Binôme 1 :Alida NANA SUNJI KOUANANG
Binôme 2: Lala MAMMADOVA

Rapport du Projet des momies et des pyramides.

L'objectif du projet:

Le but de ce projet était de créer une momie placée dans une pyramide composée de plusieurs labyrinthes superposés et de l'aider à s'échapper du labyrinthe.

Introduction:

Pour ce projet, nous avons décidé de diviser le travail, Alida est chargée de la création de la momie, de son implémentation dans le labyrinthe et de la faire bouger. Lala à été chargé de faire la pyramide et le sol et en bonus le changement de niveau par téléportation.

Structuration du projet en détail:

Nous avons décidé de créer différentes fonctions dans des fichiers processing différents pour avoir une facilité de lisibilité. Notre projet est divisé en 9 fichiers processing interne, qui sont Labyrinthe(fichier principal), momie(fichier ou on dessiner la momie), caméra(fichier qui gère la caméra), Minimap (qui dessine la minimap de chaque niveau), bouger(qui fait bouger la momie), newLight(qui dessine le labyrinthe avec ses lumière), newLab(qui dessine un nouveau labyrinthe), sand(qui dessine le sable) et pyramid(qui dessine la pyramid). Le projet nous a pris environ 20-25 heures a faire et on a travaillé séparément mais en échangeant notre progrès régulièrement.

La Momie:

La momie à été créé à l'aide des PSHAPEs, parmis j'ai créé un Pshape group qui regroupe tous les différents pshape que je vais citer ci-dessus.

Les mains

```
mommie= createShape(GROUP);
float da=PI/25.0;
//creer la main droite
main1= loadShape("hand1.obj");
main1.scale(5);
main1.translate(-2,-185,190);

//creer la main gauche
main2= loadShape("hand2.obj");
main2.scale(5);
main2.translate(12,-185,190);
```

ici j'ai créé des pshapes pour chaque main et j'ai modifié son scale pour pouvoir bien positionner mes mains avec la grosseur que je voulais.

les yeux

```
//l'oeil de droite
eye1=createShape(SPHERE, 10 );
eye1.translate(-16,-280,40);
eye1.setStroke(false);

eyeball1=createShape(SPHERE, 5 );
eyeball1.translate(-16,-280,45);
eyeball1.setFill(color(0));

//l'oeil de gauche
eye2=createShape(SPHERE, 10 );
eye2.translate(16,-280,40);
eye2.setStroke(false);

eyeball2=createShape(SPHERE, 5 );
eyeball2.translate(16,-280,45);
eyeball2.setFill(color(0));
```

ici j'ai créé des sphères, 2 pour chaque oeil, j'ai changer des translates et la couleur pour le mettre dans ma momie

la tête

```
head= createShape();
head.beginShape(QUAD_STRIP);
head.noStroke();
head.rotateX(PI/2);
head.beginShape(QUAD_STRIP);
//head.translate(0,0,-200);
for (int i=-95; i<=60; i++){
  float a = i/20.0*2*PI;
  float b = i/30.0*2*PI;
  float n = 20+192*noise(i/10.);
  float re =250+cos(30.0);
  head.fill(n,n,50);
  float RBig = 15+35*cos(i*PI/200);
  float R2 = RBig+cos(b);
  head.vertex(R2*cos(a), R2*sin(a), re+i);
  RBig = 15+35*cos((i+25)*PI/200);
  R2 = RBig+cos(b);
  head.vertex(R2*cos(a+da), R2*sin(a+da),re+i+25);
}
head.endShape(CLOSE);
```

ici j'ai fait des QUAD_STRIP et dans ce code j'ai utilisé deux angles pour créer des bandelettes rapprochées pour qu'on ne voit pas au travers puis j'ai créé deux rayons que je fait varié pour dessiner des bandelettes à des hauteurs différentes. Ce code est inspiré du code du corps de ma momie.

