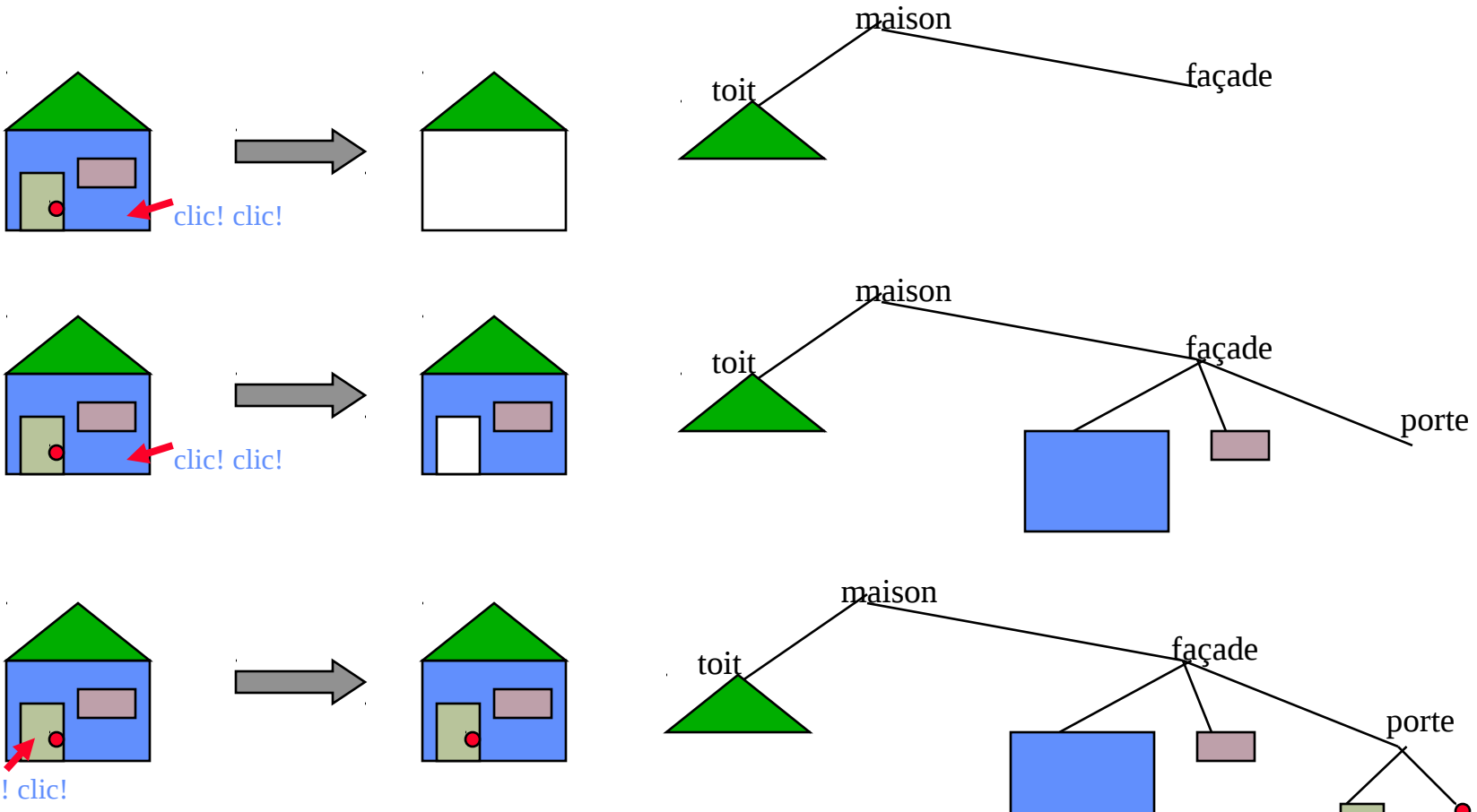
Comment sélectionner une composante d'un objet hiérarchique?

