Code du projet à titre indicatif :

ColorPanic.pde :

```
//----------------------BEGIN-IMPORT----------------------------------------------------------------------------------------------------------------------------------------------------------------//


import ddf.minim.*;

import ddf.minim.analysis.*;

import ddf.minim.effects.*;

import ddf.minim.signals.*;

import ddf.minim.spi.*;

import ddf.minim.ugens.*;


import gifAnimation.*;


import java.io.*;

import java.lang.*;

//----------------------END-IMPORT------------------------------------------------------------------------------------------------------------------------------------------------------------------//


//--------BEGIN-CREATION-GLOBAL-VARIABLES-----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------//

char GAUCHE='q', DROITE='d', HAUT='z', BAS='s', ACTION='j', PAUSE='p' ;

PImage heroJumpBlueR, heroJumpBlueL, heroJumpGreenR, heroJumpGreenL, heroJumpBlackR,
heroJumpBlackL, heroJumpPurpleR, heroJumpPurpleL, heroJumpRedR, heroJumpRedL,
heroJumpVeridianR, heroJumpVeridianL, heroJumpGSwapR, heroJumpGSwapL;

PImage heroIdleBlueL, heroIdleBlueR, heroIdleGreenL, heroIdleGreenR, heroIdleBlackL,
heroIdleBlackR, heroIdlePurpleL, heroIdlePurpleR, heroIdleRedL, heroIdleRedR, heroIdleVeridianL,
heroIdleVeridianR, heroIdleGSwapL, heroIdleGSwapR;

PImage menuPNG, menuEmpty, timer, deathPNG, coin;

PImage[] lvl=new PImage[9999];

String [][]hitboxLvl=new String[9999][];


PFont font, arial;
```

```
int levelNumber=0, initialTime=0;
int second=0, minute=0, hour=0, millisPaused=0, timeStopped, millis;

AudioPlayer actualMusic;//chargement de seulement 5 variables pour le son sinon ce que j'imagine

//AudioSample mvtInterfaceANDdeath;//être le buffer de la raspberry est surchargé et refuse
//AudioSample validationInterfaceANDpowerup;//de charger des sons supplémentaires
//AudioSample jump;
//AudioSample bonusSFX;

AudioSample jump;//et GSwap
AudioSample TP;//et dash
AudioSample powerup;
AudioSample death;
AudioSample validationInterface;
AudioSample mvtInterface;


Hero hero=new Hero();
BonusDoubleJump bonusDoubleJump=new BonusDoubleJump();
BonusDash bonusDash=new BonusDash();
BonusNoClip bonusNoClip=new BonusNoClip();
BonusTP bonusTP=new BonusTP();
BonusGravitySwap bonusGravitySwap= new BonusGravitySwap();
BonusPoints bonusPoints= new BonusPoints();
Sound sound=new Sound();
Interface interfaces=new Interface();
Script script=new Script();

Minim minim;
```

```
Gif heroBlueR, heroBlueL, heroGreenR, heroGreenL, heroBlackR, heroBlackL, heroPurpleR,
heroPurpleL, heroRedR, heroRedL, heroVeridianR, heroVeridianL, heroGSwapL, heroGSwapR;

Gif powerupN, powerupR, powerupB, powerupG, powerupP, powerupV, powerupPoints;

Gif heroDashR, heroDashL;

Gif heroTPR1, heroTPL1, heroTPR2, heroTPL2;

Gif heroDeadBlue, heroDeadGreen, heroDeadBlack, heroDeadPurple, heroDeadRed,
heroDeadVeridian;

Gif menu;


//---------------------------END-CREATION-GLOBAL-VARIABLES-----------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
-----//


void setup() {
  //fullScreen();
  size(1024,600);
  menuPNG =loadImage("data/menu.png");
  menuEmpty=loadImage("data/menuEmpty.png");
  image(menuEmpty,0,0);


  //----------------------BEGIN-GIF-DEFINITION-------------------------------------------------------------------
//
  heroBlueR = new Gif(this, "data/BOB_SPRITE/BOBWALK_B_R.gif");
  heroBlueR.play();
  heroBlueL = new Gif(this, "data/BOB_SPRITE/BOBWALK_B_L.gif");
  heroBlueL.play();
  heroGreenR = new Gif(this, "data/BOB_SPRITE/BOBWALK_G_R.gif");
  heroGreenR.play();
  heroGreenL = new Gif(this, "data/BOB_SPRITE/BOBWALK_G_L.gif");
  heroGreenL.play();
  heroBlackR = new Gif(this, "data/BOB_SPRITE/BOBWALK_N_R.gif");
  heroBlackR.play();
```

```
heroBlackL= new Gif(this, "data/BOB_SPRITE/BOBWALK_N_L.gif");
heroBlackL.play();
heroPurpleR= new Gif(this, "data/BOB_SPRITE/BOBWALK_P_R.gif");
heroPurpleR.play();
heroPurpleL= new Gif(this, "data/BOB_SPRITE/BOBWALK_P_L.gif");
heroPurpleL.play();
heroRedR= new Gif(this, "data/BOB_SPRITE/BOBWALK_R_R.gif");
heroRedR.play();
heroRedL= new Gif(this, "data/BOB_SPRITE/BOBWALK_R_L.gif");
heroRedL.play();
heroVeridianR= new Gif(this, "data/BOB_SPRITE/BOBWALK_V_R.gif");
heroVeridianR.play();
heroVeridianL= new Gif(this, "data/BOB_SPRITE/BOBWALK_V_L.gif");
heroVeridianL.play();
heroDashR=new Gif(this, "data/BOB_SPRITE/BOBDASH/BOBDASH_R.gif");
heroDashR.play();
heroDashL=new Gif(this, "data/BOB_SPRITE/BOBDASH/BOBDASH_L.gif");
heroDashL.play();
powerupN = new Gif(this, "data/powerup/Powerup_BLACK.gif");
powerupN.play();
powerupR = new Gif(this, "data/powerup/Powerup_RED.gif");
powerupR.play();
powerupB= new Gif(this, "data/powerup/Powerup_BLUE.gif");
powerupB.play();
powerupG= new Gif(this, "data/powerup/Powerup_GREEN.gif");
powerupG.play();
powerupP= new Gif(this, "data/powerup/Powerup_PURPLE.gif");
powerupP.play();
powerupV= new Gif(this, "data/powerup/Powerup_VIRIDIAN.gif");
powerupV.play();
powerupPoints= new Gif(this, "data/powerup/Powerup_Points.gif");
```

```
 powerupPoints.play();
 heroDeadBlue=new Gif(this, "data/BOB_SPRITE/BOBDEAD/BOBDEAD_BLUE.gif");
 heroDeadBlue.play();
 heroDeadGreen=new Gif(this, "data/BOB_SPRITE/BOBDEAD/BOBDEAD_GREEN.gif");
 heroDeadGreen.play();
 heroDeadBlack=new Gif(this, "data/BOB_SPRITE/BOBDEAD/BOBDEAD_BLACK.gif");
 heroDeadBlack.play();
 heroDeadPurple=new Gif(this, "data/BOB_SPRITE/BOBDEAD/BOBDEAD_PURPLE.gif");
 heroDeadPurple.play();
 heroDeadRed=new Gif(this, "data/BOB_SPRITE/BOBDEAD/BOBDEAD_RED.gif");
 heroDeadRed.play();
 heroDeadVeridian=new Gif(this, "data/BOB_SPRITE/BOBDEAD/BOBDEAD_VIRIDIAN.gif");
 heroDeadVeridian.play();
 heroGSwapR=new Gif(this, "data/BOB_SPRITE/BOBG-SWAP/BOBG-SWAP_WALK_R.gif");
 heroGSwapR.play();
 heroGSwapL=new Gif(this, "data/BOB_SPRITE/BOBG-SWAP/BOBG-SWAP_WALK_L.gif");
 heroGSwapL.play();
 heroTPR1=new Gif(this, "data/BOB_SPRITE/BOBTP/TP_RIGHT1.gif");
 heroTPR1.play();
 heroTPR2=new Gif(this, "data/BOB_SPRITE/BOBTP/TP_RIGHT2.gif");
 heroTPR2.play();
 heroTPL1=new Gif(this, "data/BOB_SPRITE/BOBTP/TP_LEFT1.gif");
 heroTPL1.play();
 heroTPL2=new Gif(this, "data/BOB_SPRITE/BOBTP/TP_LEFT2.gif");
 heroTPL2.play();
 menu=new Gif(this, "data/menu.gif");
 menu.play();
 //------------------------END-GIF-DEFINITION-------------------------------------------------------------------//


 //------------------------BEGIN-LEVELS-DEFINITION-------------------------------------------------------------------
//
```

```
 File file = new File(dataPath("levels/lvlsHitbox"));
 String[] lvls = file.list();
 for (int i=0; i<lvls.length; i++) {
  String actualLevel="lvl"+str(i+1);
  lvl[i]=loadImage("data/levels/lvlsPrint/"+actualLevel+".png");
  hitboxLvl[i]=loadStrings("data/levels/lvlsHitbox/"+actualLevel+".txt");
 }
 //------------------------END-LEVELS-DEFINITION-----------------------------------------------------------------------------
//


 //----------------------BEGIN-PNG-DEFINITION-----------------------------------------------------------------------------
//
 heroJumpBlueR=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_B_R.png");
 heroJumpBlueL=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_B_L.png");
 heroJumpGreenR=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_G_R.png");
 heroJumpGreenL=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_G_L.png");
 heroJumpBlackR=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_N_R.png");
 heroJumpBlackL=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_N_L.png");
 heroJumpPurpleR=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_P_R.png");
 heroJumpPurpleL=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_P_L.png");
 heroJumpRedR=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_R_R.png");
 heroJumpRedL=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_R_L.png");
 heroJumpVeridianR=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_V_R.png");
 heroJumpVeridianL=loadImage("data/BOB_SPRITE/BOBJUMP/BOBJUMP_V_L.png");

 heroJumpGSwapR=loadImage("data/BOB_SPRITE/BOBG-SWAP/BOBG-SWAP_JUMP_R.png");
 heroJumpGSwapL=loadImage("data/BOB_SPRITE/BOBG-SWAP/BOBG-SWAP_JUMP_L.png");
 heroIdleGSwapR=loadImage("data/BOB_SPRITE/BOBG-SWAP/BOBG-SWAP_IDLE_R.png");
 heroIdleGSwapL=loadImage("data/BOB_SPRITE/BOBG-SWAP/BOBG-SWAP_IDLE_L.png");

 heroIdleBlueL=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_B_LEFT.png");
 heroIdleBlueR=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_B_RIGHT.png");
```

```
 heroIdleGreenL=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_G_LEFT.png");
 heroIdleGreenR=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_G_RIGHT.png");
 heroIdleBlackL=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_N_LEFT.png");
 heroIdleBlackR=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_N_RIGHT.png");
 heroIdlePurpleL=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_P_LEFT.png");
 heroIdlePurpleR=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_P_RIGHT.png");
 heroIdleRedL=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_R_LEFT.png");
 heroIdleRedR=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_R_RIGHT.png");
 heroIdleVeridianL=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_V_LEFT.png");
 heroIdleVeridianR=loadImage("data/BOB_SPRITE/BOBIDLE/BOBIDLE_V_RIGHT.png");
 //----------------------END-PNG-DEFINITION-----------------------------------------------------------------------
//


 //----------------------BEGIN-MUSIC-DEFINITION-------------------------------------------------------------------
-//
 minim= new Minim(this);
 //musicBurn= minim.loadFile("Sound/Music/Burn.ogg");
 //musicColorPanic=minim.loadFile("Sound/Music/ColorPanic.ogg");
 //musicJourneyBegin=minim.loadFile("Sound/Music/JourneyBegin.ogg");
 //musicNewPower=minim.loadFile("Sound/Music/NewPower.ogg");
 //musicRetroRide=minim.loadFile("Sound/Music/RetroRide.ogg");
 //musicRise=minim.loadFile("Sound/Music/Rise.ogg");
 //musicTheLastBattle=minim.loadFile("Sound/Music/TheLastBattle.ogg");
 //musicTheOne=minim.loadFile("Sound/Music/TheOne.ogg");
 //musicValk=minim.loadFile("Sound/Music/Valk.ogg");
 //actualMusic=minim.loadFile("Sound/Music/musicRetroRide.ogg");
 //--------------------END-MUSIC-DEFINITION-----------------------------------------------------------------------
//


 //-------------------BEGIN-SFX-DEFINITION-----------------------------------------------------------------------
//
 validationInterface=minim.loadSample("Sound/SFX/validation.mp3");
```

```
  mvtInterface=minim.loadSample("Sound/SFX/mvtInterface.mp3");
  //--------------------END-SFX-DEFINITION-----------------------------------------------------------------------------//
  timer=loadImage("data/timer.png");
  deathPNG=loadImage("data/death.png");
  coin=loadImage("data/coin.png");

  font = createFont("Super Mario Bros. NES.ttf", 22);
  arial = createFont("Arial", 21);
  actualMusic=sound.musicBegin("data/Sound/Music/musicRetroRide.mp3");
  interfaces.playerBase=loadStrings("data/playerBase.txt");
}

void draw() {
  if (interfaces.firstScreen==true) {
    image(menu, 0, 0);
  } else if (interfaces.ecranTitre==true) {
    interfaces.ecranTitre();
  } else if (interfaces.ecranTitre==false && interfaces.setUsername==true) {
    image(menuEmpty, 0, 0);
    textSize(13);
    fill(255, 255, 255);
    interfaces.visualKeyboard();
  } else if (interfaces.ecranTitre==false && interfaces.load==true) {
    image(menuEmpty, 0, 0);
    interfaces.load();
  } else if (interfaces.importing==true) {
    interfaces.importing();
  } else if (interfaces.exporting==true) {
    interfaces.exporting();
  } else if (interfaces.ecranTitre==false && interfaces.leaderboard==true) {
    image(menuEmpty, 0, 0);
```

```
  interfaces.leaderboard();
  interfaces.visualLeaderboard();
 } else if (interfaces.ecranTitre==false && interfaces.credit==true) {
  image(menuEmpty, 0, 0);
  interfaces.credits();
 } else if (interfaces.save==true) {
  image(menuEmpty, 0, 0);
  interfaces.save();
 } else if (interfaces.pause==true) {
  interfaces.pause();
 } else if (levelNumber==59) {
  interfaces.endGame();
 } else {
  image(lvl[levelNumber], 0, 0);
  sound.musicLoop();
  sound.musicFirst();
  hero.dessin();
  hero.recommencer();
  hero.contactEnd();
  hero.TP();

  deplacements("PI");
  hero.dash();
  hero.confirmPosition();
  bonusDoubleJump.animation();
  bonusDash.animation();
  bonusGravitySwap.animation();
  bonusNoClip.animation();
  bonusTP.animation();
  bonusPoints.animation();
  textSize(22);
```

```
  if (((interfaces.firstLoadedMillis+millis()-initialTime)-timeStopped)%1000<=45) {
   second++;
   if (interfaces.stepMillisIntegration==0) {
    interfaces.firstLoadedMillis=-millis()+initialTime+timeStopped;
    interfaces.stepMillisIntegration++;
    interfaces.is1000=1;
   }
   if (interfaces.stepMillisIntegration==1) {
    interfaces.firstLoadedMillis=0;
    interfaces.stepMillisIntegration=2;
   }
  }
  if (second>=60) {
   minute++;
   second-=60;
  }
  if (minute>=60) {
   hour++;
   minute-=60;
  }
  fill(#E4E823);
  image(timer, 400, 5);
  text(hour+":"+minute+":"+second+":"+(((millis()-initialTime)-
timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute
*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-
(minute*60000)-(hour*3600000)), 445, 35);//880
  image(deathPNG, 5, 5);
  text(hero.nbMort, 50, 35);
  image(coin, 5, 55);
  text(bonusPoints.nbPoints, 50, 88);
  fill(255, 255, 255);
  hero.contactSpike();
```

```
    hero.contactEnd();
  }
}

void deplacements(String console) {
  if (console=="PC") {
