LISTES D'AFFICHAGE, GROUPES D'ATTRIBUTS, HIÉRARCHIE D'OBJETS

Listes d'affichage

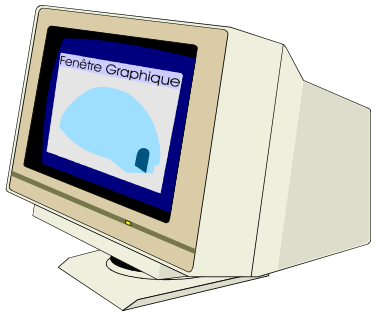
- Une liste d'affichage est un groupe de commandes OpenGL qui peuvent être mémorisées pour une exécution ultérieure.
- L'utilisation d'une liste d'affichage permet:
  - Une bonne organisation des données;
  - Une amélioration de la performance
    - définition d'une géométrie qui a besoin d'être affichée à plusieurs reprises dans un programme;
    - définition d'une série de transformations (déplacements, mise à l'échelle, etc.) qui ont besoin d'être appliquées à plusieurs reprises;
    - dans un modèle client-serveur les listes d 'affichage sont gérées par le serveur d'où une réduction du coût de transmission des données à travers le réseau;
    - une liste d'affichage peut être mémorisée dans une mémoire dédiée ou sous forme optimisée plus compatible avec le matériel graphique.
  - un *bémol* depuis l'avènement des VBO et des nuanceurs qui permettent de mieux envoyer des coordonnées et des énoncés à la carte graphique.

Listes d'affichage : fonctionnement

- Tout peut être affiché en mode immédiat
- Mémoriser les commandes pour une exécution ultérieure

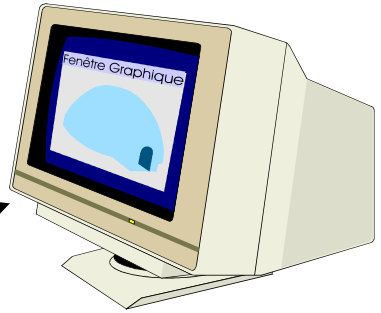
- Mode Immédiat

Appel à OpenGL → Pipeline graphique →



- Mode Liste d'affichage (différé)

Appel à OpenGL → Mémoire → Mémoire → Pipeline graphique →



Listes d'affichage : création

- Début de la liste d'affichage

  void glNewList(GLuint list, GLenum mode);

  list entier positif

  mode GL_COMPILE, GL_COMPILE_AND_EXECUTE

- Fin de la liste d'affichage

  void glEndList(void);

- Exemple

```
GLfloat color_vector[3] = { 0.0, 0.0, 0.0  };
glNewList( 1, GL_COMPILE );
    glColor3fv( color_vector );
glEndlist();
color_vector[0]=1.0;   // rouge
```

Listes d'affichage : générer un nom

- Exécuter une liste d'affichage
  void glCallList (GLuint list);
  list entier positif

- Générer un nom
  GLuint glGenLists (GLsizei range);
  range nombre de noms de liste à générer (consécutif)

- Détruire une ou plusieurs listes
  void glDeleteLists (GLuint list, GLsizei range);
  list entier positif
  range nombre de noms de liste à détruire (consécutif)

Listes d'affichage : plusieurs listes

- Vérifier l'existence d'une liste

  GLboolean glIsList(GLuint list);

  retourne *TRUE* si la liste existe déjà

- Exécuter plusieurs listes

  *void glCallLists(GLsizei n, GLenum type, const GLvoid *lists);*

  n nombre de listes

  type GL_BYTE, GL_UNSIGNED_BYTE, GL_SHORT, GL_UNSIGNED_SHORT, GL_INT, GL_UNSIGNED_INT, GL_FLOAT, GL_2_BYTES, GL_3_BYTES, GL_4_BYTES.

  lists tableau de numéro de listes d'affichage

- Spécifier un décalage à ajouter à glCallLists()