le corps

```
//draw body without the head
body= createShape();
body.beginShape(QUAD_STRIP);
body.noStroke();
body.rotateX(PI/2);
for (int i=-200; i<=200; i++){
  float a = i/20.0*2*PI;
  float b = i/30.0*2*PI;
  float re = 3+cos(PI/6);
  float n = 20+192*noise(i/10.);
  body.fill(n,n,50);
  float RBig = 50+8*cos(i*PI/200);
  float R2 = RBig+cos(b);
  body.vertex(R2*cos(a), R2*sin(a),re+i);

  RBig = 50+8*cos((i+25)*PI/200);
  R2 = RBig+cos(b);
  body.vertex(R2*cos(a+da), R2*sin(a+da), re+(i+25));

}
body.endShape(CLOSE);
```

ici j'ai fait la même chose que pour mon corps mais j'ai modifié le RBig pour avoir un corps plus mince et long. La boucle aussi est plus longue pour créer un corps plus élancé.

les bras

```
//draw the left hand
hands= createShape();
hands.scale(0.8);
hands.beginShape(QUAD_STRIP);
hands.rotateX(-PI/35.);
hands.translate(-40,-180,50);
//hands.rotateZ(PI/2);
hands.noStroke();
int  ii=-90;
int max=0;
for(int j = 1;j<34;j++){
   for (int i=ii ; i<=50-max; i++){
    float a = i/50.0*2*PI;
    float n = 20+192*noise(i/10.);
    float re = 30+cos(30.0);
    hands.fill(n,n,50);
    float RBig,R2;
    RBig = 10+(max)+15*cos(i*PI/200);
    R2 = RBig*cos(a);
    hands.vertex(R2*cos(a), R2*sin(a), re+i);

    RBig = 10+(max)+15*cos((i+25)*PI/200);
    R2 = RBig*cos(a);
    hands.vertex(R2*cos(a), R2*sin(a),re+(i+25));
   }
     ii++;
     max++;
  }
```

ici j'ai fait une double boucle imbriquées où je modifie la boucle intérieure pour pouvoir dessiner des bandelettes sur des bandelettes et crée un effet gonflé sur mes bras. j'ai dessiné deux fois sur chaque bras pour avoir un effet de coud sur mon bras

```
//draw the right hand
hand= createShape();
hand.scale(0.8);
hand.beginShape(QUAD_STRIP);
hand.rotateX(-PI/35.);
hand.translate(15,-180,55);
hand.noStroke();
ii=-90;
max=0;
for(int j = 1;j<34;j++){
   for (int i=ii ; i<=50-max; i++){
    float a = i/50.0*2*PI;
    float n = 20+192*noise(i/10.);
    float re = 30+cos(30.0);
    hand.fill(n,n,50);
    float RBig,R2;
    RBig = 10+(max)+15*cos(i*PI/200);
    R2 = RBig*cos(a);
    hand.vertex(R2*cos(a), R2*sin(a), re+i);

    RBig = 10+(max)+15*cos((i+25)*PI/200);
    R2 = RBig*cos(a);
    hand.vertex(R2*cos(a), R2*sin(a),re+(i+25));

   }
     ii++;
```

```
      max++;
    }
   max=0;
   ii=-90;
  for(int j = 1;j<34;j++){
    for (int i=ii ; i<=50-max; i++){
      float a = i/50.0*2*PI;
      float n = 20+192*noise(i/10.);
      float re = 150+cos(30.0);
      hand.fill(n,n,50);
      float RBig,R2;
      RBig = 10+max+15*cos(i*PI/200);
      R2 = RBig*cos(a);
      hand.vertex(R2*cos(a), R2*sin(a), re+i);

      RBig = 10+max+15*cos((i+25)*PI/200);
      R2 = RBig*cos(a);
      hand.vertex(R2*cos(a), R2*sin(a),re+(i+25));

    }
          ii++;
          max++;
    }


     hand.endShape(CLOSE);
```

j’ai aussi changé la coordonnée de Z de chaque vertex pour positionner mon bras.