    if (bonusDash.trigDash==false || bonusTP.trigTP==false) {
      hero.mvtHaut();
      hero.distanceSol();
      if (bonusGravitySwap.timeActivationGSwap==true && hero.nbMontee!=0) {
        hero.reset=true;
      } else if (bonusGravitySwap.timeActivationGSwap==true && hero.nbMontee==0) {
        bonusGravitySwap.timeActivationGSwap=false;
      }
      if (hero.nbMontee==0 && bonusDash.trigDash==false) {
        hero.descente();
      }
      if (hero.nbDescente==0) {
        hero.montee();
        hero.montee();
      }
    }
    hero.mvtGauche();
    hero.mvtDroite();
  } else if (console=="PI") {
    if (bonusDash.trigDash==false || bonusTP.trigTP==false) {
      hero.mvtHaut();
      hero.distanceSol();
      if (bonusGravitySwap.timeActivationGSwap==true && hero.nbMontee!=0) {
        hero.reset=true;
      } else if (bonusGravitySwap.timeActivationGSwap==true && hero.nbMontee==0) {
```

```
      bonusGravitySwap.timeActivationGSwap=false;
    }
    if (hero.nbMontee==0) {
      hero.descente();
      hero.descente();
    }
    if (hero.nbDescente==0) {
      hero.montee();
      hero.montee();
      hero.montee();
      hero.montee();
    }
  }
  hero.mvtGauche();
  hero.mvtGauche();
  hero.mvtDroite();
  hero.mvtDroite();
 }
}


void keyReleased() {
  if (hero.VIEHasBeenFalse==false && interfaces.setUsername==false && interfaces.load==false &&
interfaces.ecranTitre==false && interfaces.leaderboard==false && interfaces.save==false &&
interfaces.pause==false && interfaces.importing==false && interfaces.exporting==false) {//si on est
dans aucunes interfaces
  if (key==GAUCHE && key!=HAUT) {//on relâche la touche gauche, arrête le mouvement
    hero.gauche=false;
    hero.droite=false;
  }
  if (key==DROITE && key!=HAUT) {//on relâche la touche droite,arrête le mouvement
    hero.droite=false;
    hero.gauche=false;
```

```
   }
  }//end if
}
void keyPressed() {
  if (key=='m') {
    interfaces.load();
  }
  if (interfaces.firstScreen==true) {//actions disponibles pour le premier écran du jeu
    if (key==PAUSE) {
      interfaces.firstScreen=false;
      interfaces.ecranTitre=true;
    }
  } else if (interfaces.ecranTitre==false && interfaces.setUsername==false && interfaces.load==false
&& interfaces.leaderboard==false && interfaces.save==false && interfaces.pause==false &&
interfaces.importing==false && interfaces.exporting==false) {//actions disponibles en jeu
    if (hero.VIEHasBeenFalse==false && hero.TPActivationP1==false && hero.TPActivationP2==false
&& hero.VIE==true) { // ne peut réaliser une action dans le jeu que si le hero est en vie et qu'il n'est
pas en plein tp
      if (((key==HAUT && hero.saut==false && bonusGravitySwap.GSwap==false) &&
((bonusGravitySwap.trigGSwap==false &&
hitboxLvl[levelNumber][hero.heroPos1+128].startsWith("wall")
||hitboxLvl[levelNumber][hero.heroPos2+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos3+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos4+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos5+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos6+128].startsWith("wall")) ||
(bonusGravitySwap.trigGSwap==true && hitboxLvl[levelNumber][hero.heroPos18-
128].startsWith("wall") ||hitboxLvl[levelNumber][hero.heroPos17-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos16-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos15-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos14-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos13-128].startsWith("wall"))))) {// si on touche le sol et que l'on
a pas de bonus pour inverser la gravité et que l'on souhaite sauter alors
       hero.saut=true;
       jump.trigger();//son du saut
      }
     if (key==HAUT  && bonusDoubleJump.doubleJump==true && hero.jumping==true &&
bonusGravitySwap.GSwap==false) { // si on souhaite sauter en l'air quand on a le bonus double jump
```

```
        bonusDoubleJump.trigDoubleJump=true;
        jump.trigger();//son du saut
      }
     if (key==ACTION) {
       if (bonusDash.dash==true && bonusDash.canDash==true) {//si on a le bonus pour dash et que l'on a pas déjà dash une fois en l'air
         bonusDash.trigDash=true;
         bonusDash.canDash=false;
         TP.trigger();
       }
       if (bonusTP.bonusTP==true && bonusTP.canTP==true) {//si on a le bonus tp et que l'on pas déjà tp une fois en l'air
         bonusTP.trigTP=true;
         bonusTP.canTP=false;
         TP.trigger();//son du tp
       }
       //-----------------------------------------------------------------DEBUT-BONUS-NON-FONCTIONNEL-----------------------------------------------------//
       if (bonusNoClip.noClip==true && bonusNoClip.trigNoClip==false) {
         bonusNoClip.trigNoClip=true;
       } else if (bonusNoClip.noClip==true && bonusNoClip.trigNoClip==true) {
         bonusNoClip.trigNoClip=false;
       }
       //-------------------------------------------------------------FIN-BONUS-NON-FONCTIONNEL-------------------------------------------//
       if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false && hero.jumping==false) { //si on a le bonus d'inversion de gravité, sans sauter, et que la gravité n'est pas inversé alors on l'inverse
         bonusGravitySwap.trigGSwap=true;
         bonusGravitySwap.timeActivationGSwap=true;
         jump.trigger(); //son inversion de gravité
       } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==true && hero.jumping==false) {//si on a le bonus d'inversion de gravité, sans sauter, et que la gravité n'est inversé alors on la remet normalement
```

```
        bonusGravitySwap.trigGSwap=false;
        jump.trigger();//son inversion de gravité
       }
     }
    if (key==PAUSE && isLooping() && interfaces.pause==false) {// si on appuie sur la touche de pause et que le jeu tourne, le met en pause
       millisPaused=millis();//récupère le temps auquel le jeu s'est mis en pause
       interfaces.pause=true;
       interfaces.pause();
    } else if (key==PAUSE && interfaces.pause==true) {//si on appuie sur la touche de pause et que le jeu ne tourne pas, le relance
       timeStopped+=millis()-millisPaused;//recupère la valeur où le jeu était en pause
       interfaces.pause=false;
    }
   }
  } else if (interfaces.ecranTitre==true || interfaces.setUsername==true || interfaces.load==true || interfaces.leaderboard==true || interfaces.save==true ||interfaces.pause==true ||interfaces.importing==true || interfaces.exporting==true) {// actions disponibles dans une interface
   if (key==HAUT ||key==BAS ||key==GAUCHE ||key==DROITE) {
    interfaces.leftAction();
    interfaces.rightAction();
    interfaces.topAction();
    interfaces.bottomAction();
    mvtInterface.trigger();//son du mouvement dans les interfaces
   } else if (key==ACTION) {
    interfaces.confirmAction();
    validationInterface.trigger();//son de validation dans les interfaces
   }
  }
}
```

```
void keyTyped() {

  if (interfaces.setUsername==false && interfaces.ecranTitre==false && interfaces.load==false && interfaces.leaderboard==false && interfaces.save==false && interfaces.pause==false) {//si on est dans aucune interface

    if (hero.VIEHasBeenFalse==false && hero.TPActivationP2==false && hero.TPActivationP1==false && hero.DashActivation==false) {//et que l'on est en vie

      if (key==GAUCHE) {//lance le mouvement vers la gauche

        hero.lastMove=GAUCHE;

        hero.droite=false;

        hero.gauche=true;

      }


      if (key==DROITE) {//lance le mouvement vers la droite

        hero.lastMove=DROITE;

        hero.gauche=false;

        hero.droite=true;

      }

    }

  }

}
```

Bonus.pde :

```
class Bonus{//classe qui sera hérité pour chacun des bonus
  protected int x, y;


  public int nbElem(String bonus) {//compte le nombre de cases où il y a un bonus dans le niveau courant
    int nbElems=0;
    for (int i=0; i<128*38; i++) {
      if (hitboxLvl[levelNumber][i].equals(bonus)) {
        nbElems++;
      }
    }
    return nbElems;
  }
  public int[] position(String bonus) { //récupère la position en x, si le numéro de la case est pair, en y, si le numéro de la case est impair
    int []tableau=new int[nbElem(bonus)*2];
    //println(tableau.length);
    int j=0;
    for (int i=0; i<128*38; i++) {
      if (hitboxLvl[levelNumber][i].equals(bonus)) {
        tableau[j]=(i%128)*8;
        tableau[j+1]=(i/128)*16;
        j+=2;
      }
    }
    return tableau;
  }
  public void animation(Gif anime, String bonus) {// positionne l'image du bonus en x,y
    for (int i=0; i<position(bonus).length; i+=2) {
      image(anime, position(bonus)[i], position(bonus)[i+1]);
    }
```

```
  }
}
```

BonusDash.pde :

```
class BonusDash extends Bonus {
  public boolean dash=false, trigDash=false,canDash=false;

  public void animation() {//positionne l'animation de la pièce de bonus dash
    super.animation(powerupR, "BDash");
  }
}
```

BonusDoubleJump.pde :

```
class BonusDoubleJump extends Bonus {
  public boolean doubleJump=false, trigDoubleJump=false, doubleJumpOn=false;
  public void animation() {//positionne l'animation de la pièce de bonus double jump
    super.animation(powerupG,"BJump");
  }
}
```

BonusGravitySwap.pde :

```
class BonusGravitySwap extends Bonus {
  public boolean GSwap=false,trigGSwap=false,timeActivationGSwap=false;

  public void animation() {//positionne l'animation du bonus d'inversion de gravité
    super.animation(powerupV,"BGSwap");
  }
}
```

BonusNoClip.pde :

```
//--------------------------------------DEBUT-BONUS-NON-FONCTIONNEL-----------------------------//
class BonusNoClip extends Bonus {
  public boolean noClip=false,trigNoClip=false;
```

```
  public void animation() {
    super.animation(powerupN,"BnoClip");
  }
}
//-----------------------------------FIN-BONUS-NON-FONCTIONNEL---------------------------------//
```

bonusPoints.pde :

```
class BonusPoints extends Bonus {
  public boolean Points=false;
  public int nbPoints=0,nbPointsLvl=0;

  public void animation() {//positionne l'animation du bonus de points
    super.animation(powerupPoints,"BPoints");
  }
}
```

bonusTp.pde :

```
class BonusTP extends Bonus {
  public boolean bonusTP=false,trigTP=false,canTP=false;;

  public void animation() {//positionne l'animation du bonus de téléportation
    super.animation(powerupP,"BTP");
  }
}
```

Hero.pde :

```
class Hero {
  protected int heroPos1, heroPos2, heroPos3, heroPos4, heroPos5, heroPos6, heroPos7, heroPos8,
heroPos9, heroPos10, heroPos11, heroPos12, heroPos13, heroPos14, heroPos15, heroPos16,
heroPos17, heroPos18;
  protected float x, y, previousX=0, previousY=0, actualX, actualY;
  public float vitX=8, vitY=2, mvt=0;
  public int nbMort=0;
```

```
  private int i=0, j=0, nbMontee=0, nbDescente=0, timeDeath, timeAnimP1, timeAnimP2,
nbDashAnim=0;

  public boolean saut=false, jumping=false, droite=false, gauche=false, reset=true, firstChute=true,
VIE=true, VIEHasBeenFalse=false, TPActivationP1=false, TPActivationP2=false, DashActivation=false;

  public char lastMove='A';



  public void resetDessin() {//fonction permettant de réactualiser la position du hero


    for (int i=0; i<128*38; i++) {

      if (hitboxLvl[levelNumber][i].endsWith("hero")) {

        heroPos1=i;//en bas à droite

        heroPos2=i-1;

        heroPos3=i-2;

        heroPos4=i-3;

        heroPos5=i-4;

        heroPos6=i-5;

        heroPos7=i-128;

        heroPos8=i-129;

        heroPos9=i-130;

        heroPos10=i-131;

        heroPos11=i-132;

        heroPos12=i-133;

        heroPos13=i-256;

        heroPos14=i-257;

        heroPos15=i-258;

        heroPos16=i-259;

        heroPos17=i-260;

        heroPos18=i-261;//en haut à gauche

        x=(heroPos18%128)*8;

        y=(heroPos18/128)*16;

        previousX=x;
```

```
      previousY=y;
    }
  }
 }

 public void dessin() {//fonction dessinant le personnage
  if (reset==true) {
    resetDessin();
    reset=false;
  }
  //rect((heroPos1%128)*8, (heroPos1/128)*16, 8, 16);
  animation();//appelle la fonction où toutes les animations possible du personnage sont détaillées en fonction de son bonus, son orientation,s'il bouge ou non, s'il saute etc..
 }

 public void animation() {
  if (VIEHasBeenFalse==false) {//si jamais le hero est en vie / vient de revenir à la vie
    if (keyPressed==true && (key==DROITE ||lastMove==DROITE) && jumping==false) {//si on appuie sur la touche droite et que l'on ne saute pas
      if (bonusDoubleJump.doubleJump==true || bonusDoubleJump.doubleJumpOn==true) {//si le bonus est le double saut
        image(heroGreenR, x, y);//met l'image du hero quand il a le bonus de double saut
      } else if (bonusNoClip.noClip==true) {//bonus non existant
        image(heroBlackR, x, y);
      } else if (bonusTP.bonusTP==true && bonusTP.trigTP==false && TPActivationP1==false && TPActivationP2==false) {//si le bonus est le tp et qu'il n'est pas activé
        image(heroPurpleR, x, y);//met l'image du hero quand il a le bonus du tp
      } else if (TPActivationP1==true ||TPActivationP2==true) {// ne met pas d'image quand on active le tp
      } else if (bonusDash.dash==true && bonusDash.trigDash==false && DashActivation==false) {//si le bonus est le dash et qu'il n'est pas activé
        image(heroRedR, x, y);//met l'image du hero quand il a le bonus de dash
      } else if (DashActivation==true) {//ne met pas d'image quand on active le dash
```

```
      } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false) {//si le bonus est l'inversion de gravité et qu'il n'est pas activé

        image(heroVeridianR, x, y);//met l'image du hero quand il a le bonus d'inversion de gravité non activé