- p.e. la poignée de porte

Regardons deux stratégies possibles:

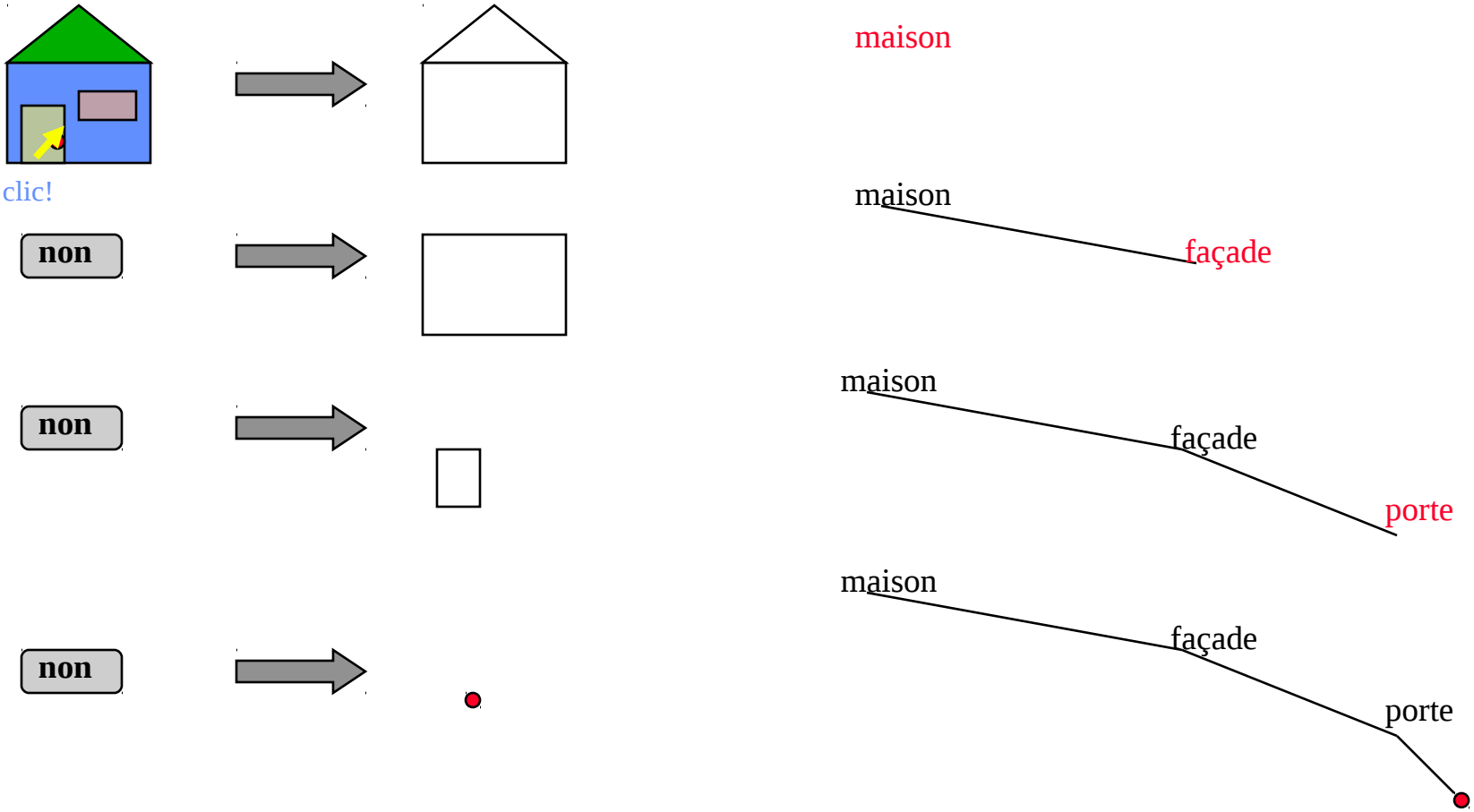
Sélection 2D

a) des sélections successives *ouvrent* des groupes d'une hiérarchie



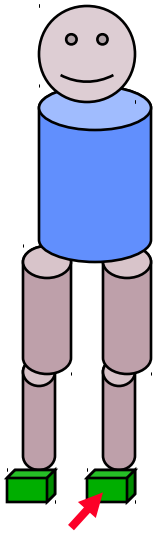
Sélection 2D

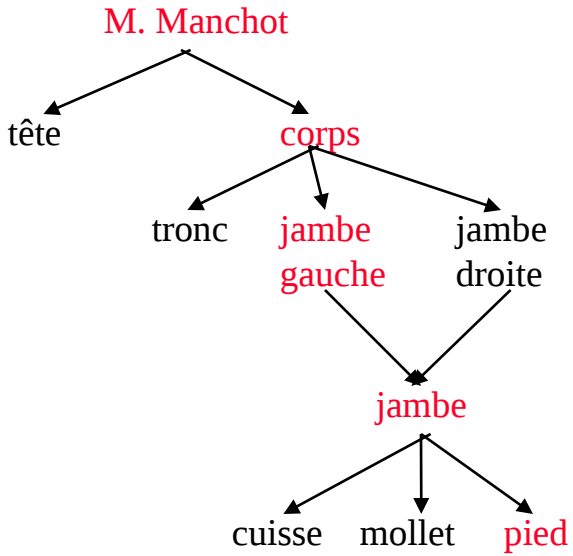
b) une *touche de rejet* permet d'en arriver à l'objet souhaité



