SYNTHESE D'IMAGE

BAH Mamadou - REPECAUD Benjamin

TABLE DES MATIERES

INTRODUCTION

Consignes

Afin de mettre en pratique les compétences acquises lors de l'UE Synthèse d'Image sur *OpenGL*, il nous a été demandé de modéliser un personnage à partir de primitives. Il sera ensuite habillé (couleur et texture) et animé.

Personnage choisi

Pour répondre à la demande, nous avons choisi le super-héros de Marvel Spiderman. Ce dernier a des particularités (geste pour lancer des toiles d'araignée) et un costume connu de tous et transposable en texture, il nous semblait donc pertinent de le modéliser.

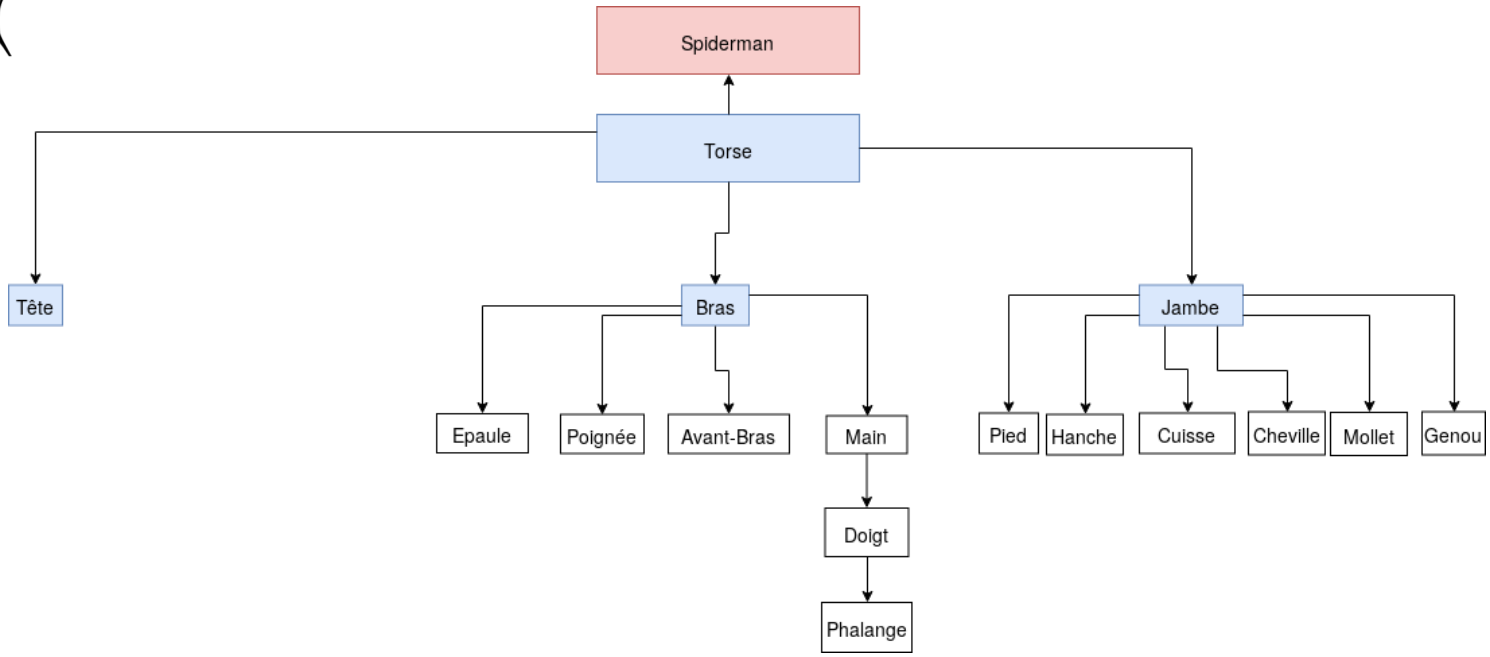
Avant de se lancer dans le développement, un travail préalable était de mise.