ma momie

```
mommie.addChild(body);
mommie.addChild(head);
mommie.addChild(hands);
mommie.addChild(hand);
mommie.addChild(eye1);
mommie.addChild(eyeball1);
mommie.addChild(eye2);
mommie.addChild(eyeball2);
mommie.addChild(main1);
mommie.addChild(main2);
```

ici j’ajoute chaque de mes parties dans mon PShape groupe.

le RÉSULTAT final

BougerMomie:

Cette fonction à pour but de bouger la momie aléatoirement dans le labyrinthe. Sa change de direction quand sa voit un obstacle(un mur), devant la momie. Aussi, quand la momie change de position, ça change aussi la forme de la momie avec de rotation par rapport à la direction de la momie.

```
void bougeMommie(int SIZE, int etage){
  int wallW=width/LAB_SIZE;
  int wallH=height/LAB_SIZE;

  if(dx==0 && dy==1){
    pushMatrix();
      translate(mx*wallW,my*wallH, -35);
      rotateX(-PI/2);
      //rotateY(PI);
      scale(0.1);
      shape(mommie);
      popMatrix();
  }else if (dx==-1 && dy==0){
    pushMatrix();
    translate(mx*wallW,my*wallH, -35);
    rotateX(-PI/2);
    rotateY(-PI/2);
    scale(0.1);
    shape(mommie);
    popMatrix();
  }else if(dx==1 && dy==0){
    pushMatrix();
    translate(mx*wallW,my*wallH, -35);
    rotateX(-PI/2);
    rotateY(PI/2);
    scale(0.1);
    shape(mommie);
    popMatrix();
  }else if (dx==0 && dy==-1){
      pushMatrix();
      translate(mx*wallW,my*wallH, -35);
      rotateX(-PI/2);
      rotateY(PI);
      scale(0.1);
      shape(mommie);
      popMatrix();
  }
  if(my+dy>=0 && my+dy<SIZE && mx+dx>=0 && mx+dx<SIZE &&
    labyrinthe[etage][my+dy][mx+dx]!='#' && dx==0 && dy==1){
       my=((frameCount/50)-(mx))%(SIZE-1);
       //println("boucle 1 mx= "+ mx+ " my= "+my);
    }
    else if(my+dy>=0 && my+dy<SIZE-1 && mx+dx>=0 && mx+dx<SIZE-1 &&
    labyrinthe[etage][my+dy][mx+dx]!='#' && dx==1 && dy==0){
        mx=((frameCount/50)-(my))%(SIZE-1);
       // println("boucle 2 mx+dx= "+ (mx+dx)+ " my+dy= "+(my+dy));
        if(my+dy<0 || my+dy>SIZE-1 || mx+dx<0 || mx+dx>SIZE-1){
          dy=int(random(-2,2));
          dx=int(random(-2,2));
        }

    }
```

```
  else if(my+dy>=0 && my+dy<SIZE && mx+dx>=0 && mx+dx<SIZE &&
  labyrinthe[etage][my+dy][mx+dx]!='#' && dx==0 && dy==-1){
      my=((frameCount/50)-(mx+count1))%(SIZE-1);
      count1++;
     // println("count1 : "+count1);
      //println("boucle 3 mx= "+ mx+ " my= "+my);
  }
  else if(my+dy>=0 && my+dy<SIZE && mx+dx>=0 && mx+dx<SIZE &&
      labyrinthe[etage][my+dy][mx+dx]!='#' && dx==-1 && dy==0){
      mx=((frameCount/50)-(my+count2))%(SIZE-1);
      count2++;
      //println("count2 : "+count2);
      //println("boucle 4 mx= "+ mx+ " my= "+my);
  }
 if(my+dy<0 || my+dy>=SIZE-1 || mx+dx<0 || mx+dx>=SIZE-1 || labyrinthe[etage][my+dy]
   // println("enter and dx= "+ dx + "dy "+ dy);
    dy=int(random(-2,2));
    dx=int(random(-2,2));
    if(dx==-1 && dy==0) count2=0;
    if(dx==0 && dy==-1) count1=0;
    //println("enter and dx= "+ dx + " dy "+ dy);
    // println("boucle 2 my = "+ my+ ", mx= "+mx);
  }
 // println((frameCount/50));
```

La pyramide:

Pour la création de chaque labyrinthe placé dans la pyramide on a utilisé le code qui était donné par notre professeur et on l'a mis dans le fichier newLab.

