Ouattara Ismael Simeon Herve – IE03-05b

Rapport de Projet

Info3b

I- Camera

```
camera {

location <12,7,-2>
look_at <5,0,3>
right<-image_width/image_height,0,0>

}


light_source {
        <-1,10,0>
        color < 0.5,0.5,0.5 >
        }

light_source {
        <-1,100,0>
        color < 1,1,1 >
        }

light_source {
        <-3.5,20.1,-5>
        color < 1,1,1 >
        spotlight
        tightness 10
        radius 1
        point_at <4,0,4>
        }
```

Pour pouvoir voir notre échequier, il nous faut **une camera** afin de pouvoir regarder **la scène en 3D**.

D'où on met sa localisation à « **<12,7,-2>»** ce qui nous permet de voir **à distance** (grâce à **l'axe des X** ici égal à 12) un peu en hauteur(grâce à **l'axe des Y** ici égal à 7) et **l'axe des Z** ici égal à -2.

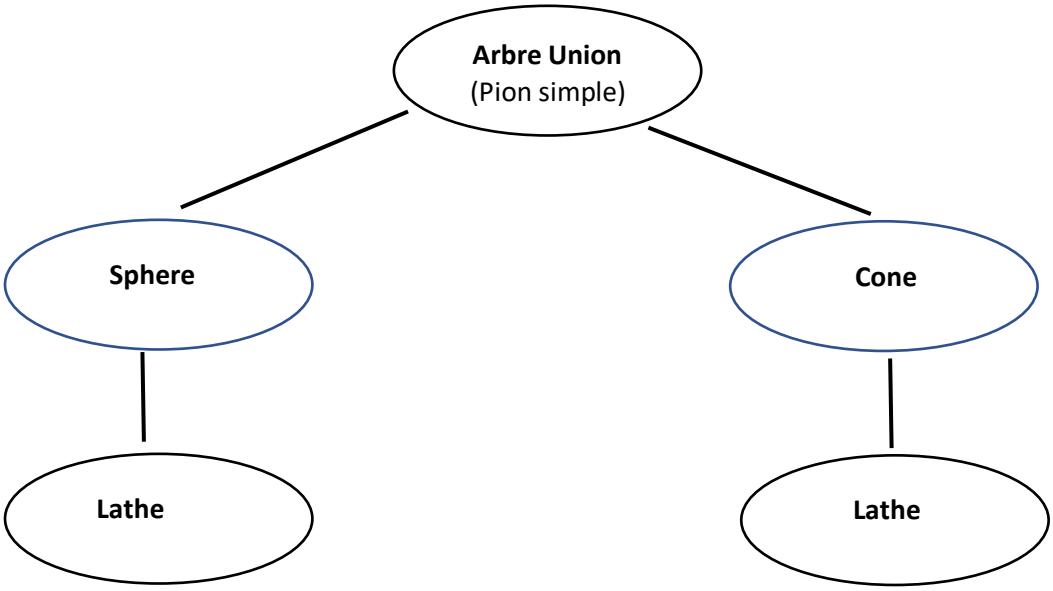
Ensuite, le **« look_at »** qui est suivi du vecteur position de la cible nous permet de connaître la **direction du regard**. Et le « **right <-image_width/image_height,0,0>** » nous permet de définir le **repère de notre scène 3D**.