      } else if (bonusGravitySwap.trigGSwap==true) {//si le bonus d'inversion de gravité est activé

        image(heroGSwapR, x, y);//met l'image du hero quand il a le bonus d'inversion de gravité activé

      } else {

        image(heroBlueR, x, y);//sinon met l'image du hero quand il se déplace sur la droite et n'a rien de particulier

      }

    } else if (keyPressed==true && (key==GAUCHE || lastMove==GAUCHE) && jumping==false) {//si on appuie sur la touche de gauche et que l'on ne saute pas même principe qu'au dessus mais à gauche

      if (bonusDoubleJump.doubleJump==true || bonusDoubleJump.doubleJumpOn==true) {

        image(heroGreenL, x, y);

      } else if (bonusNoClip.noClip==true) {

        image(heroBlackL, x, y);

      } else if (bonusTP.bonusTP==true && TPActivationP1==false && TPActivationP2==false) {

        image(heroPurpleL, x, y);

      } else if (TPActivationP1==true ||TPActivationP2==true) {

      } else if (bonusDash.dash==true && bonusDash.trigDash==false && DashActivation==false) {

        image(heroRedL, x, y);

      } else if (DashActivation==true) {

      } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false) {

        image(heroVeridianL, x, y);

      } else if (bonusGravitySwap.trigGSwap==true) {

        image(heroGSwapL, x, y);

      } else {

        image(heroBlueL, x, y);

      }

    } else if (lastMove==GAUCHE && jumping==false) {//si le dernier mouvement est à gauche, et que l'on appuie pas à gauche, toutes les prochaines opérations sont des images idle sur la gauche

      if (bonusDoubleJump.doubleJump==true || bonusDoubleJump.doubleJumpOn==true) {//si on a le bonus de double saut
```

```
        image(heroIdleGreenL, x, y);//affiche l'image du hero quand il a le bonus de double saut
      } else if (bonusNoClip.noClip==true) {//bonus non fonctionnel
        image(heroIdleBlackL, x, y);
      } else if (bonusTP.bonusTP==true && TPActivationP1==false && TPActivationP2==false) {//si on a le bonus de tp sans qu'il soit activé
        image(heroIdlePurpleL, x, y);//met l'image du hero quand il a le bonus de tp
      } else if (TPActivationP1==true || TPActivationP2==true) {//si on a le bonus de tp activé ne fait rien
      } else if (bonusDash.dash==true && bonusDash.trigDash==false && DashActivation==false) {//si on a le bonus de dash sans qu'il soit activé
        image(heroIdleRedL, x, y);//met l'image du hero quand il a le bonus de dash
      } else if (DashActivation==true) {//si on a le bonus de dash activé, ne fait rien
      } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false) {//si on a le bonus d'inversion de gravité qui n'est pas activé
        image(heroIdleVeridianL, x, y);//met l'image du hero quand il le bonus d'inversion de gravité non activé
      } else if (bonusGravitySwap.trigGSwap==true) {//si le bonus d'inversion de gravité est activé
        image(heroIdleGSwapL, x, y);//met l'image du hero quand il a le bonus d'inversion de gravité activé
      } else {
        image(heroIdleBlueL, x, y);// sinon met l'image du hero quand il n'a aucun bonus
      }
    } else if ( jumping==false || (jumping==false && lastMove=='A')) { // de même quand on a pas encore bougé/ quand on bouge sur la droite
      if (bonusDoubleJump.doubleJump==true || bonusDoubleJump.doubleJumpOn==true) {
        image(heroIdleGreenR, x, y);
      } else if (bonusNoClip.noClip==true) {
        image(heroIdleBlackR, x, y);
      } else if (bonusTP.bonusTP==true && TPActivationP1==false && TPActivationP2==false) {
        image(heroIdlePurpleR, x, y);
      } else if (TPActivationP1==true ||TPActivationP2==true) {
      } else if (bonusDash.dash==true && bonusDash.trigDash==false && DashActivation==false) {
        image(heroIdleRedR, x, y);
      } else if (DashActivation==true) {
```

```
      } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false) {
        image(heroIdleVeridianR, x, y);
      } else if (bonusGravitySwap.trigGSwap==true) {
        image(heroIdleGSwapR, x, y);
      } else {
        image(heroIdleBlueR, x, y);
      }
    }
    if (key==DROITE ||lastMove==DROITE) {//si on appuie sur la droite, ou que l'on a appuyé sur la droite
      if (TPActivationP1==true) {//et que le tp a été activé
        image(heroTPR1, actualX, actualY);//met l'animation de la première partie du tp
      } else if (TPActivationP2==true) {
        image(heroTPR2, x, y);//met l'animation de la seconde partie du tp
      }
      if (DashActivation==true) {//si le dash a été activé
        image(heroDashR, x, y);//met l'animation du dash
      }
    } else if (key==GAUCHE ||lastMove==GAUCHE) {//si on appuie sur la gauche, ou que l'on a appuyé sur la gauche
      if (TPActivationP1==true) {//et que le tp a été activé
        image(heroTPL1, actualX, actualY);//met l'animation de la première partie du tp
      } else if (TPActivationP2==true) {
        image(heroTPL2, x, y);//met l'animation de la seconde partie du tp
      } else if (DashActivation==true) {//si le dash a été activé
        image(heroDashL, x, y);//met l'animation du dash
      }
    }
    if ((jumping==true && lastMove==DROITE)||( lastMove=='A' && jumping==true)) {//si on saute vers la droite, ou que l'on saute sans avoir fait de mouvement(gauche ou droite) depuis le début du niveau
      if (bonusDoubleJump.doubleJump==true || bonusDoubleJump.doubleJumpOn==true) {
```

```
    image(heroJumpGreenR, x, y);
  } else if (bonusNoClip.noClip==true) {
    image(heroJumpBlackR, x, y);
  } else if (bonusTP.bonusTP==true && TPActivationP1==false && TPActivationP2==false) {
    image(heroJumpPurpleR, x, y);
  } else if (TPActivationP1==true ||TPActivationP2==true) {
  } else if (bonusDash.dash==true && DashActivation==false) {
    image(heroJumpRedR, x, y);
  } else if (DashActivation==true) {
  } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false) {
    image(heroJumpVeridianR, x, y);
  } else if (bonusGravitySwap.trigGSwap==true) {
    image(heroJumpGSwapR, x, y);
  } else {
    image(heroJumpBlueR, x, y);
  }
}
if (jumping==true && lastMove==GAUCHE) {//si on saute vers la gauche
  if (bonusDoubleJump.doubleJump==true || bonusDoubleJump.doubleJumpOn==true) {
    image(heroJumpGreenL, x, y);
  } else if (bonusNoClip.noClip==true) {
    image(heroJumpBlackL, x, y);
  } else if (bonusTP.bonusTP==true && TPActivationP1==false && TPActivationP2==false) {
    image(heroJumpPurpleL, x, y);
  } else if (TPActivationP1==true ||TPActivationP2==true) {
  } else if (bonusDash.dash==true && DashActivation==false) {
    image(heroJumpRedL, x, y);
  } else if (DashActivation==true) {
  } else if (bonusGravitySwap.GSwap==true && bonusGravitySwap.trigGSwap==false) {
    image(heroJumpVeridianL, x, y);
  } else if (bonusGravitySwap.trigGSwap==true) {
```

```
        image(heroJumpGSwapL, x, y);
      } else {
        image(heroJumpBlueL, x, y);
      }
    }
  } else {//sinon on est mort met l'image de mort en fonction du bonus
    if (bonusDoubleJump.doubleJump==true ||bonusDoubleJump.doubleJumpOn==true) {
      image(heroDeadGreen, actualX, actualY);
    } else if (bonusNoClip.noClip==true) {
      image(heroDeadBlack, actualX, actualY);
    } else if (bonusTP.bonusTP==true) {
      image(heroDeadPurple, actualX, actualY);
    } else if (bonusDash.dash==true) {
      image(heroDeadRed, actualX, actualY);
    } else if (bonusGravitySwap.GSwap==true) {
      image(heroDeadVeridian, actualX, actualY);
    } else {
      image(heroDeadBlue, actualX, actualY);
    }
  }
}



 public void mvtGauche() {//déplace d'une case vers la gauche dans la matrice du niveau
   try {//si il n'y a pas de mur à gauche
    if (((!hitboxLvl[levelNumber][hero.heroPos6-1].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos12-1].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos18-1].startsWith("wall"))||(bonusNoClip.trigNoClip==true
&&(hitboxLvl[levelNumber][hero.heroPos18-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos17-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos16-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos15-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos14-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos13-1].startsWith("wallNoClip") ||
```

```
hitboxLvl[levelNumber][hero.heroPos12-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos11-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos10-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos9-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos8-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos7-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos6-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos5-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos4-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos3-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos2-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos1-1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos7-1].startsWith("wallNoClip"))))) {

      if (hero.gauche==true && x>previousX-8) {//mvt d'une case

        x-=vitX;

        hero.heroHitboxGauche(1);

        hero.lastMove=GAUCHE;

        hero.previousX=hero.x;

      }

    }

  }

   catch(ArrayIndexOutOfBoundsException e) {//récupère les exceptions pour ne pas avoir de
problème d'essayer de lire une valeur d'un tableau dont la clef est plus grande que la length du
tableau

     ArrayIndexException();

   }

 }


  public void mvtDroite() {//déplace d'une case vers la droite dans la matrice du niveau

   try {//si il n'y a pas de mur à droite

     if ( ((!hitboxLvl[levelNumber][hero.heroPos1+1].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos7+1].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos13+1].startsWith("wall"))||(bonusNoClip.trigNoClip==true
&&(hitboxLvl[levelNumber][hero.heroPos18+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos17+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos16+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos15+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos14+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos13+1].startsWith("wallNoClip") ||
```

```
hitboxLvl[levelNumber][hero.heroPos12+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos11+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos10+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos9+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos8+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos7+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos6+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos5+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos4+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos3+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos2+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos1+1].startsWith("wallNoClip") ||
hitboxLvl[levelNumber][hero.heroPos7+1].startsWith("wallNoClip"))))) {

      if (hero.droite==true && x<previousX+8) {//mvt d'une case

        x+=vitX;

        hero.heroHitboxDroite(1);

        hero.lastMove=DROITE;

        hero.previousX=hero.x;

      }

    }

  }

  catch(ArrayIndexOutOfBoundsException e) {//récupère les exceptions pour ne pas avoir de problème d'essayer de lire une valeur d'un tableau dont la clef est plus grande que la length du tableau

    ArrayIndexException();

  }

 }


 public void heroHitbox() {//place la hitbox du hero sur la matrice

  if (hitboxLvl[levelNumber][heroPos1]=="end"+str(levelNumber)) {//si la case où l'on va placer une case du hero est une case end

    hitboxLvl[levelNumber][hero.heroPos1]="end"+str(levelNumber);//on laisse la case end (notamment pour les cases end qui mène de la droite à la gauche du niveau)

  } else if (hitboxLvl[levelNumber][heroPos1].startsWith("wallNoClip")) {//bonus non fonctionnel

    hitboxLvl[levelNumber][heroPos1]="wallNoCliphero";

  } else {//sinon place une case hero à cette place

    hitboxLvl[levelNumber][hero.heroPos1]="hero";
```

```
  }
  if (hitboxLvl[levelNumber][hero.heroPos2]=="end"+str(levelNumber)) {// de même pour les 17 autres cases
    hitboxLvl[levelNumber][hero.heroPos2]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos2].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos5]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos2]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos3]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos3]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos3].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos5]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos3]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos4]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos4]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos4].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos5]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos4]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos5]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos5]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos5].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos5]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos5]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos6]=="end"+str(levelNumber)) {
```

```