Pour la pyramide on a décidé de créer une boucle dans le fichier labyrintheFinale dans le void setup():

```
  //pyramide();
  int hauteur=0;
  int largeur=0;

  int j = 8;
  for(int i = LAB_SIZE ; i >=5; i-=2) {
    //drawLaby(j, i);
    Laby(j, i,hauteur,largeur);
    hauteur = hauteur+ 100;
    largeur-=30;
    println(i + ", " + j);
    j--;
  }
  hauteur=0;
  largeur=0;
}
```

La boucle for est utilisée pour dessiner plusieurs pyramides avec des tailles différentes. Les variables hauteur et largeur sont utilisées pour ajuster la position des pyramides dessinées.

Quand on sort de la pyramide, autour on a du sable. Et pour le créer on a fait ça:

```
float wallLen = 4000/LAB_SIZE;
PShape sand;
PImage texture2;

void sand() {

   texture2 = loadImage("sand.png");

   sand = createShape();
   sand.beginShape(QUAD);
   sand.texture(texture2);
   sand.noStroke();
   for(int j = -LAB_SIZE; j < LAB_SIZE*2; j++) {
     for(int i = -LAB_SIZE; i < LAB_SIZE*2; i++) {
       sand.vertex(wallLen * i-wallLen/2, wallLen * j-wallLen/2, -60+noise(-30, 30),
                           0, 0);
       sand.vertex(wallLen * i+wallLen/2, wallLen * j-wallLen/2, -60+noise(-30, 30),
                           0, 0*texture2.height);
       sand.vertex(wallLen *i + wallLen/2, wallLen * j + wallLen/2,-60+ noise(-30, 30),
                           texture2.width, texture2.height);
       sand.vertex(wallLen * i - wallLen/2, wallLen * j+wallLen/2, -60+noise(-30, 30),
                           texture2.width, 0);

     }
   }
   sand.endShape();
}
```

Ce code crée une forme 3D qui ressemble à du sable en utilisant une image de texture de sable.

Une mini-map:

Cette fonction permet de visualiser la structure du labyrinthe à l'aide d'une mini-map en 3D, ce qui peut être utile pour aider le joueur à se repérer et à naviguer dans le labyrinthe.

La fonction prend deux arguments : SIZE, qui représente la taille du labyrinthe, et étage, qui représente l'étage du labyrinthe à afficher.

Le code commence par définir la taille des murs en fonction de la taille de l'écran et de la taille du labyrinthe. Ensuite, pour chaque case du labyrinthe, la fonction vérifie si elle a été visitée

(stockée dans un tableau de booléens appelé déplace). Si la case a été visitée et qu'elle contient un mur (#), la fonction dessine un petit cube coloré représentant ce mur à sa position respective.

La fonction utilise également les coordonnées actuelles du joueur (posX, posY) pour dessiner une sphère verte représentant la position actuelle du joueur et une sphère jaune représentant la position de la souris (mx, my) sur la mini-map.

```
void drawMiniMap(int SIZE, int etage) {
  float wallW = width/SIZE;
  float wallH = height/SIZE;
  for (int j=0; j<SIZE; j++) {
    for (int i=0; i<SIZE; i++) {
     if( posY+dirY>-1 && posX+dirX>-1 && posX+dirX<SIZE-1 &&
     posY+dirY<SIZE-1 && inLab) {deplace[etage][posY+dirY][posX+dirX]=true;}
     if (labyrinthe[etage][j][i]=='#'&& deplace[etage][j][i]) {
        fill(i*25, j*25, 255-i*10+j*10);
        pushMatrix();
        translate(50+i*wallW/8, 50+j*wallH/8, 50);
        box(wallW/10, wallH/10, 5);
        popMatrix();
        if(i>0 && j>0 && j<SIZE-1 && i<SIZE-1){
              deplace[etage][j-1][i-1]=true;
              deplace[etage][j-1][i]=true;
              deplace[etage][j+1][i]=true;
              deplace[etage][j][i-1]= true;
              deplace[etage][j+1][i+1]= true;
              deplace[etage][j+1][i]=true;
              deplace[etage][j][i+1]=true;
         }
     }
    }
  }
  pushMatrix();
  fill(0, 255, 0);
  noStroke();
  translate(50+posX*wallW/8, 50+posY*wallH/8, 50);
  sphere(3);
  popMatrix();

  pushMatrix();
  fill(255, 255,0);
  noStroke();
  translate(50+mx*wallW/8, 50+my*wallH/8, 50);
  sphere(3);
  popMatrix();
}
```