II- Définition du sol

```
//pour le sol

plane {
        y,-3
        pigment { color <1,1,1> }
        finish { reflection 1
         specular 0.3 }
        }
```

III- Définition des arbres

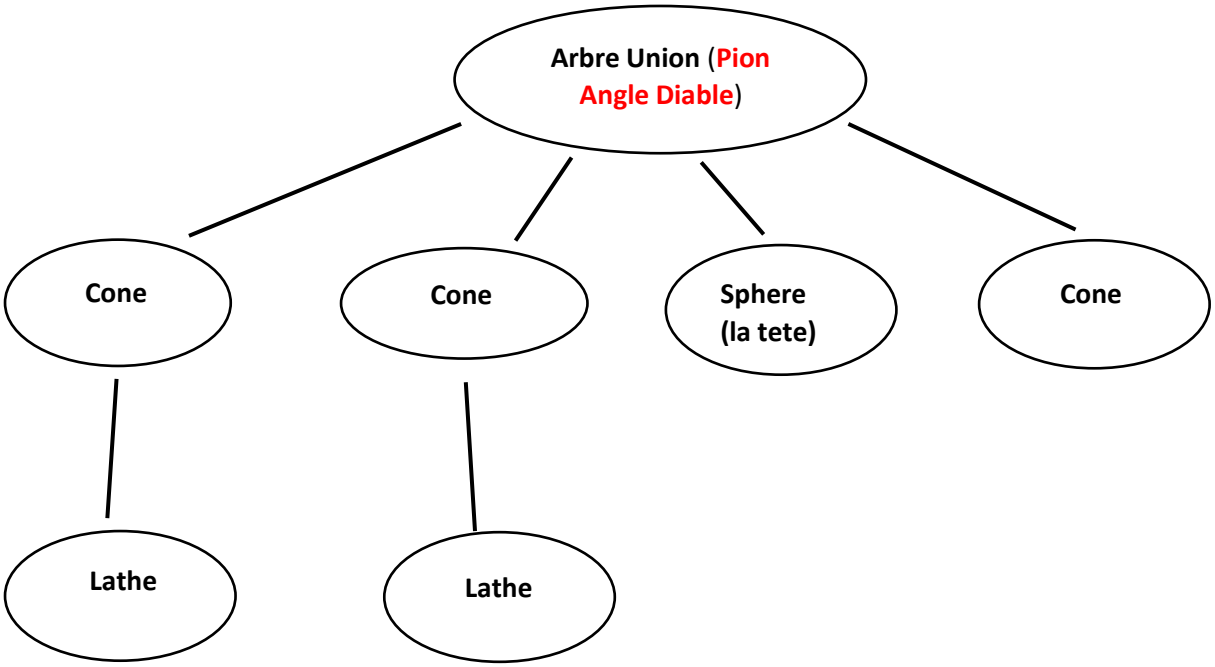
Pour faire un **arbre du pion simple**, nous avec réuni un sphere et une cone .