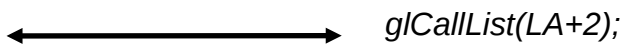
  void glListBase(Gluint base);

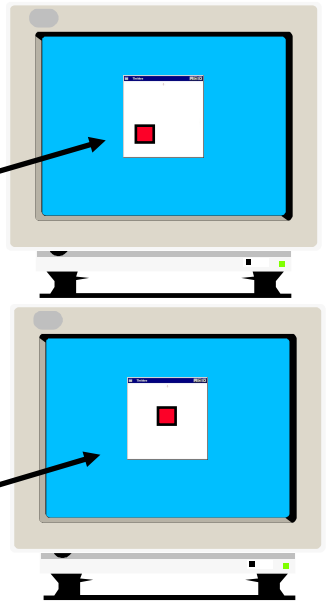
  base entier positif

« Édition » de listes d'affichage

```
GLuint LA = glGenLists( 3 );
glNewList( LA+0, GL_COMPILE );
   glColor3f( 1.0, 0.0, 0.0 );
glEndList();

glNewList( LA+1, GL_COMPILE );
   glRectf( 0.0, 0.0, 2.0, 2.0 );
glEndList();

glNewList( LA+2, GL_COMPILE );
   glCallList(LA+0);
   glCallList(LA+1);
glEndList();
```


```
glDeleteLists(LA+1, 1); // pas nécessaire !

glNewList( LA+1, GL_COMPILE );
   glRectf( 2.0, 2.0, 4.0, 4.0 );
glEndList();
```





Listes d'affichage : opérations

- Opérations qui peuvent être effectuées facilement
  - générer des noms pour les listes d'affichage
  - créer une nouvelle liste d'affichage
  - faire appel (exécuter) à une ou plusieurs listes d'affichage
  - organisation hiérarchique de listes d'affichage
- Opérations qui ne peuvent être effectuées facilement
  - éditer une liste d'affichage
  - écrire ou lire le contenu d'une liste d'affichage dans un fichier
  - ne peut mettre des fonctions d'interrogation ou des requêtes de client dans une liste d'affichage

Commandes d'OpenGL non permises dans une liste d'affichage

- glDeleteLists()
- glFeedbackBuffer()
- glFinish()
- glFlush
- glGenLists()
- glGet*()

… et quelques autres fonctions

- glIsEnabled
- glIsList()
- glPixelStore
- glReadPixels()
- glRenderMode()
- glSelectBuffer()

Affichage de Texte

- spécification de la police
  - famille: Courier Avant Garde Times
  - style: Times **Times gras** *Times italique*

standard
- taille: Times 18 points Times 10 points Times 32 points
- autre information, moins standard...

Affichage de Texte

- `glRasterPos{2,3,4}{s,i,f,d}[v](x,y,z,w)`
  - indique les coordonnées pour dessiner le caractère (« bitmap »)
  - ces coordonnées sont transformées en une position à l'écran
- `glBitMap(largeur,hauteur,xo,yo,xi,yi,bitmap)`
  - *largeur,hauteur* : taille du « bitmap », en pixels
  - *xo,yo* : origine du « bitmap » devant coïncider avec la position spécifiée
  - *xi,yi* : incréments de la position courante après l'affichage de ce « bitmap »
  - *bitmap* : pointeur au « bitmap »
- Exemple:

```
Glubyte Fmaj[24] =
{0xc0,0x00,0xc0,0x00,0xc0,0x00,0xc0,0x00,0xc0,0x00,0xff,0x00,0x
ff,0x00,0xc0,0x00,0xc0,0x00,0xc0,0x00,0xff,0xc0,0xff,0xc0};
...
glRasterPos2i(20,20);
glBitmap(10,12,0.0,0.0,11.0,0.0,Fmaj);
```