Caméra:

Cette fonction ajuste la position et les paramètres de la caméra ainsi que les lumières utilisées en fonction de la situation dans laquelle se trouve le joueur.

La première partie du code calcule la largeur et la hauteur des murs en fonction de la taille de la grille, qui est déterminée par l'argument "SIZE". Si la variable "inLab" ou "outlab" est vraie, cela signifie que le joueur est à l'intérieur ou à l'extérieur d'un labyrinthe, respectivement. Dans ce cas, la perspective de la caméra est modifiée et la position et la direction de la caméra sont calculées en fonction de la position et de la direction du joueur, ainsi que de certaines animations (si elles sont activées). Si la variable "outlab" est vraie, cela signifie que le joueur est à l'extérieur du labyrinthe et une lumière ambiante et une lumière ponctuelle sont ajoutées. Sinon, la lumière ambiante est désactivée et seule une lumière ponctuelle est utilisée.
Si le joueur n'est ni à l'intérieur ni à l'extérieur d'un labyrinthe, la caméra est simplement positionnée à la position et à la

direction du joueur, et une lumière ponctuelle est utilisée.

```
void setCamera(int SIZE){
  float wallW = width/SIZE;
  float wallH = height/SIZE;
  if (inLab|| outlab) {
    perspective(2*PI/3, float(width)/float(height), 1, 10000);
    if (animT)
      camera((posX-dirX*anim/20.0)*wallW,      (posY-dirY*anim/20.0)*wallH,      -15+2*sin(anim*PI/5.0),
             (posX-dirX*anim/20.0+dirX)*wallW, (posY-dirY*anim/20.0+dirY)*wallH, -15+4*sin(anim*PI/5.0), 0, 0, -1);
    else if (animR)
      camera(posX*wallW, posY*wallH, -15,
            (posX+(odirX*anim+dirX*(20-anim))/20.0)*wallW, (posY+(odirY*anim+dirY*(20-anim))/20.0)*wallH, -15-5*sin(anim*PI/20.0), 0, 0, -1);
    else {
      camera(posX*wallW, posY*wallH, -15,
             (posX+dirX)*wallW, (posY+dirY)*wallH, -15, 0, 0, -1);
    }
    //camera((posX-dirX*anim/20.0)*wallW, (posY-dirY*anim/20.0)*wallH, -15+6*sin(anim*PI/20.0),
    //  (posX+dirX-dirX*anim/20.0)*wallW, (posY+dirY-dirY*anim/20.0)*wallH, -15+10*sin(anim*PI/20.0), 0, 0, -1);

    //lightFalloff(0.0, 0.01, 0.0001);
    //spotLight(255, 255, 255,
              ///posX*wallW, posY*wallH, -15,
              //dirX, dirY, -1,
              //PI/2.0, 1);
      if(outlab){
           ambientLight(50,50,50);
           pointLight(255, 255, 255, posX*wallW, posY*wallH, 15);
      }else{
          lightFalloff(0.0, 0.01, 0.0001);
          //ambientLight(15,15,15);
          pointLight(255, 255, 255, posX*wallW, posY*wallH, 15);
        }
  } else{
    lightFalloff(0.0, 0.05, 0.0001);
    pointLight(255, 255, 255, posX*wallW, posY*wallH, 15);
    camera(posX*wallW, posY*wallH, -15,
             (posX+dirX)*wallW, (posY+dirY)*wallH, -15, 0, 0, -1);
  }
}
```