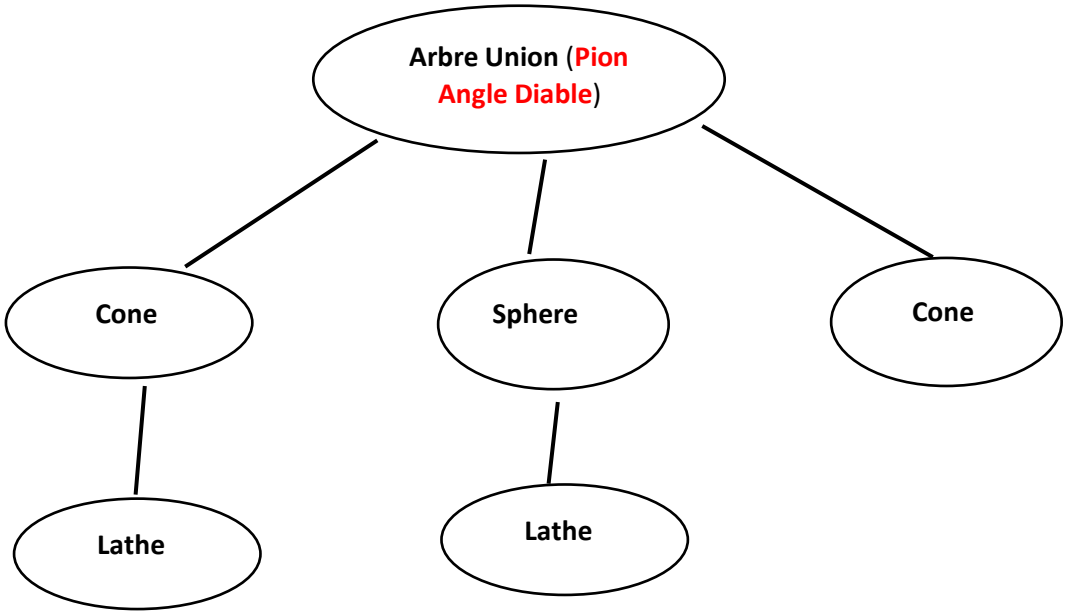


Code et Affichage des arbres

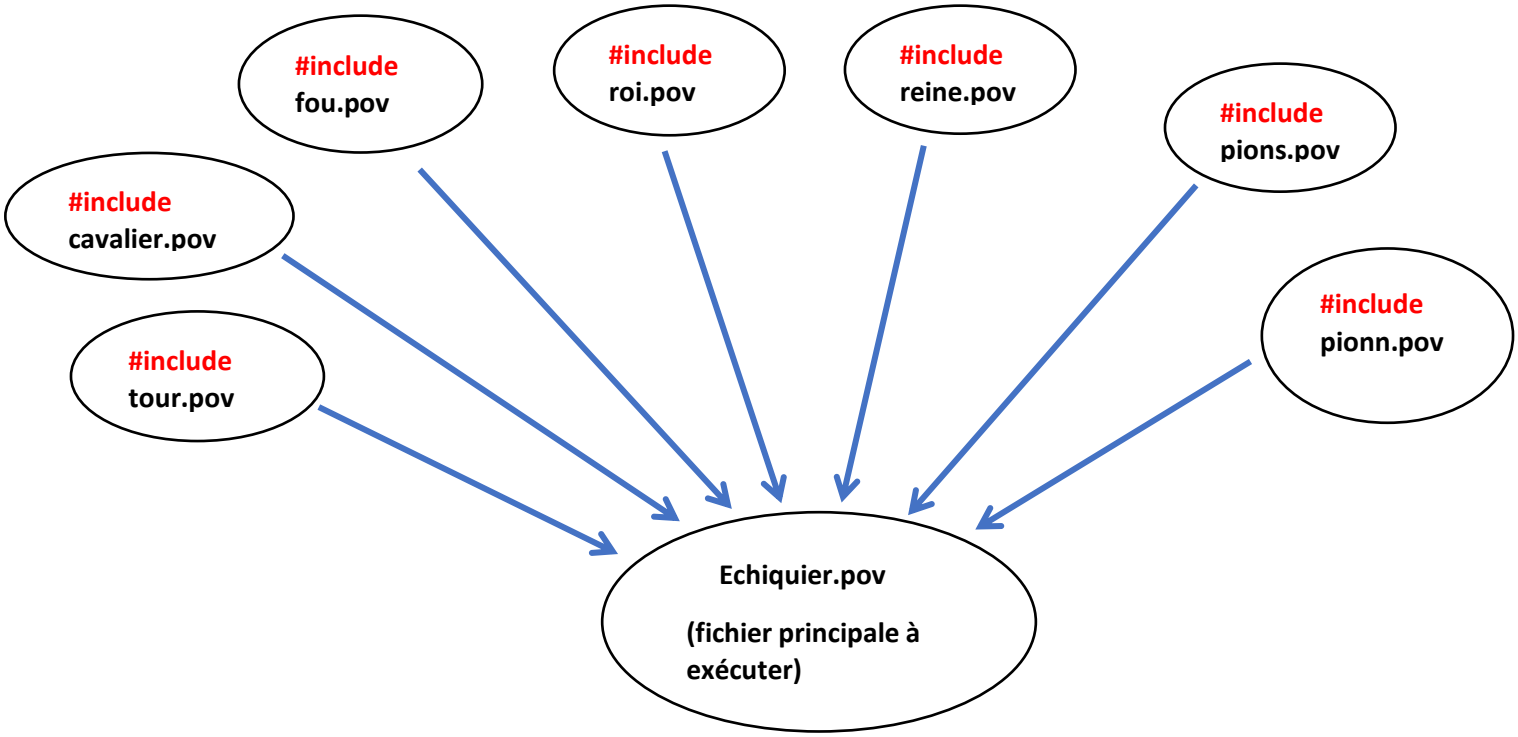
```
union{

    lathe {    //utilisation d'un lathe lisse

    bezier_spline
    4 //nbr_Pt
    <0,0,0>,<0,0.5,0>,
    pigment { Black transmit .5}
}


lathe {     //utilisation d'un lathe lisse
     bezier_spline
     4   //nbr_Pt
     <0,1.7,0>,<0,0,0>,<0,1.5,0>,<0,1.7,0>
     pigment { Black transmit .5}
}

}
```

```
#if (typeDePion = 1)  // Pion simple
          union {       sphere {  <0,1.7,0> , 0.4  }
                        cone {  <0,0,0> ,  0.5 , <0,1.5,0> , 0  }
 #end

 #if (typeDePion = 2)  // Ange VS Diable
                      #if (joueur = 1)  // Si c'est un pion du joueur 1 on met un Diable
                           union {

                                 cone { <0,0,0>, 0.05, <0,0.5,0>, 0 // Corne du diable
                                  rotate 20*z
                                  translate <-0.15,2,0>
                                      }

                                 cone { <0,0,0>, 0.05, <0,0.5,0>, 0 // Corne du diable
                                  rotate -20*z
                                  translate <0.15,2,0>
                                      }
                                 sphere { <0,1.7,0>, 0.4 }  // tete

                                 cone { <0,0,0>, 0.5, <0,1.5,0>, 0 }

                       #else       // Si c'est un pion du joueur 0 on met un Ange

                              union {
                                torus { 0.4, 0.03     // Oréole de l'ange
                                        rotate -20*x
                                        translate <0,2.2,-0.1>
                                      }

                                sphere { <0,1.7,0>, 0.4 }

                                cone { <0,0,0>, 0.5, <0,1.5,0>, 0 }
                       #end

   #end
```