  hitboxLvl[levelNumber][hero.heroPos6]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos6].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos5]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos6]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos7]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos7]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos7].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos7]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos7]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos8]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos8]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos8].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos8]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos8]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos9]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos9]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos9].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos9]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos9]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos10]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos10]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos10].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos10]="wallNoCliphero";
```

```
} else {
  hitboxLvl[levelNumber][hero.heroPos10]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos11]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos11]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos11].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos11]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos11]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos12]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos12]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos12].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos12]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos12]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos13]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos13]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos13].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos13]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos13]="hero";
}
if (hitboxLvl[levelNumber][hero.heroPos14]=="end"+str(levelNumber)) {
  hitboxLvl[levelNumber][hero.heroPos14]="end"+str(levelNumber);
} else if (hitboxLvl[levelNumber][heroPos14].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][heroPos14]="wallNoCliphero";
} else {
  hitboxLvl[levelNumber][hero.heroPos14]="hero";
}
```

```
  if (hitboxLvl[levelNumber][hero.heroPos15]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos15]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos15].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos15]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos15]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos16]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos16]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos16].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos16]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos16]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos17]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos17]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos17].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos17]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos17]="hero";
  }
  if (hitboxLvl[levelNumber][hero.heroPos18]=="end"+str(levelNumber)) {
    hitboxLvl[levelNumber][hero.heroPos18]="end"+str(levelNumber);
  } else if (hitboxLvl[levelNumber][heroPos18].startsWith("wallNoClip")) {
    hitboxLvl[levelNumber][heroPos18]="wallNoCliphero";
  } else {
    hitboxLvl[levelNumber][hero.heroPos18]="hero";
  }
 }
 //------------------------début-fonction-du-bonusNoClip-non-fonctionnel-----------------------------------------
-------------------------------------------------//
```

```
 public void wallNoClipheroDelete() {

  for (int i=1; i<128*38-1; i++) {

   if (hitboxLvl[levelNumber][i].equals("wallNoCliphero") &&
(!hitboxLvl[levelNumber][i+1].equals("wallNoCliphero") ||
!hitboxLvl[levelNumber][i+1].equals("hero") || !hitboxLvl[levelNumber][i-
1].equals("wallNoCliphero") || !hitboxLvl[levelNumber][i-1].equals("hero"))) {

     hitboxLvl[levelNumber][i]="wallNoClip";

    }

  }

 }

 //-----------------------------fin-fonction-du-bonusNoClip-non-fonctionnel----------------------------------------
-----------------------------------------------//

 public void heroHitboxGauche(int nbDeplacement) {//déplace la hitbox vers la gauche de
nbDeplacement tant qu'il n'y a pas de mur à nbDeplacement case par rapport au hero

  try {

   while ((bonusNoClip.trigNoClip==false &&(hitboxLvl[levelNumber][hero.heroPos18-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos17-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos16-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos15-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos14-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos13-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos12-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos11-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos10-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos9-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos8-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos7-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos6-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos5-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos4-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos3-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos2-
nbDeplacement].startsWith("wall") || hitboxLvl[levelNumber][heroPos1-
nbDeplacement].startsWith("wall")))||(bonusNoClip.trigNoClip==true &&
((!hitboxLvl[levelNumber][heroPos18-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos18-nbDeplacement].startsWith("wall") ) ||
(!hitboxLvl[levelNumber][heroPos17-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos17-nbDeplacement].startsWith("wall") ) ||
(!hitboxLvl[levelNumber][heroPos16-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos16-nbDeplacement].startsWith("wall") ) ||
(!hitboxLvl[levelNumber][heroPos15-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos15-nbDeplacement].startsWith("wall") )||
```

```
(!hitboxLvl[levelNumber][heroPos14-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos14-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos13-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos13-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos12-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos12-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos11-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos11-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos10-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos10-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos9-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos9-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos8-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos8-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos7-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos7-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos6-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos6-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos5-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos5-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos4-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos4-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos3-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos3-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos2-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos2-nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos1-nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos1-nbDeplacement].startsWith("wall") )))) {


  nbDeplacement--;//réduit de 1 nbDeplacement tant qu'il n'y a pas de mur
}
hero.heroPos1-=nbDeplacement;
hero.heroPos2-=nbDeplacement;
hero.heroPos3-=nbDeplacement;
hero.heroPos4-=nbDeplacement;
hero.heroPos5-=nbDeplacement;
hero.heroPos6-=nbDeplacement;
hero.heroPos7-=nbDeplacement;
hero.heroPos8-=nbDeplacement;
hero.heroPos9-=nbDeplacement;
hero.heroPos10-=nbDeplacement;
```

```
    hero.heroPos11-=nbDeplacement;
    hero.heroPos12-=nbDeplacement;
    hero.heroPos13-=nbDeplacement;
    hero.heroPos14-=nbDeplacement;
    hero.heroPos15-=nbDeplacement;
    hero.heroPos16-=nbDeplacement;
    hero.heroPos17-=nbDeplacement;
    hero.heroPos18-=nbDeplacement;
    contactSpike();
    contactBonusDoubleJump();
    contactBonusDash();
    contactBonusTP();
    contactBonusGSwap();
    contactBonusPoints();
    contactBonusNoClip();
    //----------------------------------------------Code non necessaire car le bonusNoClip ne fonctionne pas-----
----------------------------------------------------//
    if (!hitboxLvl[levelNumber][heroPos1+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos1+nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos2+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos2+nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos3+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos3+nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos4+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos4+nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos5+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos5+nbDeplacement]="empty";
```

```
}
if (!hitboxLvl[levelNumber][heroPos6+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos6+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos7+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos7+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos8+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos8+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos9+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos9+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos10+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos10+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos11+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos11+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos12+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos12+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos13+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos13+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos14+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos14+nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos15+nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos15+nbDeplacement]="empty";
}
```

```
    if (!hitboxLvl[levelNumber][heroPos16+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos16+nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos17+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos17+nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos18+nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos18+nbDeplacement]="empty";
    }
    //------------------------------fin code non nécessaire car le bonusNoClip ne fonctionne pas----------
------------------------//
    heroHitbox();
   }
   catch(ArrayIndexOutOfBoundsException e) {// récupère toutes les exceptions pour éviter les
problèmes de clef plus grande que la length de la matrice/clef négative
    ArrayIndexException();
   }
 }
  public void heroHitboxDroite(int nbDeplacement) {//déplace la hitbox vers la droite (miroir de
heroHitboxGauche)
   try {
    while ((bonusNoClip.trigNoClip==false
&&(hitboxLvl[levelNumber][hero.heroPos18+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos17+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos16+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos15+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos14+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos13+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos12+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos11+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos10+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos9+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos8+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos7+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos6+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos5+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos4+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos3+nbDeplacement].startsWith("wall") ||
```

```
hitboxLvl[levelNumber][heroPos2+nbDeplacement].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos1+nbDeplacement].startsWith("wall")))||(bonusNoClip.trigNoClip==t
rue && ((!hitboxLvl[levelNumber][heroPos18+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos18+nbDeplacement].startsWith("wall") ) ||
(!hitboxLvl[levelNumber][heroPos17+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos17+nbDeplacement].startsWith("wall") ) ||
(!hitboxLvl[levelNumber][heroPos16+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos16+nbDeplacement].startsWith("wall") ) ||