Sol:

```
PImage  texture0;
PImage  texture1;
PImage img;

void setup() {
  pixelDensity(2);
  randomSeed(2);

  // creation de ma
  Mommie();

  texture0 = loadImage("stones.jpg");
  texture1 = loadImage ("sol.jpg");
  img= loadImage("sky.jpg");
```

Pour afficher le sol on utilise une seul ligne de code dans void setup() de labyrintheFinal

NewLight:

Cette fonction affiche un labyrinthe 3D à l'écran, avec des murs et des passages.

La première étape est de calculer la largeur et la hauteur de chaque mur en fonction de la taille de l'écran et du nombre de carrés dans le labyrinthe.

La fonction utilise ensuite deux boucles pour parcourir chaque carré du labyrinthe et dessiner des murs autour de chacun d'eux. Elle utilise la fonction "beginShape" pour commencer à dessiner chaque mur.

Pour chaque mur, la fonction vérifie si le carré adjacent à un mur dans cette direction et dessine un mur s'il y en a un. La fonction utilise la fonction "pushMatrix" pour enregistrer la position et l'orientation courantes, puis la fonction "translate" pour déplacer la position de dessin en fonction de la position du mur et de sa direction. La fonction "fill" est utilisée pour définir la couleur du mur, qui dépend de la position du mur dans le labyrinthe.

La fonction utilise également des formes en 3D pour dessiner le plafond et le sol du labyrinthe, qui sont stockées dans les tableaux "laby1", "laby0" et "ceiling0". Ces formes sont dessinées à la fin de chaque rangée de carrés pour que les murs soient dessinés en premier et apparaissent devant le plafond et le sol.

Il y a également des blocs de code qui sont conditionnels, basés sur des variables booléennes "inLab" et "outlab". Ces blocs de code permettent de dessiner des éléments supplémentaires dans le labyrinthe si ces variables sont "true". Par exemple, si "outlab" est "true", la fonction dessinera une forme en 3D pour représenter le sol en sable du labyrinthe.

```
void lightLaby(int etage, int SIZE){
  //println(SIZE);
   wallW = width/SIZE;
   wallH = height/SIZE;
  for (int j=0; j<SIZE; j++) {
    for (int i=0; i<SIZE; i++) {
      pushMatrix();
      translate(i*wallW, j*wallH, 0);
      if (labyrinthe[etage][j][i]=='#') {
        beginShape(QUADS);
        if (sides[etage][j][i][3]==1) {
          pushMatrix();
          translate(0, -wallH/2, 40+Hauteur);
          if (i==posX || j==posY) {
            fill(i*25, j*25, 255-i*10+j*10);
            //sphere(5);
            //spotLight(i*25, j*25, 255-i*10+j*10, 0, -15, 15, 0, 0, -1, PI/4, 1);
          } else {
            fill(128);
            //sphere(15);
          }
          popMatrix();
        }

        if (sides[etage][j][i][0]==1) {
          pushMatrix();
          translate(0, wallH/2, 40+Hauteur);
          if (i==posX || j==posY) {
            fill(i*25, j*25, 255-i*10+j*10);
            //sphere(5);
            //spotLight(i*25, j*25, 255-i*10+j*10, 0, -15, 15, 0, 0, -1, PI/4, 1);
          } else {
            fill(128);
            //sphere(15);
          }
          popMatrix();
        }

         if (sides[etage][j][i][1]==1) {
          pushMatrix();
          translate(-wallW/2, 0, 40+Hauteur);
          if (i==posX || j==posY) {
```

```
         if (sides[etage][j][i][1]==1) {
          pushMatrix();
          translate(-wallW/2, 0, 40+Hauteur);
          if (i==posX || j==posY) {
            fill(i*25, j*25, 255-i*10+j*10);
            //sphere(5);
            //spotLight(i*25, j*25, 255-i*10+j*10, 0, -15, 15, 0, 0, -1, PI/4, 1);
          } else {
            fill(128);
            //sphere(15);
          }
          popMatrix();
        }

        if (sides[etage][j][i][2]==1) {
          pushMatrix();
          translate(0, wallH/2, 40+Hauteur);
          if (i==posX || j==posY) {
            fill(i*25, j*25, 255-i*10+j*10);
            //sphere(5);
            //spotLight(i*25, j*25, 255-i*10+j*10, 0, -15, 15, 0, 0, -1, PI/4, 1);
          } else {
            fill(128);
            //sphere(15);
          }
          popMatrix();
        }
      }
      popMatrix();
    }
    if(inLab){
        shape(laby1[etage], 0, 0);
        shape(laby0[etage], 0, 0);
        shape(ceiling0[etage], 0, 1);
    }
     if (outlab){
        shape(laby1[etage], 0, 0);
        shape(laby0[etage], 0, 0);
        //if(SIZE==21)
        shape(sand,0,0);
        shape(ceiling0[etage], 0, 1);
```