ATTRIBUTS DES PRIMITIVES

- Groupes d'attributs (environ 30 groupes différents)
  - Lignes (Variables d'état) : GL_LINE_BIT
    - Épaisseur du trait
    - Attribut d'état d'activation du type de ligne (GL_LINE_STIPPLE)
    - Patron du type de la ligne
    - Le facteur de répétition du patron du type de la ligne
    - Attribut d'état d'activation du lissage de la ligne (GL_LINE_SMOOTH)
- Pile pour conserver les attributs (capacité de 16 groupes)
- Déposer sur la pile

  glPushAttrib (GLbitfield mask);

  mask indique le groupe d'attributs

- Retirer de la pile

  glPopAttrib(void);

GROUPES D'ATTRIBUTS

GL_ACCUM_BUFFER_BIT	Tampon d`accumulation
GL_ALL_ATTRIB_BITS	Tous
GL_COLOR_BUFFER_BIT	Tampon de couleurs
GL_CURRENT_BIT	Courant
GL_DEPTH_BUFFER_BIT	Tampon de profondeur
GL_ENABLE_BIT	Activation
GL_EVAL_BIT	Évaluation
GL_FOG_BIT	Brouillard
GL_HINT_BIT	Suggestion
GL_LIGHTING_BIT	Éclairage
GL_LINE_BIT	Ligne
GL_LIST_BIT	Liste
GL_PIXEL_MODE_BIT	Pixel
GL_POINT_BIT	Point
GL_POLYGON_BIT	Polygone
GL_POLYGON_STIPPLE_BIT	Patron d'hachurage des polygones

PRÉSERVER LES ATTRIBUTS DANS UNE LISTE D'AFFICHAGE

```
void ConstruireCylindre (void )
{
   LA_CylindreFilFer = glGenLists( 1 );
   glNewList( LA_CylindreFilFer, GL_COMPILE_AND_EXECUTE );
   glPushAttrib( GL_CURRENT_BIT  | GL_COLOR_BUFFER_BIT );
   glColor3f( 1.0, 0.5, 0.5 );
   CreerCylindre( FIL_DE_FER );
   glPopAttrib();
   glEndList();

   LA_CylindrePlein = glGenLists( 1 );
   glNewList( LA_CylindrePlein, GL_COMPILE_AND_EXECUTE );
   glPushAttrib( GL_CURRENT_BIT | GL_COLOR_BUFFER_BIT );
   glColor3f( 0.6, 0.3, 0.3 );
   CreerCylindre( PLEIN );
   glColor3f( 1.0, 0.5, 0.5 );
   CreerCylindre( FIL_DE_FER );
   glPopAttrib();
   glEndList();
}
```

EXÉCUTER UNE LISTE D'AFFICHAGE

```
void AfficherCylindre( GLenum Plein )
{
   if ( Plein )
     glCallList( LA_CylindrePlein );
   else
     glCallList( LA_CylindreFilFer );
}
```

CRÉER UN CYLINDRE (glu)

```
void AfficherCylindre ( GLenum Plein )
{
   GLdouble rayon = 0.7;
   GLdouble hauteur = 4.0;
   GLUquadricObj *qobj = gluNewQuadric( );

   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

   if ( Axes ) TracerAxes3D( &Origine, 1.4 );

   glColor3f( MidnightBlue );
   glPushMatrix( );

   glTranslated( 0.0, 0.0, -2.0 );
   gluQuadricDrawStyle( qobj, Plein ? GL_FILL :
   GLU_SILHOUETTE );
   gluCylinder( qobj, rayon, rayon, hauteur, 10, 8 );

   glPopMatrix( );
   gluDeleteQuadric( qobj );
   glutSwapBuffers();
}
```

CRÉER UN CYLINDRE PLEIN (glu)

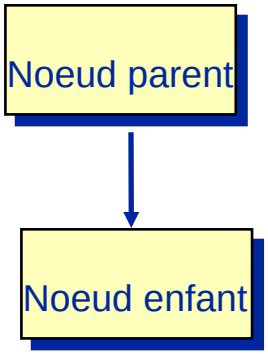
```
void ObjConstruireCylindre( void )
{
   struct
   {
     GLdouble rayon;
     GLdouble hauteur;
     GLint Pointes;
     GLint Anneaux;
   } Cyl = {0.5, 1.0, 10, 8};

   GLUquadricObj * qobjCyl = gluNewQuadric();
   GLUquadricObj * qobjBas = gluNewQuadric();
   GLUquadricObj * qobjHaut = gluNewQuadric();
```