(!hitboxLvl[levelNumber][heroPos15+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos15+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos14+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos14+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos13+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos13+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos12+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos12+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos11+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos11+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos10+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos10+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos9+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos9+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos8+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos8+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos7+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos7+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos6+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos6+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos5+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos5+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos4+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos4+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos3+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos3+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos2+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos2+nbDeplacement].startsWith("wall") )||
(!hitboxLvl[levelNumber][heroPos1+nbDeplacement].startsWith("wallNoClip") &&
hitboxLvl[levelNumber][heroPos1+nbDeplacement].startsWith("wall") )))) {

    nbDeplacement--;

  }

  hero.heroPos1+=nbDeplacement;

  hero.heroPos2+=nbDeplacement;

  hero.heroPos3+=nbDeplacement;

  hero.heroPos4+=nbDeplacement;
```

```
    hero.heroPos5+=nbDeplacement;
    hero.heroPos6+=nbDeplacement;
    hero.heroPos7+=nbDeplacement;
    hero.heroPos8+=nbDeplacement;
    hero.heroPos9+=nbDeplacement;
    hero.heroPos10+=nbDeplacement;
    hero.heroPos11+=nbDeplacement;
    hero.heroPos12+=nbDeplacement;
    hero.heroPos13+=nbDeplacement;
    hero.heroPos14+=nbDeplacement;
    hero.heroPos15+=nbDeplacement;
    hero.heroPos16+=nbDeplacement;
    hero.heroPos17+=nbDeplacement;
    hero.heroPos18+=nbDeplacement;
    contactSpike();
    contactBonusDoubleJump();
    contactBonusDash();
    contactBonusTP();
    contactBonusGSwap();
    contactBonusPoints();
    contactBonusNoClip();
    //-----------------------------------------code non fonctionnel car bonusNoClip ne fonctionne pas--------
-----------------------------------------//
    if (!hitboxLvl[levelNumber][heroPos1-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos1-nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos2-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos2-nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos3-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos3-nbDeplacement]="empty";
```

```
}
if (!hitboxLvl[levelNumber][heroPos4-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos4-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos5-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos5-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos6-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos6-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos7-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos7-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos8-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos8-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos9-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos9-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos10-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos10-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos11-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos11-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos12-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos12-nbDeplacement]="empty";
}
if (!hitboxLvl[levelNumber][heroPos13-nbDeplacement].startsWith("wallNoClip")) {
  hitboxLvl[levelNumber][hero.heroPos13-nbDeplacement]="empty";
}
```

```
    if (!hitboxLvl[levelNumber][heroPos14-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos14-nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos15-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos15-nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos16-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos16-nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos17-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos17-nbDeplacement]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos18-nbDeplacement].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos18-nbDeplacement]="empty";
    }
    //---------------------------------------------code non fonctionnel car bonusNoClip ne fonctionne pas----------
-------------------------------------------//
    heroHitbox();
   }
   catch(ArrayIndexOutOfBoundsException e) {
    ArrayIndexException();
   }
  }
  public void heroHitboxHaut() {//déplace la hitbox vers le haut
   try {
    hero.heroPos1-=128;
    hero.heroPos2-=128;
    hero.heroPos3-=128;
    hero.heroPos4-=128;
    hero.heroPos5-=128;
    hero.heroPos6-=128;
```

```
    hero.heroPos7-=128;
    hero.heroPos8-=128;
    hero.heroPos9-=128;
    hero.heroPos10-=128;
    hero.heroPos11-=128;
    hero.heroPos12-=128;
    hero.heroPos13-=128;
    hero.heroPos14-=128;
    hero.heroPos15-=128;
    hero.heroPos16-=128;
    hero.heroPos17-=128;
    hero.heroPos18-=128;
    contactBonusDoubleJump();
    contactBonusTP();
    contactBonusDash();
    contactBonusNoClip();
    contactBonusGSwap();
    contactBonusPoints();
    contactSpike();
    //-----------------------------------------code non fonctionnel car bonusNoClip ne fonctionne pas--------
-----------------------------------------//
    if (!hitboxLvl[levelNumber][heroPos1+128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos1+128]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos2+128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos2+128]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos3+128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos3+128]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos4+128].startsWith("wallNoClip")) {
```

```
      hitboxLvl[levelNumber][hero.heroPos4+128]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos5+128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos5+128]="empty";
    }
    if (!hitboxLvl[levelNumber][heroPos6+128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][hero.heroPos6+128]="empty";
    }
    //----------------------------------------------code non fonctionnel car bonusNoClip ne fonctionne pas----------
-----------------------------------------------//
    heroHitbox();
   }
   catch(ArrayIndexOutOfBoundsException e) {
    ArrayIndexException();
   }
  }
  public void heroHitboxChute() {//déplace la hitbox vers le bas(miroir de heroHitboxHaut)
   try {
    hero.heroPos1+=128;
    hero.heroPos2+=128;
    hero.heroPos3+=128;
    hero.heroPos4+=128;
    hero.heroPos5+=128;
    hero.heroPos6+=128;
    hero.heroPos7+=128;
    hero.heroPos8+=128;
    hero.heroPos9+=128;
    hero.heroPos10+=128;
    hero.heroPos11+=128;
    hero.heroPos12+=128;
    hero.heroPos13+=128;
```

```
    hero.heroPos14+=128;
    hero.heroPos15+=128;
    hero.heroPos16+=128;
    hero.heroPos17+=128;
    hero.heroPos18+=128;
    contactBonusDoubleJump();
    contactBonusTP();
    contactBonusDash();
    contactBonusNoClip();
    contactBonusGSwap();
    contactBonusPoints();
    contactSpike();
    //---------------------------------------------code non fonctionnel car bonusNoClip ne fonctionne pas---------
-------------------------------------------//
    if (!hitboxLvl[levelNumber][hero.heroPos18-128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][heroPos18-128]="empty";
    }
    if (!hitboxLvl[levelNumber][hero.heroPos17-128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][heroPos17-128]="empty";
    }
    if (!hitboxLvl[levelNumber][hero.heroPos16-128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][heroPos16-128]="empty";
    }
    if (!hitboxLvl[levelNumber][hero.heroPos15-128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][heroPos15-128]="empty";
    }
    if (!hitboxLvl[levelNumber][hero.heroPos14-128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][heroPos14-128]="empty";
    }
    if (!hitboxLvl[levelNumber][hero.heroPos13-128].startsWith("wallNoClip")) {
      hitboxLvl[levelNumber][heroPos13-128]="empty";
```

```
    }
    //-----------------------------------------code non fonctionnel car bonusNoClip ne fonctionne pas----------
-----------------------------------------//
    heroHitbox();
   }
   catch(ArrayIndexOutOfBoundsException e) {
    ArrayIndexException();
   }
 }


 public void distanceSol() {//calcul la distance entre les pieds du hero et le sol
   while (bonusGravitySwap.trigGSwap==false && nbMontee==0 &&
heroPos1+nbDescente*128<128*37 && bonusGravitySwap.trigGSwap==false &&
!hitboxLvl[levelNumber][hero.heroPos1+(nbDescente*128)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos2+(nbDescente*128)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos3+(nbDescente*128)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos4+(nbDescente*128)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos5+(nbDescente*128)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos6+(nbDescente*128)].startsWith("wall")) {
    nbDescente++;
   }
   if (hero.saut==true && nbMontee==0 && nbDescente==0 && bonusGravitySwap.trigGSwap==true)
{
    nbDescente=6;
   }
   if ((nbDescente==0 && bonusGravitySwap.trigGSwap==false)) {// si on touche le sol, on peut
sauter
    jumping=false;
   }
   if (bonusGravitySwap.trigGSwap==false) {
    jumping=true;
   }
 }
 public void confirmPosition() {//fonction pour que le hero ne 'saute/tombe' pas pour rien
```

```
   try {
     if (hitboxLvl[levelNumber][heroPos1+128].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos2+128].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos3+128].startsWith("wall")
||hitboxLvl[levelNumber][heroPos4+128].startsWith("wall") ||
hitboxLvl[levelNumber][heroPos5+128].startsWith("wall")
||hitboxLvl[levelNumber][heroPos6+128].startsWith("wall")) {
       nbMontee=0;
       nbDescente=0;
     }
   }
   catch(ArrayIndexOutOfBoundsException e) {
     ArrayIndexException();
   }
 }
  public void mvtHaut() {//calcul de la distance entre les pieds du hero et le sol, pendant un saut
   if (hero.saut==true && nbMontee==0 && nbDescente==0 &&
bonusGravitySwap.trigGSwap==false) {
     nbMontee=6;
   }
   try {
     while (bonusGravitySwap.trigGSwap==true && nbDescente==0 &&
!hitboxLvl[levelNumber][hero.heroPos18-(128*nbMontee)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos17-(128*nbMontee)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos16-(128*nbMontee)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos15-(128*nbMontee)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos14-(128*nbMontee)].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos13-(128*nbMontee)].startsWith("wall") && saut==false &&
bonusDoubleJump.trigDoubleJump==false) {
       nbMontee++;
     }
   }
   catch(ArrayIndexOutOfBoundsException e) {
     ArrayIndexException();
   }
```

```
   if (bonusDoubleJump.trigDoubleJump==true) {//si on saute en l'air relance le fonctionnement du saut à 0
     nbMontee=6;
     nbDescente=0;
   }
   if (nbMontee==0 && nbDescente==0 && bonusGravitySwap.trigGSwap==true) {//Si on ne monte pas ni ne descend, pas on touche le sol donc on ne saute pas
     jumping=false;
   }
   if (bonusGravitySwap.trigGSwap==true) {
     jumping=true;
   }
  }

  public void descente() {//fonction permettant de créer une gravité dans le jeu
   try {
     if (mvt<6 && i<6 &&  bonusDash.trigDash==false && bonusTP.trigTP==false && TPActivationP1==false && TPActivationP2==false && !hitboxLvl[levelNumber][hero.heroPos1+128].startsWith("wall") && !hitboxLvl[levelNumber][hero.heroPos2+128].startsWith("wall") && !hitboxLvl[levelNumber][hero.heroPos3+128].startsWith("wall") && !hitboxLvl[levelNumber][hero.heroPos4+128].startsWith("wall") && !hitboxLvl[levelNumber][hero.heroPos5+128].startsWith("wall") && !hitboxLvl[levelNumber][hero.heroPos6+128].startsWith("wall")) {
       if (firstChute==true) {
         vitY=0;
         firstChute=false;
         mvt=nbDescente;
       }
       y+=vitY;
       vitY+=0.31;
       i++;
     }
   }
```

```
   catch(ArrayIndexOutOfBoundsException e) {
    ArrayIndexException();
   }
   try {
    if ( mvt>=6 && i<6 && bonusDash.trigDash==false && bonusTP.trigTP==false &&
TPActivationP1==false && TPActivationP2==false &&
!hitboxLvl[levelNumber][hero.heroPos1+128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos2+128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos3+128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos4+128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos5+128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos6+128].startsWith("wall")) {
      y+=8;
      i+=3;
      firstChute=false;
    } else if (i>=6 && nbMontee==0 && nbDescente>0) {
      if (bonusGravitySwap.trigGSwap==true) {
        hero.saut=false;
      }
      nbDescente--;
      i=0;
      heroHitboxChute();
    }
   }
   catch(ArrayIndexOutOfBoundsException e) {
    ArrayIndexException();
   }
   if (bonusGravitySwap.trigGSwap==false && heroPos1+nbDescente*128<128*37 &&
(hitboxLvl[levelNumber][hero.heroPos1+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos2+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos3+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos4+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos5+128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos6+128].startsWith("wall"))) {
    jumping=false;
    resetDessin();
```

```
    nbDescente=0;
    firstChute=true;