Labyrinthes:

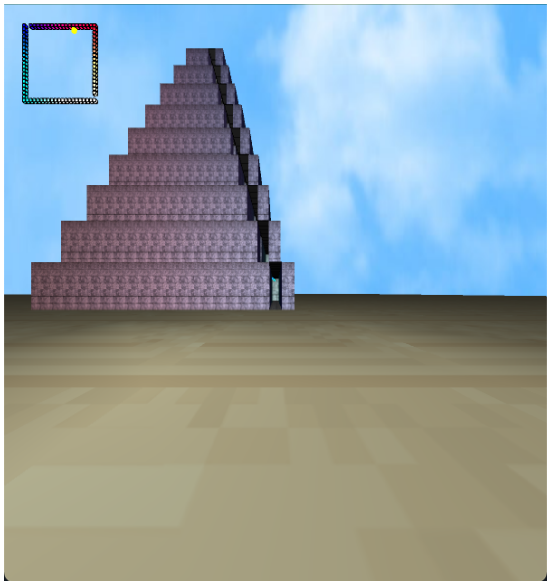
Ce code utilise comme base le code pour créer le labyrinthe en ajoutant quelques changements.

```
void keyPressed() {
  if(key=='f') {
    outlab=false;
    inLab=true;
    posX=oldPx;
    posY=oldPy;
    mx=1;
    my=0;
    currentFloor=oldcurrentFloor;
  }
  if(key=='o') {
      inLab = !inLab;
      outlab=true;
      oldPx=posX;
      oldPy=posY;
      posX=1;
      posY=-1;
      mx=1;
      my=0;
      dirX = 0;
      dirY = 1;
      oldcurrentFloor=currentFloor;
      currentFloor=8;
  }
```

Ici on a utilisé les différentes touches et leur fonctionnement.

1. La touche “o” nous fait sortir du labyrinthe.
2. La touche “f” nous fait entrer dans le labyrinthe.
3. La touche “Spacebar” nous fait changer de niveau si et seulement si nous sommes à la sortie du labyrinthe.

Le résultat final:



Difficultés rencontrés:

Lors de ce projet nous avons rencontré beaucoup de difficultés. ses difficulté était:

1. Dessiner la momie est pouvoir trouver les bonne rotations pour pouvoir la voir.
2. Dessiner la pyramide, faire en sorte que ce soit des empilement de labyrinthes.
3. Placer la momie dans le labyrinthe, bien trouver des rotation pour la voir dans le labyrinthe.
4. Faire le changement de niveau et accéder à chaque mini labyrinthe.
5. Pouvoir entrer et sortir du labyrinthe sans que sa affecte la momie.

Les Bonus:

On a ajouté quelques point bonus:

1. Notre mini-map n'est pas visible sauf quand notre personnage bouge et "découvre" le labyrinthe.
2. Le changement de niveau par téléportation ("Spacebar")

Les limitations

Voici quelques limitations que nous avons eu

1. mettre un toit à notre pyramid
2. Faire la boussole
3. Faire que la momie poursuive le fantôme dans le labyrinthe.