modélisation du projet



: appel des différents arbres ou pieces dans le fichier echiquier.pov

IV- Montage

Tour

construites à l'aide de primitives usuelles via **des arbres C.S.G**

union de plusieurs primitives à savoir torus, cylinder, cone et 2 paramétres qui sont la position du tour sur l'echequier et la couleur

```
#macro tour(position,couleur)

union {
        torus { 1, 0.1 }
        torus { 1.1, 0.2
                translate 0.3*y }

        cylinder { <0,0,0>, <0,0.3,0>, 1
                translate 0.4*y }

        cone {  <0,0,0>, 0.7, <0,2.2,0>, 0.3
                translate 0.6*y }
        cone  { <0,0,0>, 0.3, <0,0.3,0>, 0.5
                translate 2.8*y }
        cylinder { <0,0,0>, <0,0.2,0>, 0.3
                translate 3.1*y }

        difference {                    // tete de la tour
                cylinder {
                        <0,0,0>
                        <0,0.6,0>
                        0.7
                        translate 3.3*y
                        }
                cylinder {
                        <0,0,0>
                        <0,0.6,0>
                        0.5
                        translate 3.4*y
                        }

                box {
                        <-1,0,0>
                        <1,0.5,0.4>
                        translate <0,3.6,-0.2>
                        }
                box {
                        <-1,0,0>
                        <1,0.5,0.4>
                        translate <0,3.6,-0.2> rotate 90*y
                        }
        }


        pigment { color couleur}
        finish {
                reflection 0.3
                specular 0.5
        }
        scale 0.3
        translate position
       }

#end
```

Cavalier

construites à l'aide de primitives usuelles via **des arbres C.S.G**

union de plusieurs primitives à savoir torus, cylinder, sphere et 2 paramétres qui sont la position du cavalier sur l'echequier , la couleur et le rot. Ces primitives vont nous permet de faires plusieurs surface de revolution pour le bon déroulement de la construction de notre objet.