Sélection 2D

Si notre hiérarchie est un graphe orienté sans cycle (« DAG »), par exemple pour partager des groupes d'objets, alors tout le chemin dans le graphe est important pour identifier l'objet sélectionné: un nœud seul n'est pas suffisant





Sélection 3D

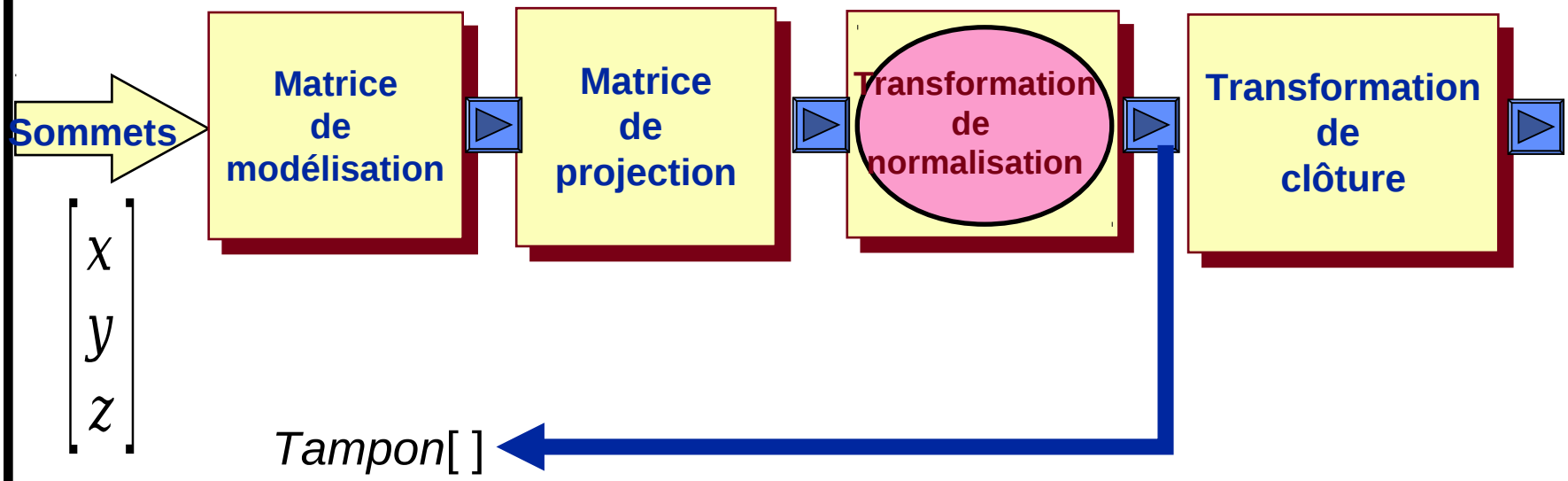
En 3D, les choses se compliquent un peu:

- les transformations sont plus complexes qu'en 2D
- la projection sur un plan avant (2D) signifie qu'une certaine position à l'écran peut correspondre à plusieurs objets à des profondeurs différentes

On peut considérer la position du pointeur à l'écran comme un rayon qui « *transperce* » les objets de la scène graphique:

- déterminer les objets transpercés
- sélectionner l'objet le plus rapproché de l'observateur (la plus faible profondeur)

Sélection 3D -- OpenGL



En mode sélection, OpenGL remplit le tampon de sélection.

Sélection 3D -- OpenGL

- OpenGL utilise une pile de noms pour identifier les objets.

- On met des noms sur la pile tout en traçant des objets;
- le contenu de la pile change donc en fonction des objets qu'on trace.

- À chaque fois qu'OpenGL « *voit* » un objet dans le volume de visualisation, *tout* le contenu courant de la pile est copié dans le tampon de sélection;
- c'est en examinant le tampon de sélection à la fin du processus que nous pouvons connaître les objets sélectionnés.