    if (bonusDoubleJump.doubleJumpOn==true) {
      bonusDoubleJump.doubleJump=true;
    }
   } else {
    if (bonusGravitySwap.trigGSwap==false) {
      jumping=true;
    }
   }
 }
 public void montee() {//créer un saut impacté par la gravité
   try {
    if (nbMontee>0 && j<6 && bonusDash.trigDash==false && bonusTP.trigTP==false &&
TPActivationP1==false && TPActivationP2==false  && !hitboxLvl[levelNumber][hero.heroPos18-
128].startsWith("wall") && !hitboxLvl[levelNumber][hero.heroPos17-128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos16-128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos15-128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos14-128].startsWith("wall") &&
!hitboxLvl[levelNumber][hero.heroPos13-128].startsWith("wall")) {
      if (nbMontee==6) {
        if (j==0) {
          vitY=7;
        }
        y-=vitY;
        vitY-=0.5;
      } else if ((nbMontee>1 && nbMontee<6)) {
        if (j==0 && nbMontee==5) {
          vitY=3.1;
        }
        y-=vitY;
        vitY-=0.1;
      }
```

```
      j++;
     }
    }
    catch(ArrayIndexOutOfBoundsException e) {
     ArrayIndexException();
    }
    if (j==6 && nbDescente==0) {
     if (bonusGravitySwap.trigGSwap==false) {
      hero.saut=false;
     }
     if (bonusDoubleJump.trigDoubleJump==true) {
      bonusDoubleJump.trigDoubleJump=false;
      bonusDoubleJump.doubleJump=false;
     }
     if (nbMontee>1) {
      hero.heroHitboxHaut();
     }
     nbMontee--;
     j=0;
    }
    try {
     if ( heroPos18-nbDescente*128>0 &&  (hitboxLvl[levelNumber][hero.heroPos18-
128].startsWith("wall") || hitboxLvl[levelNumber][hero.heroPos17-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos16-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos15-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos14-128].startsWith("wall") ||
hitboxLvl[levelNumber][hero.heroPos13-128].startsWith("wall"))) {
      resetDessin();
      jumping=false;
      nbMontee=0;
     }
    }
    catch(ArrayIndexOutOfBoundsException e) {
```

```
    ArrayIndexException();
   }
  }



  public void dash() {//fonction du fonctionnement et du timing de l'animation du dash
   if (bonusDash.dash==true && bonusDash.canDash==false && hero.jumping==false) {//si on touche le sol quand on a le bonus dash, on peut dash à nouveau
    bonusDash.canDash=true;
   }
   if (bonusDash.trigDash==true) {//si on active le dash
    if (nbDashAnim<12) {
     hero.contactEnd();
     bonusDash.canDash=false;
     if (lastMove==GAUCHE) {//dash à gauche
      heroHitboxGauche(2);
     }
     if (lastMove==DROITE || lastMove=='A' && VIEHasBeenFalse==false) {//dash à droite
      heroHitboxDroite(2);
      lastMove=DROITE;
     }
     DashActivation=true;
     timeAnimP1=millis();
     nbDashAnim++;
     resetDessin();
    } else {
     nbDashAnim=0;
     bonusDash.trigDash=false;
    }
   }
   if (bonusDash.dash==true && DashActivation==true && (millis()-timeAnimP1)%450>=300) {//temps de l'animation 300ms
```

```
    DashActivation=false;
    heroDashL.stop();//reset du gif à gauche
    heroDashR.stop();//reset du gif à droite
    heroDashL.play();//lancement du gif à gauche
    heroDashR.play();//lancement du gif à droite
   }
   if (bonusDash.dash==false && DashActivation==true) {
    DashActivation=false;
    heroDashL.stop();
    heroDashR.stop();
   }
  }

  public void TP() {//fonction du tp
   if (bonusTP.bonusTP==true && bonusTP.canTP==false && hero.jumping==false) {//si on touche le sol avec le bonus tp permet de tp à nouveau
    bonusTP.canTP=true;
   }
   if (bonusTP.trigTP==true) {//si on active le tp
    actualX=x;
    actualY=y;
    hero.contactEnd();
    bonusTP.canTP=false;
    if (lastMove==GAUCHE) {//tp à gauche
     heroHitboxGauche(16);
    }
    if (lastMove==DROITE) {//tp à droite
     heroHitboxDroite(16);
    }
    TPActivationP1=true;
    bonusTP.trigTP=false;
```

```
    heroTPR1.play();
    heroTPL1.play();
    timeAnimP1=millis();
  }
  if (TPActivationP1==true && (millis()-timeAnimP1)%400>=300) {//temps de la première animation du tp 300ms
    TPActivationP1=false;
    TPActivationP2=true;
    heroTPR1.stop();
    heroTPL1.stop();
    heroTPR2.play();
    heroTPL2.play();
    timeAnimP2=millis();
  }
  if (bonusTP.bonusTP==true && TPActivationP2==true && (millis()-timeAnimP2)%400>=300) {//temps de la seconde animation du tp 300ms
    TPActivationP2=false;
    heroTPR2.stop();
    heroTPL2.stop();
  }
  if (bonusTP.bonusTP==false && (TPActivationP1==true || TPActivationP2==true)) {
    TPActivationP2=false;
    TPActivationP1=false;
    heroTPR1.stop();
    heroTPR2.stop();
    heroTPL1.stop();
    heroTPL2.stop();
  }

  if (bonusTP.trigTP==true || (TPActivationP1==true || TPActivationP2==true)) {//replace l'image du hero à sa position
    resetDessin();
```

```
  }
}
//----------------------------fonction non fonctionnelle bonusNoClip---------------------------------//
public void NoClip() {
  if (bonusNoClip.trigNoClip==true) {
  }
}
//-------------------------------------------------------------------------------------------------//

public void recommencer() {//fonction qui reset toutes les variables quand on est mort
  if (VIE==false) {
    VIE=true;
    nbMort++;
    VIEHasBeenFalse=true;
    lastMove='A';
    jumping=false;
    gauche=false;
    droite=false;
    actualX=x;
    actualY=y;
    bonusGravitySwap.trigGSwap=false;
    bonusGravitySwap.timeActivationGSwap=false;
    bonusPoints.nbPoints-=bonusPoints.nbPointsLvl;
    bonusPoints.nbPointsLvl=0;
    animation();
    String actualLevel="lvl"+str(levelNumber+1);
    hitboxLvl[levelNumber]=loadStrings("data/levels/lvlsHitbox/"+actualLevel+".txt");
    //println("Vous êtes mort.");
    resetDessin();
    death.trigger();
    timeDeath=millis();
```

```
    println("aaaa");
   }
   if (VIEHasBeenFalse==true && (millis()-timeDeath)%500>=400) {//400ms le temps de l'animation
de la mort
    VIEHasBeenFalse=false;
    nbMontee=0;
    nbDescente=0;
    bonusDoubleJump.trigDoubleJump=false;
    hero.saut=false;
    bonusDoubleJump.doubleJump=false;
    bonusDoubleJump.doubleJumpOn=false;
    bonusDash.dash=false;
    bonusTP.bonusTP=false;
    bonusTP.trigTP=false;
    bonusNoClip.noClip=false;
    bonusGravitySwap.GSwap=false;
   }
  }

  public void contactSpike() {//fonction si on touche les piques, nous tue
   if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos17].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos16].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos15].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos14].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos13].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos12].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos11].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos10].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos9].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos8].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos7].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos6].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos5].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos4].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos3].startsWith("spike") ||
```

```
hitboxLvl[levelNumber][heroPos2].startsWith("spike") ||
hitboxLvl[levelNumber][heroPos1].startsWith("spike")) {

    VIE=false;

  }

 }

  public void contactBonusDoubleJump() {//fonction si on touche un bonusDoubleJump, désactive les autres bonus et active celui-ci

   if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos17].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos16].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos15].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos14].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos13].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos12].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos11].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos10].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos9].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos8].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos7].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos6].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos5].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos4].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos3].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos2].startsWith("BJump") ||
hitboxLvl[levelNumber][heroPos1].startsWith("BJump")) {

    bonusDoubleJump.doubleJump=true;

    bonusDoubleJump.doubleJumpOn=true;

    bonusDash.dash=false;

    bonusDash.canDash=false;

    bonusTP.bonusTP=false;

    bonusTP.canTP=false;

    bonusNoClip.noClip=false;

    bonusDoubleJump.x=2000;

    bonusDoubleJump.y=2000;

    bonusGravitySwap.GSwap=false;

    bonusGravitySwap.trigGSwap=false;

    bonusGravitySwap.timeActivationGSwap=false;

    powerup.trigger();//son du bonus
```

```
  }
 }
  public void contactBonusDash() {//fonction si on touche un bonusDash, désactive les autres bonus et active celui-ci
   if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos17].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos16].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos15].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos14].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos13].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos12].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos11].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos10].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos9].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos8].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos7].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos6].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos5].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos4].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos3].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos2].startsWith("BDash") ||
hitboxLvl[levelNumber][heroPos1].startsWith("BDash")) {
    bonusDash.dash=true;
    // sound.loadBonusSFX();
    bonusDash.x=2000;
    bonusDash.y=2000;
    bonusDash.canDash=true;
    bonusTP.bonusTP=false;
    bonusTP.canTP=false;
    bonusDoubleJump.doubleJump=false;
    bonusDoubleJump.doubleJumpOn=false;
    bonusNoClip.noClip=false;
    bonusGravitySwap.GSwap=false;
    bonusGravitySwap.trigGSwap=false;
    bonusGravitySwap.timeActivationGSwap=false;
    powerup.trigger();//son du bonus
  }
```

```
 }

  public void contactBonusTP() {//fonction si on touche un bonusTP, désactive les autres bonus et
active celui-ci

    if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos17].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos16].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos15].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos14].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos13].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos12].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos11].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos10].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos9].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos8].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos7].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos6].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos5].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos4].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos3].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos2].startsWith("BTP") ||
hitboxLvl[levelNumber][heroPos1].startsWith("BTP")) {

      bonusTP.bonusTP=true;

      //sound.loadBonusSFX();

      bonusTP.x=2000;

      bonusTP.y=2000;

      bonusTP.canTP=true;

      bonusDash.dash=false;

      bonusDash.canDash=false;

      bonusDoubleJump.doubleJump=false;

      bonusDoubleJump.doubleJumpOn=false;

      bonusNoClip.noClip=false;

      bonusGravitySwap.GSwap=false;

      bonusGravitySwap.trigGSwap=false;

      bonusGravitySwap.timeActivationGSwap=false;

      powerup.trigger();//son

    }

  }
```

```
 //----------------------------fonction pour le bonusNoClip non fonctionnel-----------------------------------------
-----//

 public void contactBonusNoClip() {

   if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos17].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos16].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos15].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos14].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos13].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos12].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos11].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos10].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos9].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos8].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos7].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos6].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos5].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos4].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos3].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos2].startsWith("BnoClip") ||
hitboxLvl[levelNumber][heroPos1].startsWith("BnoClip")) {

    bonusNoClip.x=2000;

    bonusNoClip.y=2000;

    bonusNoClip.noClip=true;

    bonusTP.bonusTP=false;

    bonusTP.canTP=false;

    bonusDash.dash=false;