TRAVAIL PREALABLE

Décomposition du personnage

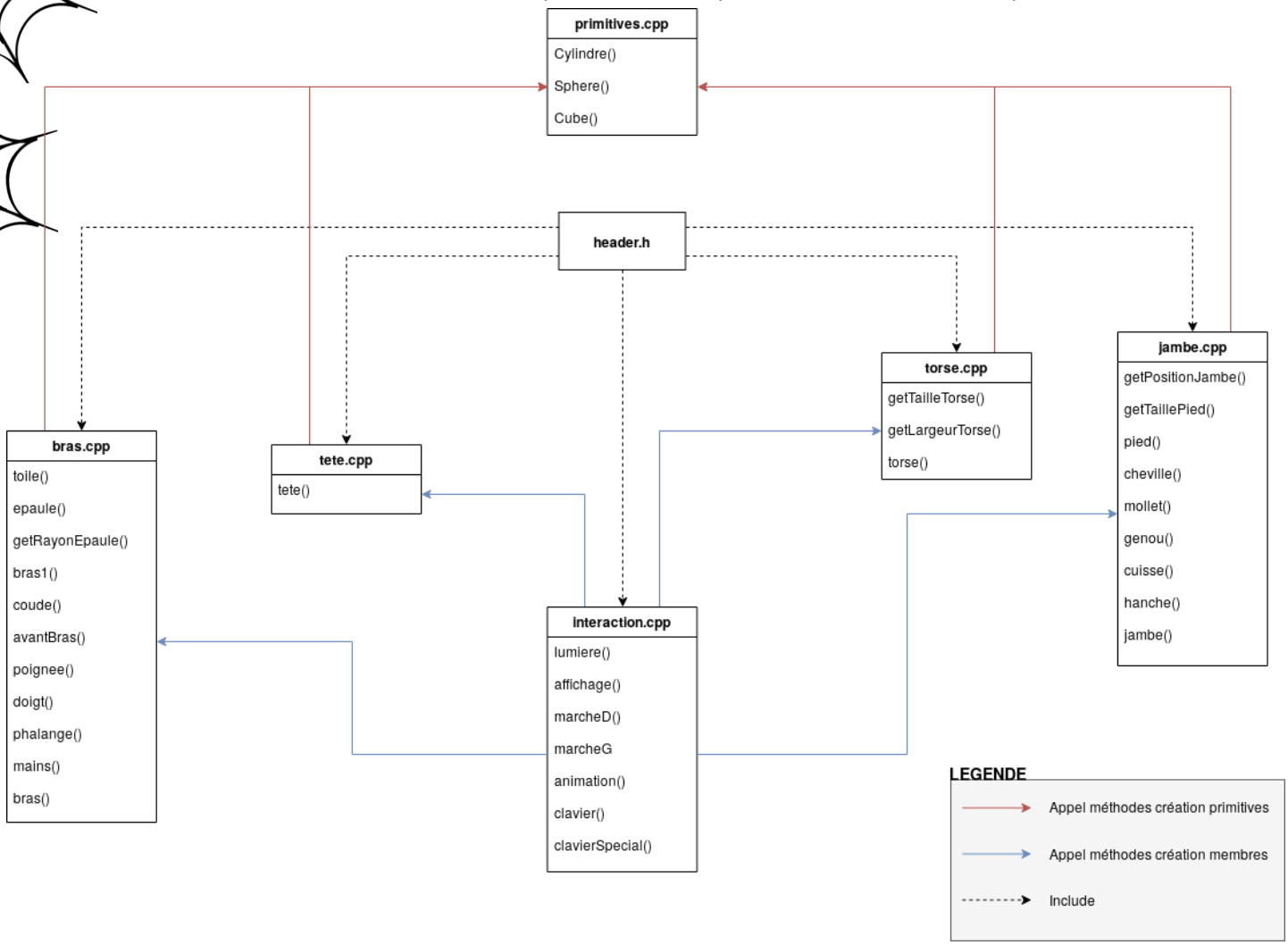
Comme il a été vu en cours, il est important de bien décomposer chaque partie de son personnage.

Chaque partie du corps est ainsi décomposée dans le schéma ci-dessous. Celui-ci nous a permis gérer les dépendances de chaque élément du personnage.



Architecture du projet

Après la décomposition du personnage vient l'architecture du projet.
En effet, plutôt que de tout coder dans un seul et même fichier, nous avons divisé le code selon les différentes parties du corps vues dans la décomposition.



Le prototype des différentes méthodes de chaque fichier se fait dans le fichier en-tête *header.h*.

```
/*
**********************************
INTERACTION.CPP
**********************************
*/
void clavier(unsigned char touche,int x,int y);
void clavierSpecial(int touche, int x, int y);
void mouse(int bouton,int etat,int x,int y);
void mousemotion(int x,int y);
void affichage();
void reshape(int x,int y);
void idle();
void mouse(int bouton,int etat,int x,int y);
void mousemotion(int x,int y);
//void affiche();
void repere();
void ecran(int argc,char **argv);

/*
**********************************
PRIMITIVES.CPP
**********************************
*/
void Cylindre(GLdouble r,GLdouble h, GLint lg,GLint lat);
void Sphere (GLdouble r, GLint lg, GLint lat);
void Cube(GLdouble taille);
```

Pour réaliser les membres du personnage, la méthode OpenGL *glutSolid* a été utilisée.