Sélection 3D -- OpenGL

Pour sélectionner *tout ce qui est visible* dans la fenêtre, il faut:

- spécifier un tampon de sélection qui servira à accumuler de l'information sur les objets transpercés
- se mettre en mode de sélection
- initialiser la pile des noms (qui seront donnés aux objets de la scène)
- « dessiner » les objets et leur attribuer des noms qu'on mettra sur la pile des noms
- sortir du mode de sélection (revenir en mode de rendu graphique)
- examiner le tampon de sélection et déterminer l'objet sélectionné

Sélection 3D -- OpenGL

Pour sélectionner *un objet*, il faut:

- spécifier un tampon de sélection qui servira à accumuler de l'information sur les objets transpercés
- se mettre en mode de sélection
  - *définir le rayon de sélection (typiquement selon la position du pointeur)*
- initialiser la pile des noms (qui seront donnés aux objets de la scène)
- « dessiner » les objets et leur attribuer des noms qu'on mettra sur la pile des noms
- sortir du mode de sélection (revenir en mode de rendu graphique)
- examiner le tampon de sélection et déterminer l'objet sélectionné

Sélection 3D -- OpenGL
tampon de sélection

```
void glSelectBuffer( GLsizei taille, GLuint *tampon )
```

où

`taille` est le nombre maximum d'éléments dans le tampon

Ce qu'on y retrouve, pour chaque objet sélectionné/transpercé:

- Le nombre d'identificateurs sur la pile des noms au moment où l'objet est transpercé
- Les valeurs minimum et maximum de la coordonnée z des sommets des primitives formant l'objet (GLint)
- Le ou les identificateurs de désignation (noms), en commençant par le dessous de la pile

Sélection 3D -- OpenGL
mode de sélection

```
GLint glRenderMode( GLenum mode )
```

où

`mode` est l'un de `GL_RENDER, GL_SELECT, GL_FEEDBACK`

En mode `GL_SELECT`, les calculs de rendu graphique sont effectués mais on ne dessine pas à l'écran (la mémoire de trame reste inchangée)

Lorsqu'on revient en mode `GL_RENDER` après avoir été en mode `GL_SELECT`, la valeur de retour indique le nombre d'objets sélectionnés

```
glRenderMode(GL_RENDER);
// afficher
glRenderMode(GL_SELECT);
// afficher pour sélection
nb_objets = glRenderMode(GL_RENDER);
```

Sélection 3D -- OpenGL
rayon de sélection

```
void gluPickMatrix( GLdouble x, GLdouble y,
      GLdouble delX, GLdouble delY, Glint *viewport )
```

où

`x,y` sont les coordonnées du centre du rayon (position du pointeur)

`delX,delY` sont la largeur et hauteur du rayon (delta, précision)

`viewport` donne les limites de la clôture obtenues par

```
glGetIntegerv( GL_VIEWPORT, &viewport );
```

```
glMatrixMode(GL_PROJECTION); glPushMatrix();
glLoadIdentity();
gluPickMatrix ( .......);
gluPerspective, glOrtho ou glFrustum
glMatrixMode(GL_MODELVIEW);
// afficher pour sélection
glMatrixMode(GL_PROJECTION); glPopMatrix();
```

Sélection 3D -- OpenGL
rayon de sélection

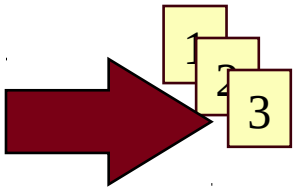
La fonction `gluPickMatrix` modifie le volume de projection et affecte la matrice GL_PROJECTION.

- On doit `gluPickMatrix` **avant** de définir notre projection.
- Tous les arguments de `gluPickMatrix` sont en coordonnées d'écran (en pixels).
- On doit donc fournir la clôture courante afin qu'OpenGL puisse faire la transformation inverse et convertir les valeurs `(x,y)` et `(delX,delY)` en coordonnées normalisées.