Roi

construites à l'aide de primitives usuelles via **des arbres C.S.G**

union de plusieurs primitives à savoir torus, cylinder, cone et box et 2 paramétres qui sont la position du cavalier sur l'echequier et la couleur. Ces primitives vont nous permet de faires plusieurs surface de revolution pour le bon déroulement de la construction de notre objet.

Reine

construites à l'aide de primitives usuelles via **des arbres C.S.G**

union de plusieurs primitives à savoir torus, cylinder, sphere et 2 paramétres qui sont la position du cavalier sur l'echequier , la couleur et le rot. Ces primitives vont nous permet de faires plusieurs surface de revolution pour le bon déroulement de la construction de notre objet.

Fou

Pour les fous nous avons utiliser **4 blobs**

qui ont été rempli de spheres de taille différentes mais par contre les blobs.

Cependant , **un problème** arrive j'arrive pas a associe les blobs a mes macro cela à pour effet que lorsqu'il soit entièrement dependant des blobs.

le macro aide au positionnement du fou

les fous sont construits par des blobs. **Ces blobs vont nous permet de faires plusieurs surface de revolution pour le bon déroulement de la construction de notre objet.**

et l'union de plusieurs surface de revolution à savoir torus, cylinder et du cone.

#macro (position,couleur,rot) : qui permet de positionner du fou, sa couleur

```
#macro fou(position,couleur,rot)

union
 {


          torus {
                        1
                        0.1
                       }

           torus {      1.1
                        0.2
                       translate 0.3*y
                 }

             cylinder {
                        <0,0,0>
                        <0,0.3,0>
                        1
                        translate 0.4*y
                        }


             cone {
                <0,0,0>
                0.5
                <0,1.5,0>
                0.2
                translate 0.7*y

             }

             cone {
                <0,0,0>
                0.2
                <0,0.2,0>
                0.4
                translate 2.2*y

             }


             torus {
                        0.4
                        0.05
                        translate 2.4*y
             }

             torus {
                        0.3
                        0.05
                        translate 2.5*y
```

Pions

```
#macro pions(position,couleur)


union {
            torus {
                        1
                        0.1
                       }
            torus {      1.1
                        0.2
                       translate 0.3*y
                   }
              cylinder {
                        <0,0,0>
                        <0,0.3,0>
                        1
                        translate 0.4*y
                        }
                cone {
                        <0,0,0>
                        0.7
                        <0,1,0>
                        0.3
                        translate 0.7*y

                }
                cone {
                        <0,0,0>
                        0.5
                        <0,0.2,0>
                        0.3
                        translate 1.7*y

                }
                sphere {
                        <0,0,0>
                        0.4
                        translate 2.2*y

                }

          pigment { color  couleur}
          finish {
                  reflection 0.3
                  specular 0.5
          }

          scale 0.3
          translate position

         }
#end
```

les pions sont construits à l'aide de **trois surfaces de révolution** avec un raccord G1 entre chaque surface, l'utilisation de deux **« lathe » lisses** est obligatoire et celles-ci sont se raccorder entre elles ;

V-Création de l'echequier

Appel des macros pour **créé les pièce de l'échiquier**

Toutes les macros prennent au moins 2 paramètres : **(- la position, - la couleur**)