Un fichier *primitives.cpp* a été ajouté au projet afin de répertorier toutes les primitives que nous utilisons et de faire appel aux différents *glutSolid*. Chaque fonction (sphere, cylindre…) permet de créer rapidement une de ces formes dans les différents fichiers.

```
/*
Fichier contenant les méthodes de base de création de primitives
*/

#include <GL/freeglut.h>


//------------------------Cylindre
void Cylindre(GLdouble r,GLdouble h,GLint lg,GLint lat)
{
   glutSolidCylinder(r,h,lg,lat);

}

//------------------------Sphère
void Sphere (GLdouble r, GLint lg, GLint lat)
{
    glutSolidSphere(r, lg, lat);
}

//------------------------Cube
void Cube(GLdouble taille)
{
    glutSolidCube(taille);
}
```

Les méthodes des fichiers *tete.cpp*, *jambe.cpp*, ... permettant de créer les éléments souhaités sont appelées ensuite dans la méthode *affichage* du fichier *interaction.cpp* (ce fichier étant appelé ensuite dans le fichier principal *main.cpp*).

```
void jambe(GLdouble angleJambe, GLdouble angleMollet)
{

/*
************************************
    HANCHE
************************************
*/
    glPushMatrix();
        glColor3d(0,0,1);
        hanche(0, positionHanche, -tpied/2);
    glPopMatrix();

/*
************************************
    JAMBE ENTIERE
************************************
    -Positionnement de toute la jambe au niveau de la hanche
    -Rotation de toute la jambe
    -Création de chaque primitive dans un Push/PopMatrix
    -Rotation du mollet, indépendamment des autres parties de l
*/
    glPushMatrix();
        glTranslated(0, positionCuisse, -tpied/2);
        glRotated(angleJambe,1,0,0);
        glColor3d(0,0,1);

        //------------------------Cuisse
        glPushMatrix();
            cuisse(0, 0, 0);
        glPopMatrix();

        //------------------------Genou
        glPushMatrix();
            genou(0, positionGenou, 0);
/*
************************************
    PIED
************************************
*/
void pied(GLdouble scalex,GLdouble scaley,GLdouble scalez)
{
    glPushMatrix();
        glRotated(-90,0,1,0);
        glScaled(scalex, scaley, scalez);
        Cube(1);
    glPopMatrix();
}

/*
************************************
    CHEVILLE
************************************
*/
void cheville(GLdouble translatex,GLdouble translatey,GLdouble translatez)
{
    glPushMatrix();
        glTranslated(translatex,translatey,translatez);
        Sphere(rcheville,lg,lat);
    glPopMatrix();
}

/*
************************************
    MOLLET
************************************
*/
```

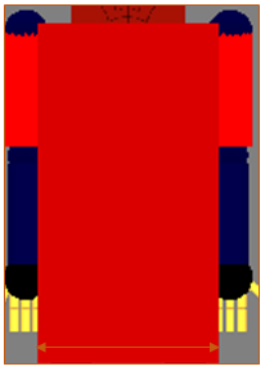
```
 /*
************************************CUBE
*/

//----------Dessous
glBegin(GL_POLYGON);
  glTexCoord2f(0.7,0.5);   glVertex3f(-0.5, 0.5, 0.5);
  glTexCoord2f(0.7,1.0);   glVertex3f(-0.5,-0.5, 0.5);
  glTexCoord2f(1.0,1.0);   glVertex3f( 0.5,-0.5, 0.5);
  glTexCoord2f(1.0,0.5);   glVertex3f( 0.5, 0.5, 0.5);
glEnd();

//---------Dessus
glBegin(GL_POLYGON);
    glTexCoord2f(0.3,0.5);   glVertex3f( 0.5, 0.5,-0.5);
    glTexCoord2f(0.3,1.0);   glVertex3f( 0.5,-0.5,-0.5);
    glTexCoord2f(0.7,1.0);   glVertex3f(-0.5,-0.5,-0.5);
    glTexCoord2f(0.7,0.5);   glVertex3f(-0.5, 0.5,-0.5);
glEnd();

//-------------Gauche
glBegin(GL_POLYGON);
    glTexCoord2f(0.0,0.0);   glVertex3f( 0.5, 0.5, 0.5);
    glTexCoord2f(0.0,0.5);   glVertex3f( 0.5,-0.5, 0.5);
    glTexCoord2f(0.3,0.5);   glVertex3f( 0.5,-0.5,-0.5);
    glTexCoord2f(0.3,0.0);   glVertex3f( 0.5, 0.5,-0.5);
glEnd();
```