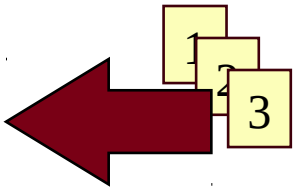
```
glMatrixMode(GL_PROJECTION); glPushMatrix();
glLoadIdentity();
gluPickMatrix ( .......);
gluPerspective, glOrtho ou glFrustum
glMatrixMode(GL_MODELVIEW);
// afficher pour sélection
glMatrixMode(GL_PROJECTION); glPopMatrix();
```

Sélection 3D -- OpenGL
pile des noms (identificateurs de désignation)

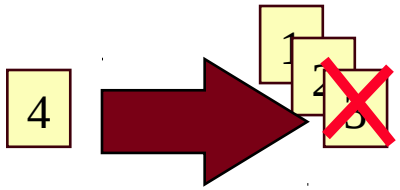
```
void glInitNames( void );
void glPushName( GLuint nom );
```




```
void glPopName( void );
```



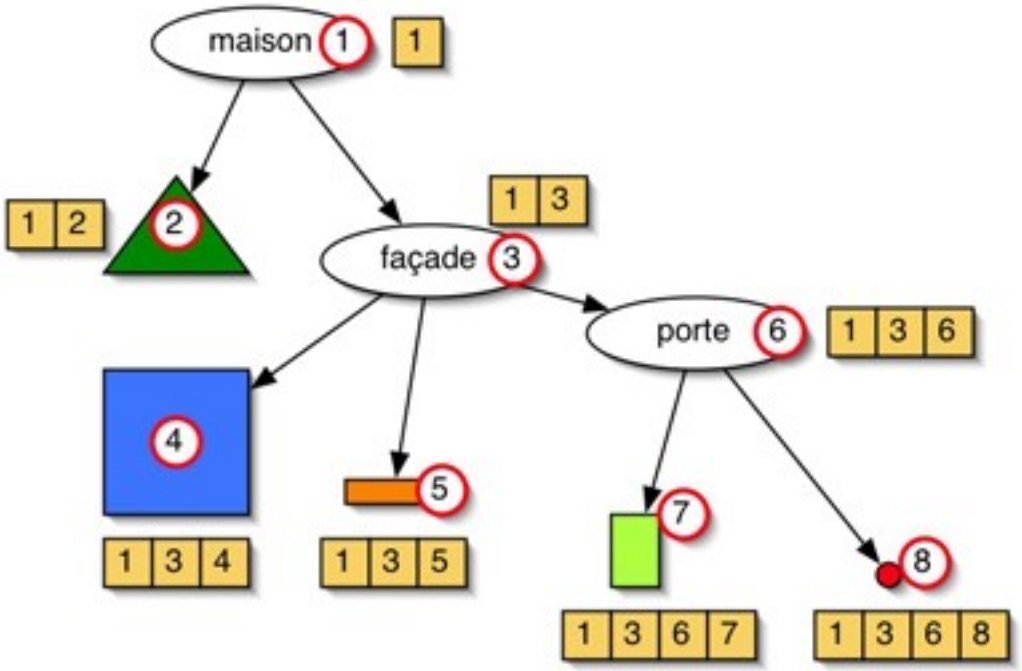

```
void glLoadName( GLuint nom );
```



Note: ces appels sont ignorés si on est pas en mode GL_SELECT

Sélection 3D -- OpenGL
contenu du tampon de sélection

Exemple des noms en suivant l'arbre:



Sélection 3D -- OpenGL
contenu du tampon de sélection

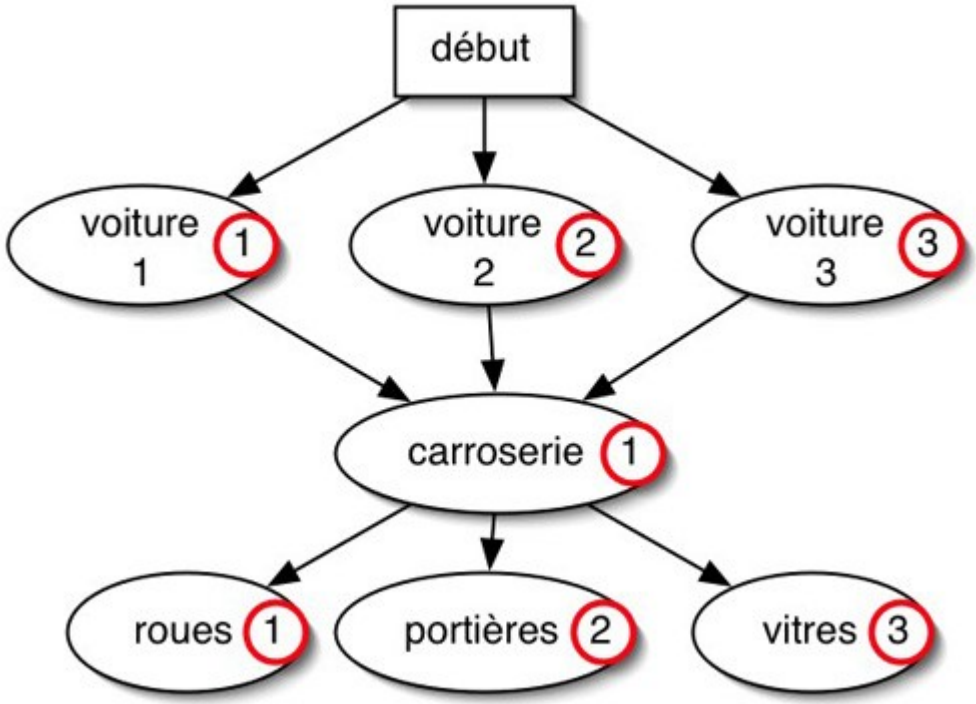
Exemple :