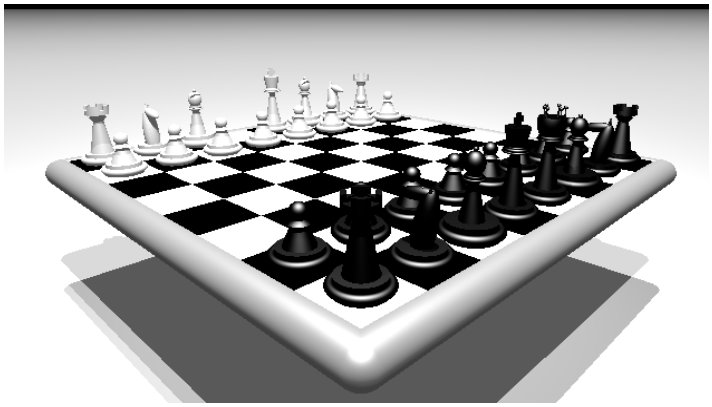
Certaine prennent en plus leur orientation comme **le cavalier et le fou**

construites à l'aide de primitives usuelles via **des arbres C.S.G**

```
    tour(<0.5,0,0.5>,Black)
    tour(<7.5,0,0.5>,Black)
    tour(<0.5,0,7.5>,White)
    tour(<7.5,0,7.5>,White)

    cavalier(<1.5,0,0.5>,Black,-90*y)
    cavalier(<6.5,0,0.5>,Black,-90*y)
    cavalier(<1.5,0,7.5>,White,90*y)
    cavalier(<6.5,0,7.5>,White,90*y)

    fou(<2.5,0,0.5>,Black,-90*y)
    fou(<5.5,0,0.5>,Black,-90*y)
    fou(<2.5,0,7.5>,White,90*y)
    fou(<5.5,0,7.5>,White,90*y)

    roi(<3.5,0,0.5>,Black)
    roi(<4.5,0,7.5>,White)

    reine(<4.5,0,0.5>,Black)
    reine(<3.5,0,7.5>,White)

    posePions()
#end


#macro posePions()
        #declare i=0;

        #while (i<8)

            pions(<i+0.5,0,1.5>,Black)
            pions(<i+0.5,0,6.5>,White)

            #declare i=i+1 ;
        #end

#end
```

VI - Création du Contour de l'echéquier

```
//------------ Création de l'échiquier (grâce a l'appel des différents fichiers importé) ------\\
#macro echiquier()

                // Création du Damier \\
    box  {
          <0,0,0>
          <8 , -0.5 , 8 >
          pigment {
                checker
                color <1,1,1>
                color <0,0,0>
          }
    finish { reflection 0.3
             specular 0.3  }
        }


            // Création du Contour du damier \\
    union {

    box {  <-0.17,-0.5,-0.17> , <8+0.17 , 0 ,0>  }
    box { <8,-0.5,0> , <8+0.17 , 0 ,8+0.17> }
    box { <-0.17,-0.5,8> , <8+0.17 , 0 ,8+0.17> }
    box { <-0.17,-0.5,0> , <0 , 0 ,8+0.17> }

    cylinder {<-0.1 , -0.25 , -0.1>, <8+0.1 , -0.25 , -0.1>, 0.275}
    cylinder {<8+0.1 , -0.25 , -0.1>  ,<8+0.1 , -0.25 , 8+0.1> ,  0.275 }
    cylinder {  <8+0.1 , -0.25 , 8+0.1>, <-0.1 , -0.25 , 8+0.1>,  0.275 }
    cylinder { <-0.1 , -0.25 , 8+0.1> ,  <-0.1 , -0.25 , -0.1>, 0.275 }

    sphere { <-0.1 , -0.25 , -0.1> ,   0.275}
    sphere { <8+0.1 , -0.25 , -0.1> ,  0.275}
    sphere {  <8+0.1 , -0.25 , 8+0.1> ,  0.275 }
    sphere { <-0.1 , -0.25 , 8+0.1> 0.275 }

    pigment {color <0.5,0.5,0.5> }
    finish { reflection 0.1
             specular 1  }


    }
```



VII- Animation

1)Création du fichier d'extension.ini

On créé notre fichier d'extension .ini et qui à le même nom que le fichier povray (ici **echequier.ini**) et ensuite on ajoute le code ci-après pour qu'il puisse nous générer plusieurs (10) images de notre scène qui seront regroupées grâce **au logiciel GIMP** (sous windows) afin d'obtenir le fichier gif. Génére plusieurs images au format png puis les convertir en un fichier gif.