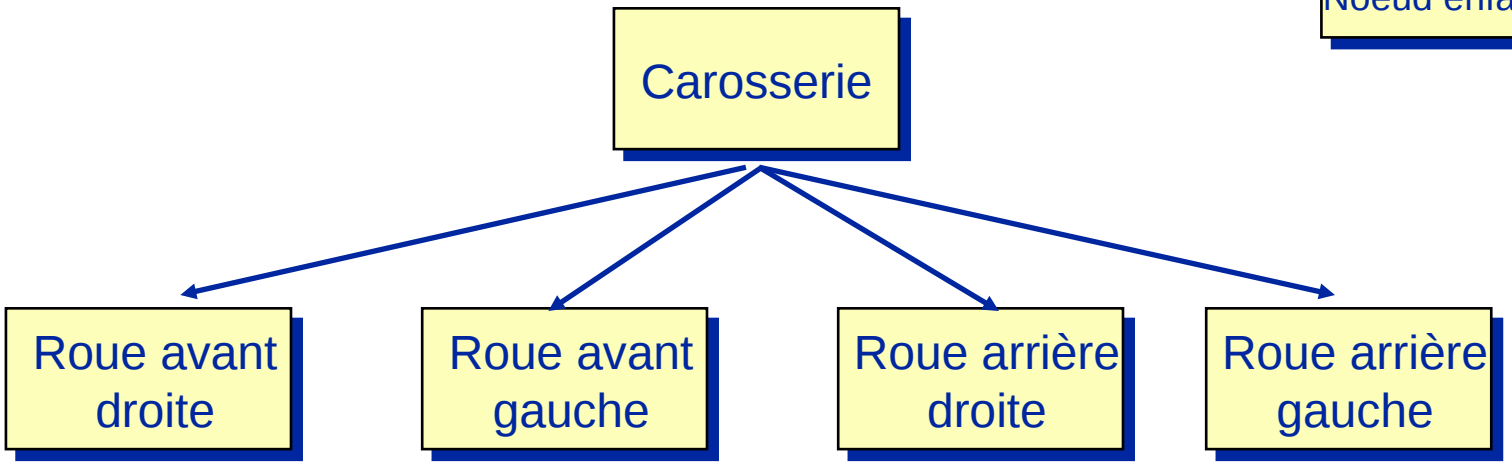
CRÉER UN CYLINDRE (glu)

```
    LA_Cylindre = glGenLists( 1 );
    glNewList( LA_Cylindre, GL_COMPILE );
    glPushAttrib( GL_ALL_ATTRIB_BITS );
    gluQuadricDrawStyle( qobjCyl, GL_FILL );
    gluQuadricOrientation( qobjCyl, GLU_OUTSIDE );

    glPushMatrix();
    glRotatef( -90.0, 1.0, 0.0, 0.0 );
    gluCylinder( qobjCyl, Cyl.rayon, Cyl.rayon,
                 Cyl.hauteur, Cyl.Pointes, Cyl.Anneaux );
    gluDisk( qobjBas, 0.0, Cyl.rayon, Cyl.Pointes, Cyl.Anneaux );
    glPopMatrix();

    glPushMatrix();
    glTranslatef( 0.0, Cyl.hauteur, 0.0 );
    glRotatef( -90.0, 1.0, 0.0, 0.0 );
    gluDisk( qobjHaut, 0.0, Cyl.rayon, Cyl.Pointes, Cyl.Anneaux );
    glPopMatrix();

    glPopAttrib();
    glEndList();
}
```