Contenu de la pile si une vitre 3 de la carrosserie 1 de la voiture 1 est sélectionnée:

1	1	3

Si c'est la carrosserie 1 de la voiture 3:

3	1



Sélection 3D -- OpenGL utilisation de la profondeur

On met un ou plusieurs identificateurs sur la pile des noms, on dessine l'objet, on retire/remplace possiblement des identificateurs de la pile et puis on recommence...

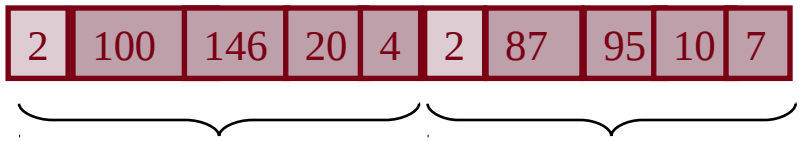
```
glInitNames();
...
glPushName(10);  // on dessine des micros, identifié par 10
glPushName(1);   // premier micro...
TracerMicroUn(); // identifié par 1
glLoadName(2);   // second micro...
TracerMicroDeux(); // identifié par 2
...
glPopName();     // le no de micro particulier
glPopName();     // l'identificateur des micros
glPushName(20);  // on dessine des musiciens, identifié par 20
...
```

Sélection 3D -- OpenGL
examen du tampon de sélection

Supposons que deux objets, un musicien et un micro, ont été transpercés :

- l 'appel `glRenderMode(GL_RENDER)` a donc retourné « 2 » puisque deux objets ont été vus.

- le tampon de sélection pourrait avoir le contenu suivant:



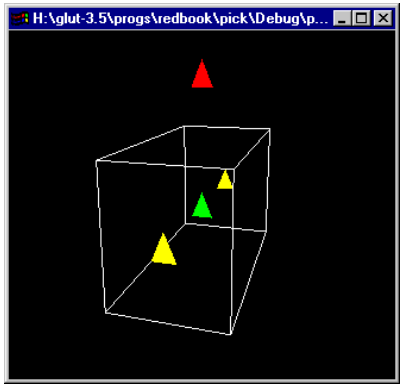
le 4e musicien, de profondeur 100 à 146

le 7e micro, de profondeur 87 à 95

- Le micro est plus en avant → on sélectionne le micro...

Sélection 3D -- OpenGL
examen du tampon de sélection

Autre exemple :



Supposons

3 identificateurs (noms) pour le premier objet

1 identificateur (nom) pour le deuxième objet

2 identificateurs (noms) pour le troisième objet

```
NbChoix = glRenderMode( GL_RENDER );
```
// NbChoix = 3

3	zmin	zmax	nom1	nom2	nom3
1	zmin	zmax	nom1		
2	zmin	zmax	nom1	nom2	

3	zmin	zmax	nom1	nom2	nom3	1	zmin	zmax	nom1	2	zmin	zmax	nom1	nom2

Sélection 3D -- OpenGL
examen du tampon de sélection

```
        glMatrixMode( GL_PROJECTION );
        glPushMatrix();
        glLoadIdentity();
        gluPickMatrix( x, viewport[3]-y, 3, 3, viewport );
        glOrtho( -8.0, 8.0, -8.0, 8.0, 1.0, 10.0 );

        // Informer de la zone tampon
        const GLsizei taille = 200;
        GLuint tampon[taille];
        glSelectBuffer( taille, tampon );

        // Faire la sélection
        glRenderMode( GL_SELECT );
        affichage();
        int nbObjets = glRenderMode( GL_RENDER );
...
```