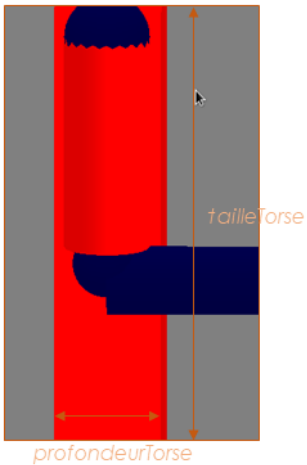
Un fichier pour une partie du corps

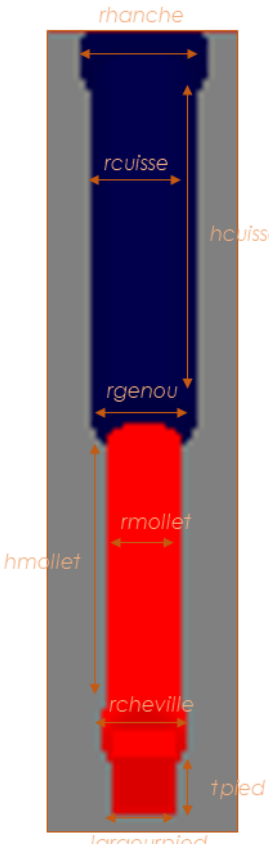
```
void affichage()
{
    GLdouble positionJambe = getPositionJambe();
    GLdouble taillePied = getTaillePied();
    GLdouble tailleTorse = getTailleTorse();
    GLdouble largeurTorse = getLargeurTorse();
    GLdouble rayonEpaule = getRayonEpaule();

    repere();
    lumiere();
/*
    JAMBE
*****************************************
*/
    //---------------------Jambe Gauche
    glPushMatrix();
        jambe(angleJambeD, angleMolletD);
    glPopMatrix();

     //---------------------Jambe Droite
     glPushMatrix();
        glTranslated(ecartJambe, 0, 0);
        jambe(angleJambeG,angleMolletG);
     glPopMatrix();

/*
    TORSE
*****************************************
*/
    //----------------TRANSLATE EN X (par rapport à la taille du pied /
    glPushMatrix();
        torse(ecartJambe/2, positionJambe+tailleTorse/2, -taillePied/2);
    glPopMatrix();
```

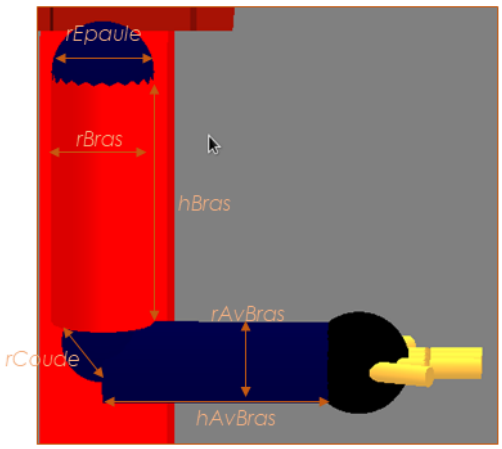
Appel des méthodes permettant de créer les membres du personnage

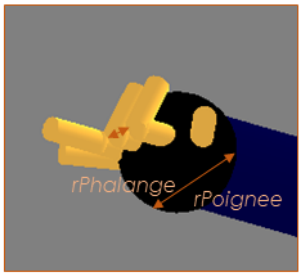
Chaque élément est placé par rapport à la taille des autres membres du personnages. Une cuisse aura pour position la taille du pied, de la cheville, du mollet et du genou additionnés.











PRESENTATION DU PERSONNAGE

Membre après membre, Spiderman a pris forme, sans oublier de respecter les différentes consignes.