Code :

```
Antialias=On
Antialias_Threshold=0.3
Antialias_Depth=3

Input_File_Name=echequier.pov

Initial_Frame=1
Final_Frame=10
Initial_Clock=0
Final_Clock=1

Cyclic_Animation=on
Pause_when_Done=off
```

2)Paramètre à ajouter dans notre fichier povray

Pour pouvoir faire sauter nos personnages il ajouter une variable qui sera notre horloge et qu sera implémenter sur l'axe des Y du translate des animaux afin de les faire sauter en fonction du temps.

Code

Variable horloge : Pour Linux

```
#declare horloge = clock;
//#declare My_Clock = Start + (End-Start)*clock;

//#declare n=10;
// #declare i=0;
//#declare Start = 0; //Pi+9*Pi/12
//povray -W1280 -H1024 +A +R9 +KFI1 + KFF5 echequier.pov


//#declare End = 1*Pi;
//povray -W1280 -H1024 +A +R9 +KFI1 + KFFn echequier.pov =>#declare End = n+3;
//#declare thetaR = My_Clock;
//#declare sca = 2.75;
```

Les Déplacements Lors des animations

-deplacement rectiligne ou en L

-un deplacement vertical

-un deplacement en forme d'un arc de parabole via une courbe de bézier

```
//pour la boucle afin de faire l'animation (Blanc)

   #declare _P3 = <0,2.5,0>;
   #declare _P2 = <0,3,2>;
   #declare _P1 = <0,3.5,2>;
   #declare _P0 = <0,3.5,0>;

  #declare n=clock;
  #declare frame=10*n;

  #for(i,0,n)
   #if(frame>=1 & frame<5)
      reine(<3.5,0,7.5-(0.5*frame)>,White)   //deplacement horizontal ou vertical
   #end

   #if(frame>=5 & frame<7)
      reine(<4.5,0,0.5+(0.5*frame)>,Black)   //deplacement horizontal ou vertical
   #end



     #if(frame>=7 & frame<9)
      roi(<3.5,0,0.5+(0.5*frame)>,Black)   //deplacement horizontal ou vertical
   #end



   #if(frame>=9 & frame<14)
      tour(<7.5-(0.5*(frame-4)),0,7.5-(0.5*(frame-4))>,White)    //deplacement rectiligne ce fait selon l'axe X et Z
   #end

     #if(frame>=14 & frame<17)
      cavalier(<1.5,0,7.5-(0.5*frame)>,White)   //deplacement horizontal ou vertical
   #end

   #if(frame>=17 & frame<20)
      tour(<0.5+(0.5*(frame-4)),0,0.5+(0.5*(frame-4))>,Black)    //deplacement rectiligne ce fait selon l'axe X et Z
   #end


   #if(frame>=20 & frame<25)       //deplacement courbe de beziers
   #declare _t = (0.1*(frame-18));
   #declare _y0 = pow((1-_t),3)*_P0.y + 3*_t*pow((1-_t),2)*_P1.y+3*pow(_t,2)*(1-_t)*_P2.y+pow(_t,3)*_P3.y;
   #declare _z0 = pow((1-_t),3)*_P0.z + 3*_t*pow((1-_t),2)*_P1.z+3*pow(_t,2)*(1-_t)*_P2.z+pow(_t,3)*_P3.z;
   #declare coord = <0.5,_y0,_z0>;
   #declare tabCavalierPosition[2] = coord;
   #end

  #end
```