Sélection 3D -- OpenGL
examen du tampon de sélection

```
// Imprimer le résultat de la sélection
GLuint* ptr = tampon;
cout << endl << "--- Nombre d'objets: " << nbObjets << endl;
for( int iobj = 0; iobj < nbObjets; ++iobj )
{
   GLuint nbnoms = *ptr; ++ptr;
   cout << "Nombre de noms: " << nbnoms << endl;
   cout << "Zmin = " << (float) *ptr/0xffffffff << endl; ++ptr;
   cout << "Zmax = " << (float) *ptr/0xffffffff << endl; ++ptr;
   cout << "Noms: ";
   for( unsigned int inom = 0; inom < nbnoms; ++inom )
   {
      if ( *ptr == ListSphere ) cout << "Sphere ";
      else if ( *ptr == ListCube ) cout << "Cube ";
      ++ptr;
   }
   cout << endl;
}
...
```

Sélection 3D -- OpenGL
examen du tampon de sélection

```
// Remettre les valeurs initiales
glMatrixMode( GL_PROJECTION );
glPopMatrix();
glMatrixMode( GL_MODELVIEW );
```

----------Nombre d'objets: 2
Nombre de noms: 1
Zmin = 0.182276
Zmax = 0.59095
Noms: Sphere
Nombre de noms: 1
Zmin = 0.397765
Zmax = 0.791124
Noms: Cube

(Exemple Selection)

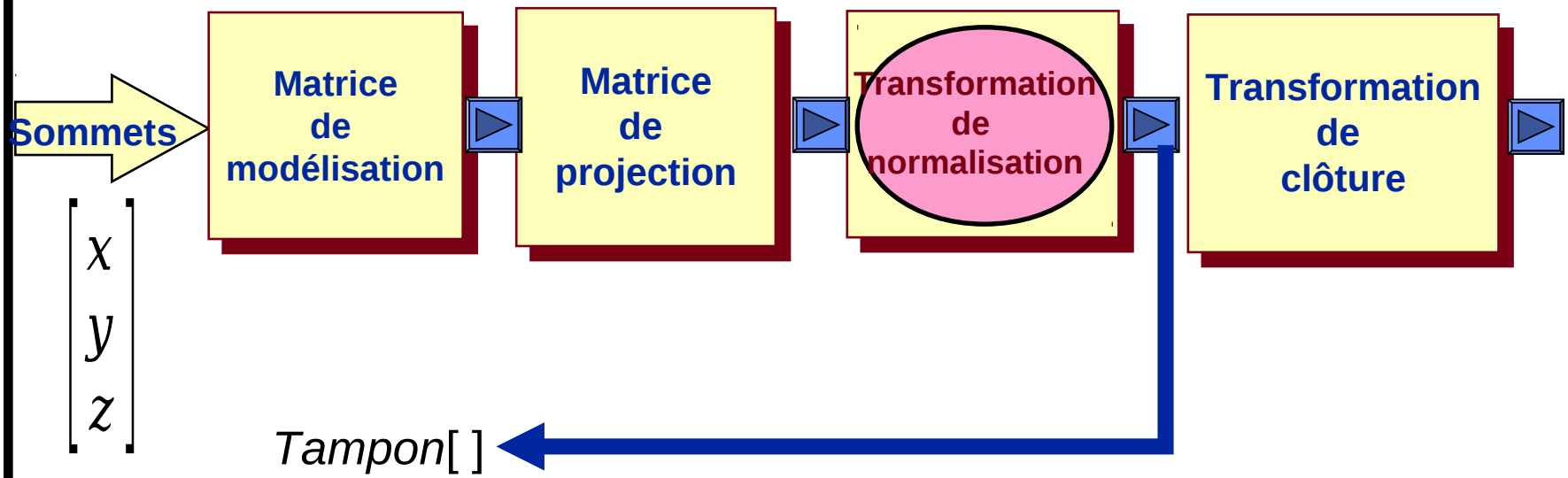
Rétroaction -- OpenGL

OpenGL peut aussi retourner des informations sur ce qui serait dessiné à l'écran, c'est-à-dire tout ce qui est visible dans le volume de visualisation.

- Exemples d'utilisation :
  - Sauvegarde en mode vectoriel (InkScape, Adobe/Illustrator)
  - Sauvegarde ou impression d'un fichier PostScript
  - Exporter vers un format de fichier 3D
  - ...

- Les étapes du mode *rétroaction (feedback)* sont très semblables à celles du mode *sélection.*

Rétroaction -- OpenGL



En mode *feedback*, OpenGL remplit le tampon de retour d'information.