Primitive à partir d'une représentation paramétrique

La tête est un cube réalisé à partir d'un *GL_POLYGON* disponible sur *OpenGL*. Chaque face a ainsi été initialisée, point par point, dans la méthode *tete* du fichier *tete.cpp*.

```
 /*
****************************************CUBE
*/

//-----------Dessous
glBegin(GL_POLYGON);
  glTexCoord2f(0.7,0.5);    glVertex3f(-0.5, 0.5, 0.5);
  glTexCoord2f(0.7,1.0);    glVertex3f(-0.5,-0.5, 0.5);
  glTexCoord2f(1.0,1.0);    glVertex3f( 0.5,-0.5, 0.5);
  glTexCoord2f(1.0,0.5);    glVertex3f( 0.5, 0.5, 0.5);
glEnd();

//---------Dessus
glBegin(GL_POLYGON);
    glTexCoord2f(0.3,0.5);    glVertex3f( 0.5, 0.5,-0.5);
    glTexCoord2f(0.3,1.0);    glVertex3f( 0.5,-0.5,-0.5);
    glTexCoord2f(0.7,1.0);    glVertex3f(-0.5,-0.5,-0.5);
    glTexCoord2f(0.7,0.5);    glVertex3f(-0.5, 0.5,-0.5);
glEnd();

//-------------Gauche
glBegin(GL_POLYGON);
    glTexCoord2f(0.0,0.0);    glVertex3f( 0.5, 0.5, 0.5);
    glTexCoord2f(0.0,0.5);    glVertex3f( 0.5,-0.5, 0.5);
    glTexCoord2f(0.3,0.5);    glVertex3f( 0.5,-0.5,-0.5);
    glTexCoord2f(0.3,0.0);    glVertex3f( 0.5, 0.5,-0.5);
glEnd();
```

Nous y avons ensuite ajouté une texture plaquée sur chaque face.

Textures

La gestion des textures s'est faite à partir des méthodes réalisées en TP de Synthèse d'Images (chargement des images pour texture via *loadJpegImage*).

Nous avons aussi initialisé un tableau de texture afin d'en gérer plusieurs.

Le tableau *idTexture* contient l'identifiant des deux textures (plaquée et enroulée) dont nous avons besoin.

Ces identifiants sont affectés à l'image souhaitée.

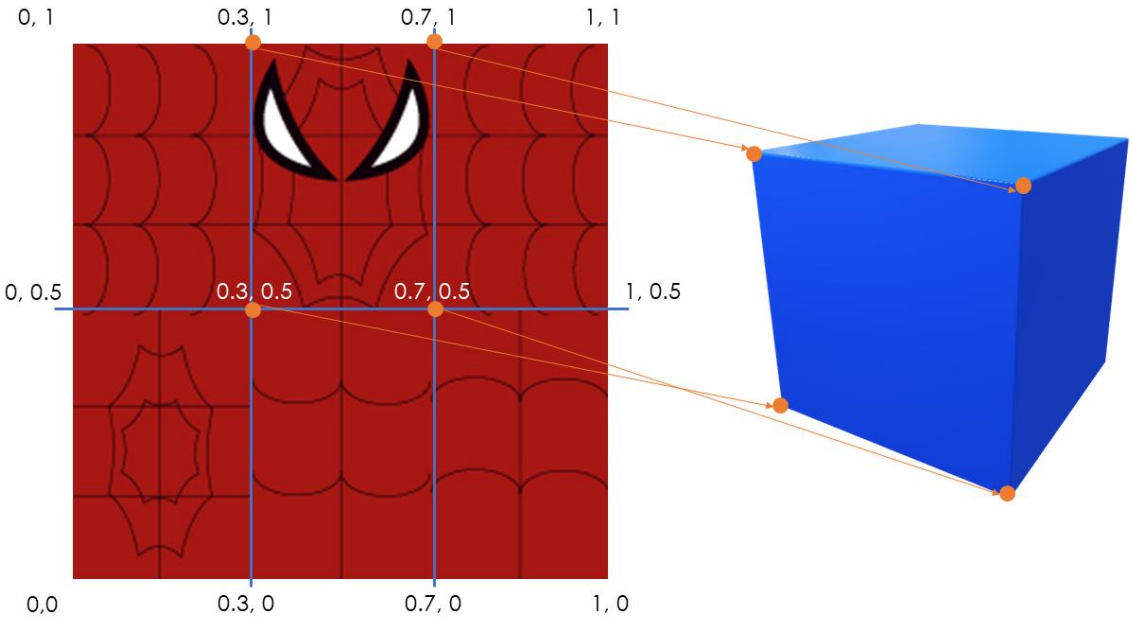
```
//----Init tableau de texture
glGenTextures(2, idTexture);

//--------------------------------------------TEXTURE1
glBindTexture(GL_TEXTURE_2D, idTexture[0]);
loadJpegImage("textures/toile.jpg", texture);

//--------------------------------------------TEXTURE2
glBindTexture(GL_TEXTURE_2D, idTexture[1]);
loadJpegImage("textures/masque.jpg", texture);
```