MODÈLE HIÉRARCHIQUE VOITURE

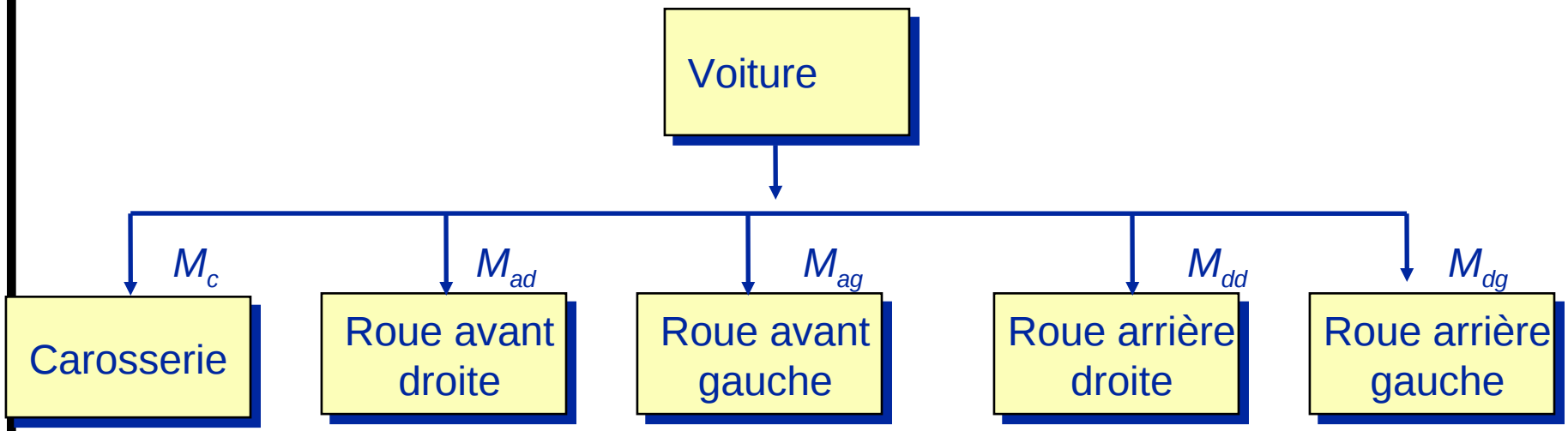
- Structure arborescente d'une voiture
- Graphe : ensemble de noeuds et d'arcs
- Arbre : graphe dirigé





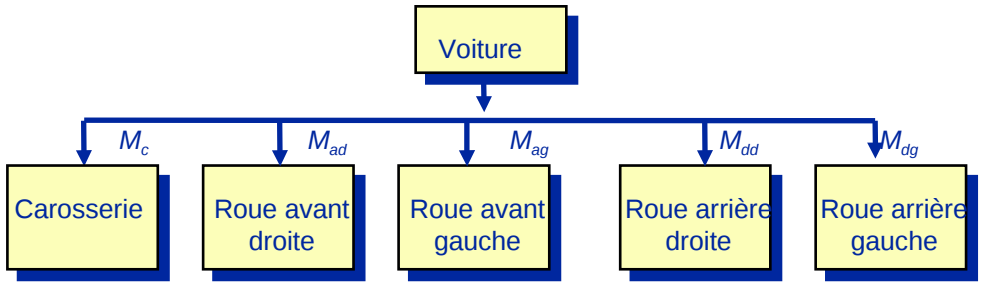
MODÈLE HIÉRARCHIQUE VOITURE

- Arbre avec transformations (matrices)



MODÈLE HIÉRARCHIQUE VOITURE

```
glPushMatrix();

glTranslate();
Carosserie();

glPushMatrix();
glTranslate()
TracerRoue();
glPopMatrix();

glPushMatrix()
glTranslate();
glRotate(180);
TracerRoue();
glPopMatrix();

glPushMatrix();
glTranslate()
TracerRoue();
glPopMatrix();

glPushMatrix()
glTranslate();
glRotate(180);
TracerRoue();
glPopMatrix();
```




```
glCallList(Tourner)
```

avec :

```
glNewList(Tourner);
glRotate (angle,...);
glEndList();
```