Rétroaction -- OpenGL

Pour obtenir un *retour d'information*, il faut:

- spécifier un tampon de retour qui servira à recevoir les données sur les objets dessinés
- se mettre en mode de rétroaction
- « dessiner » les objets
- *Éventuellement, faire des appels à glPassThrough() pour insérer des marqueurs dans le tampon de retour*
- sortir du mode de retour (revenir en mode de rendu graphique)
- examiner le tampon de retour

Rétroaction -- OpenGL

```
void glFeedbackBuffer( GLsizei taille,
                       GLenum type, GLfloat *tampon )
```

où

`taille` est le nombre maximum d'éléments dans le tampon

`type` est une constante symbolique : `GL_2D`, `GL_3D`, `GL_3D_COLOR`, `GL_3D_COLOR_TEXTURE`, `GL_4D_COLOR_TEXTURE`

- Chaque primitive visible génère, selon `type`, un bloc de valeurs (sommets, couleur, texture) qui sont copiées dans le tampon
- Chaque bloc commence par un code indiquant le type de primitive, suivi par les sommets et les autres données associées
- Les primitives ont été découpées par Cohen-Sutherland
- Les polygones ou quadrilatères peuvent avoir été découpés en triangles, ce qui est souvent utile.

Rétroaction -- OpenGL

```
GLint glRenderMode( GLenum mode )
```

où

`mode` est l'un de `GL_RENDER, GL_SELECT, GL_FEEDBACK`

En mode `GL_FEEDBACK`, les calculs de rendu graphique sont effectués mais on ne dessine pas à l'écran (la mémoire de trame reste inchangée).

Lorsqu'on revient en mode `GL_RENDER` après avoir été en mode `GL_FEEDBACK`, la valeur de retour indique le nombre de valeurs stockées dans le tampon.

```
glRenderMode(GL_RENDER);
// dessiner
glRenderMode(GL_FEEDBACK);
// obtenir le nombre de valeurs
nb_valeurs = glRenderMode(GL_RENDER);
```

Rétroaction -- OpenGL

```
// Informer de la zone tampon
const GLsizei taille = 5000;
GLfloat tampon[taille];
glFeedbackBuffer( taille, GL_3D_COLOR, tampon );

// Faire l'affichage et obtenir un retour
glRenderMode( GL_FEEDBACK );
affichage();
int nbValeurs = glRenderMode( GL_RENDER );

// Examiner le tampon de retour
...
```

Rétroaction -- OpenGL

```
GLint count = taille;
GLfloat token;
while (count) {
   token = buffer[size-count]; count--;
   if (token == GL_PASS_THROUGH_TOKEN) {
      printf( "GL_PASS_THROUGH_TOKEN\n" );
      printf("  %4.2f\n", buffer[size-count]);
      count--;
   }
   else if (token == GL_POINT_TOKEN) {
      printf( "GL_POINT_TOKEN\n" );
      print3DcolorVertex( size, &count, buffer );
   }
   else if (token == GL_LINE_TOKEN) {
      printf( "GL_LINE_TOKEN\n" );
      print3DcolorVertex( size, &count, buffer );
      print3DcolorVertex( size, &count, buffer );
   }
   ...
```

(Voir exemple rétroaction)

Rétroaction -- OpenGL

```
GL_LINE_RESET_TOKEN
  30.00 30.00 0.00 0.84 0.84 0.84 1.00
  50.00 60.00 0.00 0.84 0.84 0.84 1.00
GL_LINE_TOKEN
  50.00 60.00 0.00 0.84 0.84 0.84 1.00
  70.00 40.00 0.00 0.84 0.84 0.84 1.00
GL_PASS_THROUGH_TOKEN
  1.00
GL_PASS_THROUGH_TOKEN
  2.00
GL_POINT_TOKEN
  50.00 50.00 0.00 0.84 0.84 0.84 1.00
GL_LINE_RESET_TOKEN
  30.00 30.00 0.00 0.84 0.84 0.84 1.00
  50.00 60.00 0.00 0.84 0.84 0.84 1.00
GL_LINE_TOKEN
  50.00 60.00 0.00 0.84 0.84 0.84 1.00
  70.00 40.00 0.00 0.84 0.84 0.84 1.00
GL_PASS_THROUGH_TOKEN
  1.00
GL_PASS_THROUGH_TOKEN
  2.00
GL_POINT_TOKEN
  50.00 50.00 0.00 0.84 0.84 0.84 1.00
```