    bonusDash.canDash=false;

    bonusDoubleJump.doubleJump=false;

    bonusDoubleJump.doubleJumpOn=false;

    bonusGravitySwap.GSwap=false;

    bonusGravitySwap.trigGSwap=false;

    bonusGravitySwap.timeActivationGSwap=false;

    powerup.trigger();

  }

 }

 //-------------------------------------------------------------------------------------------------------------------//
```

```
  public void contactBonusGSwap() {//fonction si on touche un bonus d'inversion de gravité, désactive les autres bonus et active celui-ci

    if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos17].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos16].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos15].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos14].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos13].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos12].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos11].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos10].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos9].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos8].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos7].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos6].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos5].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos4].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos3].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos2].startsWith("BGSwap") ||
hitboxLvl[levelNumber][heroPos1].startsWith("BGSwap")) {

      bonusGravitySwap.GSwap=true;

      // sound.loadBonusSFX();

      bonusGravitySwap.x=2000;

      bonusGravitySwap.y=2000;

      bonusNoClip.noClip=false;

      bonusTP.bonusTP=false;

      bonusTP.canTP=false;

      bonusDash.dash=false;

      bonusDash.canDash=false;

      bonusDoubleJump.doubleJump=false;

      bonusDoubleJump.doubleJumpOn=false;

      powerup.trigger();

    }

  }

  public void contactBonusPoints() {//fonction qui rajoute 1 au score quand on touche un bonus points
```

```
   if (hitboxLvl[levelNumber][hero.heroPos18].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos17].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos16].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos15].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos14].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos13].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos12].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos11].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos10].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos9].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos8].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos7].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos6].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos5].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos4].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos3].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos2].startsWith("BPoints") ||
hitboxLvl[levelNumber][heroPos1].startsWith("BPoints")) {

     bonusPoints.x=2000;

     bonusPoints.y=2000;

     bonusPoints.nbPoints++;

     bonusPoints.nbPointsLvl++;

     powerup.trigger();

   }

 }


  public void contactEnd() {//fonction travaillant avec les cases end de la matrices

    int actualLevelNumber=levelNumber;

    try {// si le "end" est celui qui permet d'aller dans le niveau courant et que l'on y passe par la
gauche nous amène à droite

     if (lastMove==GAUCHE && (hitboxLvl[levelNumber][hero.heroPos18-
1].equals("end"+levelNumber) || hitboxLvl[levelNumber][heroPos17-1].equals("end"+levelNumber)
|| hitboxLvl[levelNumber][heroPos16-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos15-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos14-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos13-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos12-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos7-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos6-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos5-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos4-1].equals("end"+levelNumber) ||
```

```
hitboxLvl[levelNumber][heroPos3-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos2-1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos1-1].equals("end"+levelNumber))) {

      heroHitboxDroite(120);

    }

  }

  catch(ArrayIndexOutOfBoundsException e) {

    heroHitboxDroite(120);

  }

  try {//si le "end" est celui qui permet d'aller dans le niveau courant et que l'on y passe par la droite, nous y amène par la gauche

    if (lastMove==DROITE &&
(hitboxLvl[levelNumber][hero.heroPos18+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos17+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos16+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos15+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos14+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos13+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos12+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos7+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos6+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos5+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos4+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos3+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos2+1].equals("end"+levelNumber) ||
hitboxLvl[levelNumber][heroPos1+1].equals("end"+levelNumber))) {

      heroHitboxGauche(120);

    }

  }

  catch(ArrayIndexOutOfBoundsException e) {

    heroHitboxGauche(120);

  }

  if (lastMove==GAUCHE) {//si l'on vient par la gauche et qu'il y a une porte end nous envoie au niveau de la case "end" en question+1, car on finit le niveau de numéro "end"

    if (hitboxLvl[levelNumber][hero.heroPos18-1].startsWith("end")) {

      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos18-1].substring(3));

    }
```

```
if (hitboxLvl[levelNumber][hero.heroPos17-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos17-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos16-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos16-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos15-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos15-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos14-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos14-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos13-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos13-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos12-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos12-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos7-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos7-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos6-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos6-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos5-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos5-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos4-1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos4-1].substring(3));
}
if (hitboxLvl[levelNumber][hero.heroPos3-1].startsWith("end")) {
```

```
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos3-1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos2-1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos2-1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos1-1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos1-1].substring(3));
   }
  } else  if (lastMove==DROITE) {//de même par la droite
   if (hitboxLvl[levelNumber][hero.heroPos18+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos18+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos17+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos17+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos16+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos16+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos15+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos15+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos14+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos14+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos13+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos13+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos12+1].startsWith("end")) {
    levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos12+1].substring(3));
   }
   if (hitboxLvl[levelNumber][hero.heroPos7+1].startsWith("end")) {
```

```
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos7+1].substring(3));
 }
 if (hitboxLvl[levelNumber][hero.heroPos6+1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos6+1].substring(3));
 }
 if (hitboxLvl[levelNumber][hero.heroPos5+1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos5+1].substring(3));
 }
 if (hitboxLvl[levelNumber][hero.heroPos4+1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos4+1].substring(3));
 }
 if (hitboxLvl[levelNumber][hero.heroPos3+1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos3+1].substring(3));
 }
 if (hitboxLvl[levelNumber][hero.heroPos2+1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos2+1].substring(3));
 }
 if (hitboxLvl[levelNumber][hero.heroPos1+1].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos1+1].substring(3));
 }
}
try {
 if (hitboxLvl[levelNumber][heroPos1+128].startsWith("end")) {//de même par le haut ou le bas
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos1+128].substring(3));
 } else if (hitboxLvl[levelNumber][heroPos2+128].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos2+128].substring(3));
 } else if (hitboxLvl[levelNumber][heroPos3+128].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos3+128].substring(3));
 } else if (hitboxLvl[levelNumber][heroPos4+128].startsWith("end")) {
  levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos4+128].substring(3));
 } else if (hitboxLvl[levelNumber][heroPos5+128].startsWith("end")) {
```

```
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos5+128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos6+128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos6+128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos13-128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos13-128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos14-128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos14-128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos15-128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos15-128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos16-128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos16-128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos17-128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos17-128].substring(3));
    } else if (hitboxLvl[levelNumber][heroPos18-128].startsWith("end")) {
      levelNumber=Integer.parseInt(hitboxLvl[levelNumber][hero.heroPos18-128].substring(3));
    }
   }
   catch(ArrayIndexOutOfBoundsException e) {
    ArrayIndexException();
   }

   if (actualLevelNumber!=levelNumber) {//si il y a une différence avec le numéro du niveau courant le numéro du niveau global
    nextLevel();//lance cette fonction qui amène vers le niveau global
   }
  }
  void nextLevel() {//fonction qui désactive tous les mouvements et bonus et qui nous envoie au niveau suivant
    sound.firstMusic=false;
    reset=true;
    lastMove='A';
    jumping=false;
```

```
    saut=false;
    firstChute=false;
    nbMontee=0;
    nbDescente=0;
    bonusPoints.nbPointsLvl=0;
    bonusDoubleJump.doubleJump=false;
    bonusDoubleJump.doubleJumpOn=false;
    bonusDash.dash=false;
    bonusTP.bonusTP=false;
    bonusTP.trigTP=false;
    bonusNoClip.noClip=false;
    bonusGravitySwap.trigGSwap=false;
    bonusGravitySwap.GSwap=false;
    String actualLevel="lvl"+str(levelNumber+1);
    hitboxLvl[levelNumber]=loadStrings("data/levels/lvlsHitbox/"+actualLevel+".txt");
    sound.musicChange();
  }
  void ArrayIndexException() {//fonction qui nous tue si on sort de la matrice et que la case en question n'était pas une case end(car dans un catch qui avant testait si la case était un end)
    VIE=false;
    nbDescente=0;
    nbMontee=0;
    lastMove='A';
    jumping=false;
    gauche=false;
    droite=false;
    nbMontee=0;
    nbDescente=0;
    bonusDoubleJump.doubleJump=false;
    bonusDoubleJump.doubleJumpOn=false;
    bonusDash.dash=false;
```

```
    bonusTP.bonusTP=false;
    bonusTP.trigTP=false;
    bonusNoClip.noClip=false;
    bonusGravitySwap.trigGSwap=false;
    bonusGravitySwap.GSwap=false;
    bonusGravitySwap.timeActivationGSwap=false;
  }
}
```

Interface.pde :

```
class Interface {
  boolean firstScreen=true, ecranTitre=false, setUsername=false, load=false, credit=false, leaderboard=false, firstLoad=true, firstLeaderboard=true, save=false, pause=false, importing=false, exporting=false;
  ;
  int line=1, column=1;
  int actualPage=1, usernameNumber=0, usersLength=0;
  int testedValue=1;
  int creditTime=0;
  int loadedHour=0, loadedMinute=0, loadedSecond=0, loadedMillis=0, firstLoadedMillis=0, stepMillisIntegration=0, is1000=0;
  float nbPage;
  String pseudo="", charChosen;
  String[]playerBase;
  String[] player;



  public void ecranTitre() {//fonction qui affiche l'ecranTitre et qui permet d'avoir une interface dynamique si l'on bouge le joystick
    image(menuPNG, 0, 0);
    textFont(font);
    textSize(42);
    strokeWeight(4);
```

```
stroke(255, 102, 0);
fill(255, 102, 0, 50);
switch(column) {
case 1://play