Il suffit ensuite de les appeler lors de la création des primitives.

```
/*
    TETE
*****************************************
*/
    glPushMatrix();
        glBindTexture(GL_TEXTURE_2D, idTexture[0]);
        glScaled(scaleToile, scaleToile, scaleToile);
        toile();
    glPopMatrix();


    glPushMatrix();
        glTranslated(-largeurTorse/2-0.75/3,tailleTorse+positionJambe+0.5/2,-taillePied/2);
        glBindTexture(GL_TEXTURE_2D, idTexture[1])
        tete(6,4);
    glPopMatrix();
```

Texture plaquée

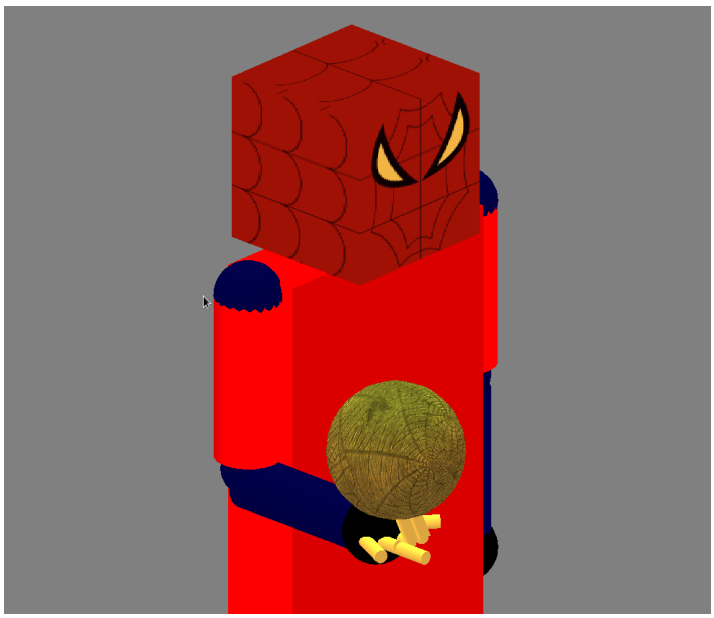
Comme évoqué plus haut, nous avons plaqué sur chaque face du cube (la tête) la partie du masque souhaitée. La taille de la texture étant de 256 pixels par 256, il nous a fallu diviser en 6 la texture, comme le nombre de face du cube. Les coordonnées de la texture ont ensuite été adaptées au besoin.



Texture enroulée

Spiderman a la particularité de produire des toiles d'araignée : c'est ce que nous avons modélisé dans notre projet. La texture est ainsi enroulée autour d'une sphère de type quadrique.

```
void toile()
{
    glEnable(GL_TEXTURE_2D);
        //--------Paramètre de création d'objet
        GLUquadric* params;
        params = gluNewQuadric();
        gluQuadricTexture(params, GL_TRUE);
            glTranslated(-rToile - tTorse/2 - rPoignee, positionEpaule+positionCoude+rToile, -positionAvBras-rToile/2);

            // paramètre quadrique, rayon, slice, stack
            gluSphere(params, rToile, 20, 20);
        //--------------Libération de la mémoire
        gluDeleteQuadric(params);
    glDisable(GL_TEXTURE_2D);
}
```



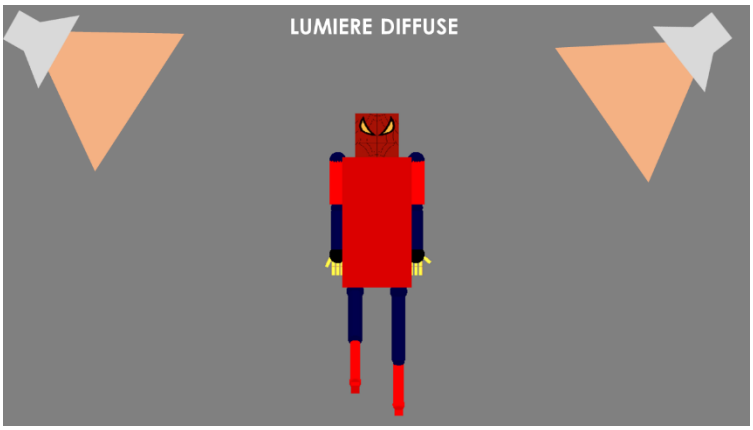
Rendu après application de la texture

Lumières

Deux types de lumières ont été placées sur la 'scène', deux diffuses et deux ambiantes. Les premières citées sont de part et d'autre du personnage, permettant d'éclairer les deux côtés.

L'éclairage devant et derrière le personnage est géré par deux lumières ambiantes.

```
glEnable(GL_LIGHTING);
glEnable(GL_COLOR_MATERIAL);
    /*
    ********************************************
    Position / couleur des sources
    ********************************************
    */
    GLfloat hauteurDevDer = 10.0;
    GLfloat hauteurDG = 10.0;
    GLfloat sourceG[] = {-5.0, hauteurDG, 0.0, 0.0};
    GLfloat sourceD[] = {5.0, hauteurDG, 0.0, 0.0};
    GLfloat sourceFace[] = {0.0, hauteurDevDer, 5.0, 0.0};
    GLfloat sourceDerriere[] = {0.0, hauteurDevDer, -5.0, 0.0};

    /*----------Couleur lumière Diffuse (r,v,b,intensité)----------*/
    GLfloat couleurLumiere[] = {0.329412, 0.223529, 0.027451, 5.0};

    /*
    ********************************************
    LUMIERE DIFFUSE
    ********************************************
    */
    //-------------------------------------------LUMIERE 1 (A GAUCHE)
    glLightfv(GL_LIGHT0, GL_POSITION, sourceG);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, couleurLumiere);

    //-------------------------------------------LUMIERE 2 (A DROITE)
    glLightfv(GL_LIGHT1, GL_POSITION, sourceD);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, couleurLumiere);

    /*
    ********************************************
    LUMIERE AMBIANTE
    ********************************************
    */
    //-------------------------------------------LUMIERE 3 (DEVANT)
    glLightfv(GL_LIGHT2, GL_POSITION, sourceFace);
    glLightfv(GL_LIGHT2, GL_AMBIENT, couleurLumiere);

    //-------------------------------------------LUMIERE 4 (DERRIERE)
    glLightfv(GL_LIGHT3, GL_POSITION, sourceDerriere);
    glLightfv(GL_LIGHT3, GL_AMBIENT, couleurLumiere);


glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
glEnable(GL_LIGHT2);
```



Zoom

Pour zoomer sur l'objet, nous avons placé une variable *zoom* dans la gestion de la perspective de l'objet (*glOrtho*). Cette variable évolue dans la méthode *clavier*, si 'Z' enfoncé : incrémentation de la variable, la caméra se rapproche du personnage, si 'z' enfoncé : décrémentation, on s'éloigne alors de l'homme araignée.

```
glOrtho(
     -4+zoom,4-zoom, // left, right
     -1+zoom,7-zoom, //bottom, top
     -4+zoom,4-zoom //near, far
);
```

Différence ou addition avec la variable pour garder l'objet au centre de la caméra (s'éloigne sur un côté sinon)

```
//----------------------Zoom
//Avance
case 'Z' :
     if(zoom<=2)
     {
          zoom = zoom + 1;
     }
break;

//S'éloigne
case 'z' :
     if(zoom>=-6)
     {
          zoom--;
     }
break;
```

Modification de la vue de l'objet

Les touches directionnelles du clavier sont considérées comme des touches 'spéciales', il fallait donc les déclarer comme telles. Une méthode *clavierSpecial* a été implémentée afin de gérer ces touches.

```
glutSpecialFunc(clavierSpecial);
```

Appel de glutSpecialFunc pour gérer les touches dites spéciales

Le déplacement autour du personnage s'est fait de la même manière que pour le zoom. Un *anglex* et un *angley* sont deux variables récupérant l'angle en abscisse et en ordonnée. Celles-ci sont incrémentées ou décrémentées suivant la touche enfoncée.

```
int incrementeangle = 5;
void clavierSpecial(int touche, int x, int y)
{
    switch(touche)
    {
        //Gauche
        case GLUT_KEY_LEFT:
                anglex+=incrementeangle ;
                glutPostRedisplay();
        break;

        //Droite
        case GLUT_KEY_RIGHT:
                anglex-=incrementeangle ;
                glutPostRedisplay();
        break;

        //Haut
        case GLUT_KEY_UP:
                angley+=incrementeangle ;
                glutPostRedisplay();
        break;

        //Bas
        case GLUT_KEY_DOWN:
                angley-=incrementeangle ;
                glutPostRedisplay();
        break;
    }
}
```

GLUT_KEY_LEFT: touche de gauche, GLUT_KEY_RIGHT: touche de droite, GLUT_KEY_DOWN: touche du bas, GLUT_KEY_UP: touche du haut

Animation

Automatique

L'animation automatique se fait au niveau des jambes, donnant l'impression que notre personnage marche sur place. La rotation des jambes par rapport à la hanche est faite grâce à deux variables (une pour chaque jambe) incrémentée jusqu'à une certaine valeur. On incrémente celle-ci dans la méthode *marcheD* et *marcheG* et on les appelle dans la méthode *animation*. *glutIdlefunc* nous permet de gérer les animations automatiques.

Les deux jambes commencent à des angles différents, cela nous permet d'alterner les deux pour un aspect relativement fluide et réaliste.

```
void animation()
{
    marcheD();
    marcheG();
    glutPostRedisplay();
}
```

```
glutIdleFunc(animation);
```

glutIdleFunc: gérer les animations

```
//--------------------Jambe droite
void marcheD()
{
    if(angleJambeD > -45)
    {
        angleJambeD--;
    }
    else if (angleJambeD == -45)
    {
        angleJambeD = 0;
    }

    glutPostRedisplay();
}
```

```
//---------------------Jambe gauche
void marcheG()
{
    if(angleJambeG <0)
    {
        angleJambeG++;
    }
    else if (angleJambeG == 0)
    {
        angleJambeG = -45;
    }

    glutPostRedisplay();
}
```

Avec les touches du clavier

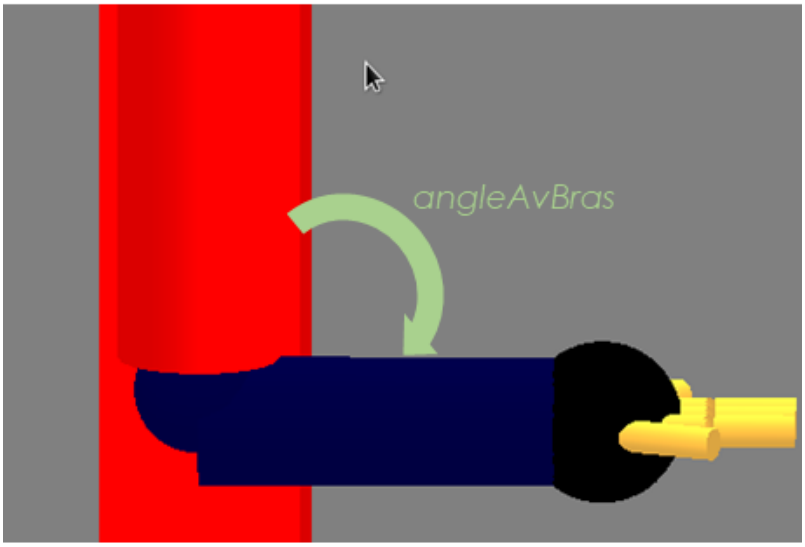
Qui dit Spiderman dit jet de toile d'araignée. Nous avons animé son bras et sa main pour qu'il puisse utiliser ses pouvoirs.

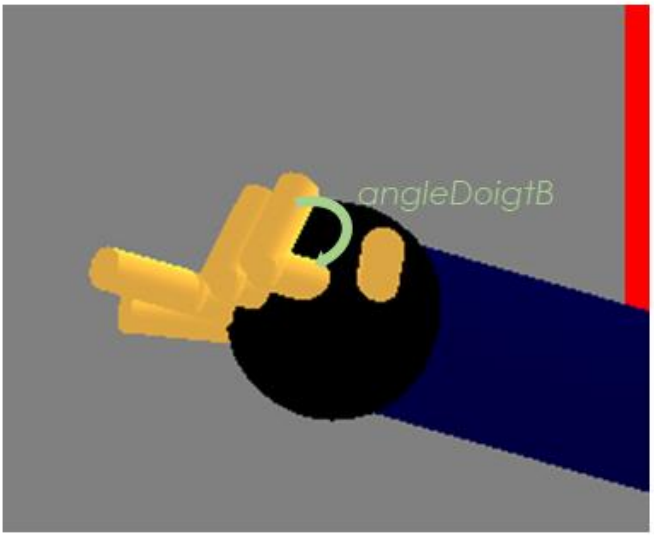
Même principe que pour l'animation automatique, des variables permettent à l'avant-bras de faire une rotation par rapport au coude et les doigts en font une par rapport aux phalanges.

La touche 'b' fait lever le bras jusqu'à un certain angle, tandis que la touche 'd' fait replier le majeur et l'index dans le creux de sa main.

```
glPushMatrix();
    glTranslated(brasX, positionBras1, profondeur);
    glRotated(angleBrasZ, 0, 0, 1);
    glRotated(angleBrasX, 1, 0, 0);
```

```
//------------------Main
glPushMatrix();
    glTranslated(0, positionAvBras, 0);
    mains(angleDoigtB);
glPopMatrix();
```





Personnage final

CONCLUSION

Malgré plusieurs difficultés (lumières, textures, compréhension d'OpenGL), nous sommes parvenus à respecter les demandes grâce à notre réflexion et aux compétences acquises en Synthèses d'Images.