  rect(200, 200, 600, 60, 20);
  fill(255, 102, 0);
  text("PLAY", 415, 250);
  stroke(100, 100, 100);
  fill(50, 50, 50, 150);
  rect(200, 280, 600, 60, 20);
  rect(200, 360, 600, 60, 20);
  rect(200, 440, 600, 60, 20);
  rect(200, 520, 600, 60, 20);
  fill(255);
  text("LOAD", 415, 330);
  text("LEADERBOARD", 262, 410);
  text("CREDIT", 372, 490);
  text("EXIT", 415, 570);
  break;
case 2://load
  rect(200, 280, 600, 60, 20);
  fill(255, 102, 0);
  text("LOAD", 415, 330);
  stroke(100, 100, 100);
  fill(50, 50, 50, 150);
  rect(200, 200, 600, 60, 20);
  rect(200, 360, 600, 60, 20);
  rect(200, 440, 600, 60, 20);
  rect(200, 520, 600, 60, 20);
  fill(255);
  text("PLAY", 415, 250);
```

```
  text("LEADERBOARD", 262, 410);
  text("CREDIT", 372, 490);
  text("EXIT", 415, 570);
  break;
 case 3://leaderboard
  rect(200, 360, 600, 60, 20);
  fill(255, 102, 0);
  text("LEADERBOARD", 262, 410);
  stroke(100, 100, 100);
  fill(50, 50, 50, 150);
  rect(200, 200, 600, 60, 20);
  rect(200, 280, 600, 60, 20);
  rect(200, 440, 600, 60, 20);
  rect(200, 520, 600, 60, 20);
  fill(255);
  text("PLAY", 415, 250);
  text("LOAD", 415, 330);
  text("CREDIT", 372, 490);
  text("EXIT", 415, 570);
  break;
 case 4://credit
  rect(200, 440, 600, 60, 20);
  fill(255, 102, 0);
  text("CREDIT", 372, 490);
  stroke(100, 100, 100);
  fill(50, 50, 50, 150);
  rect(200, 200, 600, 60, 20);
  rect(200, 280, 600, 60, 20);
  rect(200, 360, 600, 60, 20);
  rect(200, 520, 600, 60, 20);
  fill(255);
```

```
    text("PLAY", 415, 250);
    text("LOAD", 415, 330);
    text("LEADERBOARD", 262, 410);
    text("EXIT", 415, 570);
    break;
  case 5://exit
    rect(200, 520, 600, 60, 20);
    fill(255, 102, 0);
    text("EXIT", 415, 570);
    stroke(100, 100, 100);
    fill(50, 50, 50, 150);
    rect(200, 200, 600, 60, 20);
    rect(200, 280, 600, 60, 20);
    rect(200, 360, 600, 60, 20);
    rect(200, 440, 600, 60, 20);
    fill(255);
    text("PLAY", 415, 250);
    text("LOAD", 415, 330);
    text("LEADERBOARD", 262, 410);
    text("CREDIT", 372, 490);
  }
 }

 void visualKeyboard() {//fonction qui fait apparaitre un clavier visuel pour saisir un pseudo, la touche où l'on se situe au moment actuel est 'surligné'
  fill(100, 100, 100, 100);
  stroke(100, 100, 100);
  for (int i=0; i<10; i++) {
    rect(25+i*100, 150, 75, 75, 5);
  }
  for (int i=0; i<10; i++) {
```

```
  rect(25+i*100, 250, 75, 75, 5);
}
for (int i=0; i<6; i++) {
  rect(225+i*100, 350, 75, 75, 5);
}
for (int i=0; i<2; i++) {
  rect(275+i*250, 450, 225, 75, 5);
}
textFont(font);
fill(255);
text('A', 25+29, 150+45);
text('Z', 125+29, 150+45);
text('E', 225+29, 150+45);
text('R', 325+29, 150+45);
text('T', 425+29, 150+45);
text('Y', 525+29, 150+45);
text('U', 625+29, 150+45);
text('I', 725+29, 150+45);
text('O', 825+29, 150+45);
text('P', 925+29, 150+45);
text('Q', 25+29, 250+45);
text('S', 125+29, 250+45);
text('D', 225+29, 250+45);
text('F', 325+29, 250+45);
text('G', 425+29, 250+45);
text('H', 525+29, 250+45);
text('J', 625+29, 250+45);
text('K', 725+29, 250+45);
text('L', 825+29, 250+45);
text('M', 925+29, 250+45);
text('W', 225+29, 350+45);
```

```
text('X', 325+29, 350+45);
text('C', 425+29, 350+45);
text('V', 525+29, 350+45);
text('B', 625+29, 350+45);
text('N', 725+29, 350+45);
text("DELETE", 275+50, 450+45);
text("FINISH", 275+250+50, 450+45);
for (int i=0; i<9; i++) {
  line(63+i*100, 120, 138+i*100, 120);
}
stroke(0, 255, 0);
fill(0, 255, 0, 100);
switch (column) {
case 1:
  switch(line) {
  case 1:
    rect(25, 150, 75, 75, 5);
    text('A', 54, 195);
    charChosen="A";
    break;
  case 2:
    rect(125, 150, 75, 75, 5);
    text('Z', 154, 195);
    charChosen="Z";
    break;
  case 3:
    rect(225, 150, 75, 75, 5);
    text('E', 254, 195);
    charChosen="E";
    break;
  case 4:
```

```
  rect(325, 150, 75, 75, 5);
  text('R', 354, 195);
  charChosen="R";
  break;
 case 5:
  rect(425, 150, 75, 75, 5);
  text('T', 454, 195);
  charChosen="T";
  break;
 case 6:
  rect(525, 150, 75, 75, 5);
  text('Y', 554, 195);
  charChosen="Y";
  break;
 case 7:
  rect(625, 150, 75, 75, 5);
  text('U', 654, 195);
  charChosen="U";
  break;
 case 8:
  rect(725, 150, 75, 75, 5);
  text('I', 754, 195);
  charChosen="I";
  break;
 case 9:
  rect(825, 150, 75, 75, 5);
  text('O', 854, 195);
  charChosen="O";
  break;
 case 10:
  rect(925, 150, 75, 75, 5);
```

```
    text('P', 954, 195);
    charChosen="P";
    break;
   }
   break;
  case 2:
   switch(line) {
   case 1:
    rect(25, 250, 75, 75, 5);
    text('Q', 54, 295);
    charChosen="Q";
    break;
   case 2:
    rect(125, 250, 75, 75, 5);
    text('S', 154, 295);
    charChosen="S";
    break;
   case 3:
    rect(225, 250, 75, 75, 5);
    text('D', 254, 295);
    charChosen="D";
    break;
   case 4:
    rect(325, 250, 75, 75, 5);
    text('F', 354, 295);
    charChosen="F";
    break;
   case 5:
    rect(425, 250, 75, 75, 5);
    text('G', 454, 295);
    charChosen="G";
```

```
    break;
  case 6:
    rect(525, 250, 75, 75, 5);
    text('H', 554, 295);
    charChosen="H";
    break;
  case 7:
    rect(625, 250, 75, 75, 5);
    text('J', 654, 295);
    charChosen="J";
    break;
  case 8:
    rect(725, 250, 75, 75, 5);
    text('K', 754, 295);
    charChosen="K";
    break;
  case 9:
    rect(825, 250, 75, 75, 5);
    text('L', 854, 295);
    charChosen="L";
    break;
  case 10:
    rect(925, 250, 75, 75, 5);
    text('M', 954, 295);
    charChosen="M";
    break;
  }
  break;
 case 3:
  switch(line) {
  case 1:
```

```
  rect(225, 350, 75, 75, 5);
  text('W', 254, 395);
  charChosen="W";
  break;
 case 2:
  rect(325, 350, 75, 75, 5);
  text('X', 354, 395);
  charChosen="X";
  break;
 case 3:
  rect(425, 350, 75, 75, 5);
  text('C', 454, 395);
  charChosen="C";
  break;
 case 4:
  rect(525, 350, 75, 75, 5);
  text('V', 554, 395);
  charChosen="V";
  break;
 case 5:
  rect(625, 350, 75, 75, 5);
  text('B', 654, 395);
  charChosen="B";
  break;
 case 6:
  rect(725, 350, 75, 75, 5);
  text('N', 754, 395);
  charChosen="N";
  break;
 }
 break;
```

```
  case 4:
    switch(line) {
    case 1:
      rect(275, 450, 225, 75, 5);
      text("DELETE", 325, 495);
      charChosen="DELETE";
      break;
    case 2:
      rect(525, 450, 225, 75, 5);
      text("FINISH", 575, 495);
      charChosen="FINISH";
      break;
    }
    break;
  }
  char[] arrayPseudo=pseudo.toCharArray();
  textSize(44);
  fill(0, 255, 0, 180);
  for (int i=0; i<arrayPseudo.length; i++) {
    text(arrayPseudo[i], 85+i*100, 115);//affiche les lettres du pseudo en haut
  }
}

void save() {//fonction pour sauvegarder avec la même interface que le load
  load();
  if (21*nbPage==usersLength) {
    nbPage++;
  }
}
```

```
 void load() {//fonction qui fait apparaître une page permettant de choisir quelle partie on souhaite charger
   String[] users=new String[0];
   for (int i=0; i<playerBase.length-1; i++) {
     String[] pseudo=split(playerBase[i+1], '|');
     users=append(users, pseudo[0]);
   }
   if (firstLoad==true) {//quand on charge pour la première fois la fonction load, on regarde combien de page vont être nécessaire
     nbPage=(float)users.length/21;//on a 21 parties par pages
     if (nbPage-(int)nbPage!=0) {
       nbPage=(int)nbPage+1;
     }
     usersLength=users.length;
     firstLoad=false;
   }
   for (int i=0; i<(21*nbPage)-usersLength; i++) {//finit de remplir une page avec des cases "empty"
     users=append(users, "    \'empty\'");
   }
   fill(100, 100, 100, 100);
   stroke(100, 100, 100);
   strokeWeight(4);
   for (int i=0; i<7  && i<users.length/3; i++) {//place 21 rectangles dans lesquels les pseudos seront mis
     rect(80, 20+(60*i), 250, 50, 20);
     rect(380, 20+(60*i), 250, 50, 20);
     rect(680, 20+(60*i), 250, 50, 20);
   }


   fill(255);
   textSize(13);
   for (int i=0; i<21 && i<users.length-1; i+=3) {//place le nom des utilisateurs dans ces rectangles
```

```
  text(users[i+(21*(actualPage-1))], 100, 40+(i*20));
  text(users[i+1+(21*(actualPage-1))], 400, 40+(i*20));
  text(users[i+2+(21*(actualPage-1))], 700, 40+(i*20));
}
fill(100, 100, 100, 100);
rect(90, 450, 400, 100, 20);//rectangle de la case retour
rect(515, 450, 400, 100, 20);//rectangle de la case import
rect(10, 15, 50, 415, 20);//rectangle de la case pour aller à la page de gauche
rect(964, 15, 50, 415, 20);//rectangle de la case pour aller à droite
stroke(255);
line(20, 220, 50, 160);//lignes fleches
line(20, 220, 50, 280);
line(1004, 220, 974, 160);
line(1004, 220, 974, 280);//lignes fleches
fill(255);
textSize(52);
text("RETOUR", 130, 520);
if (load==true) {
  text("IMPORT", 555, 520);
} else if (save==true) {
  text("EXPORT", 555, 520);
}
textSize(10);
stroke(80, 120, 200);
strokeWeight(2);
for (int i=0; (21*(actualPage-1)+3*i<usersLength-2) && i<7; i++) {
  player=split(playerBase[21*(actualPage-1)+3*i+1], "|");//récupère les valeurs du player courant
  line(95, 25+(i*60), 105, 25+(i*60));//petits 'symboles' pour montrer la zone texte du pseudo
  line(95, 25+(i*60), 95, 35+(i*60));
  line(220, 42+(i*60), 220, 32+(i*60));
  line(220, 42+(i*60), 210, 42+(i*60));
```

```
line(395, 25+(i*60), 405, 25+(i*60));
line(395, 25+(i*60), 395, 35+(i*60));
line(520, 42+(i*60), 520, 32+(i*60));
line(520, 42+(i*60), 510, 42+(i*60));
line(695, 25+(i*60), 705, 25+(i*60));
line(695, 25+(i*60), 695, 35+(i*60));
line(820, 42+(i*60), 820, 32+(i*60));
line(820, 42+(i*60), 810, 42+(i*60));//fin petits 'symboles' pour montrer la zone texte du pseudo
String[] time=split(player[3], ":");
image(timer, 100, 45+60*i, 24, 24);//placer l'image du timer
image(deathPNG, 220, 45+60*i, 24, 24);//placer l'image de la mort
image(coin, 225, 25+60*i, 16, 16);//placer l'image de la pièce
text(player[1], 245, 40+60*i);//placer le nombre de pièce du joueur

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Intege
r.parseInt(time[3]), 120, 60*(i+1));//place le temps du joueur
text(player[4], 245, 60*(i+1));//place le nombre de mort du joueur
player=split(playerBase[21*(actualPage-1)+(3*i+1)+1], "|");
time=split(player[3], ":");
image(timer, 400, 45+60*i, 24, 24);
image(deathPNG, 520, 45+60*i, 24, 24);
image(coin, 525, 25+60*i, 16, 16);

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Intege
r.parseInt(time[3]), 420, 60*(i+1));
text(player[4], 545, 60*(i+1));
text(player[1], 545, 40+60*i);
player=split(playerBase[21*(actualPage-1)+(3*i+2)+1], "|");
time=split(player[3], ":");
image(timer, 700, 45+60*i, 24, 24);
image(deathPNG, 820, 45+60*i, 24, 24);
image(coin, 825, 25+60*i, 16, 16);
```

```
text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Integer.parseInt(time[3]), 720, 60*(i+1));
    text(player[4], 845, 60*(i+1));
    text(player[1], 845, 40+60*i);
    if (3*(i+1)+2==usersLength-21*(actualPage-1) && actualPage==nbPage) {
     line(95, 25+((i+1)*60), 105, 25+((i+1)*60));
     line(95, 25+((i+1)*60), 95, 35+((i+1)*60));
     line(220, 42+((i+1)*60), 220, 32+((i+1)*60));
     line(220, 42+((i+1)*60), 210, 42+((i+1)*60));
     line(395, 25+((i+1)*60), 405, 25+((i+1)*60));
     line(395, 25+((i+1)*60), 395, 35+((i+1)*60));
     line(520, 42+((i+1)*60), 520, 32+((i+1)*60));
     line(520, 42+((i+1)*60), 510, 42+((i+1)*60));
     player=split(playerBase[playerBase.length-2], "|");
     time=split(player[3], ":");
     image(timer, 100, 45+60*(i+1), 24, 24);
     image(deathPNG, 220, 45+60*(i+1), 24, 24);
     image(coin, 225, 25+60*(i+1), 16, 16);
     text(player[1], 245, 40+60*(i+1));

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Integer.parseInt(time[3]), 120, 60*(i+2));
     text(player[4], 245, 60*(i+2));
     player=split(playerBase[playerBase.length-1], "|");
     time=split(player[3], ":");
     image(timer, 400, 45+60*(i+1), 24, 24);
     image(deathPNG, 520, 45+60*(1+i), 24, 24);
     image(coin, 525, 25+60*(i+1), 16, 16);

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Integer.parseInt(time[3]), 420, 60*(i+2));
     text(player[4], 545, 60*(i+2));
```

```
      text(player[1], 545, 40+60*(i+1));
    } else if (3*(i+1)+1==usersLength-21*(actualPage-1) && actualPage==nbPage) {
      line(95, 25+((i+1)*60), 105, 25+((i+1)*60));
      line(95, 25+((i+1)*60), 95, 35+((i+1)*60));
      line(220, 42+((i+1)*60), 220, 32+((i+1)*60));
      line(220, 42+((i+1)*60), 210, 42+((i+1)*60));
      player=split(playerBase[playerBase.length-1], "|");
      time=split(player[3], ":");
      image(timer, 100, 45+60*(i+1), 24, 24);
      image(deathPNG, 220, 45+60*(i+1), 24, 24);
      image(coin, 225, 25+60*(i+1), 16, 16);
      text(player[1], 245, 40+60*(i+1));

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Intege
r.parseInt(time[3]), 120, 60*(i+2));
      text(player[4], 245, 60*(i+2));
    }
   }
   if (usersLength-21*(actualPage-1)==2 && actualPage==nbPage) {
    line(95, 25, 105, 25);
    line(95, 25, 95, 35);
    line(220, 42, 220, 32);
    line(220, 42, 210, 42);
    line(395, 25, 405, 25);
    line(395, 25, 395, 35);
    line(520, 42, 520, 32);
    line(520, 42, 510, 42);
    player=split(playerBase[playerBase.length-2], "|");
    String[] time=split(player[3], ":");
    image(timer, 100, 45, 24, 24);
    image(deathPNG, 220, 45, 24, 24);
    image(coin, 225, 25, 16, 16);
```

```
    text(player[1], 245, 40);

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Integer.parseInt(time[3]), 120, 60);
    text(player[4], 245, 60);
    player=split(playerBase[playerBase.length-1], "|");
    time=split(player[3], ":");
    image(timer, 400, 45, 24, 24);
    image(deathPNG, 520, 45, 24, 24);
    image(coin, 525, 25, 16, 16);

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Integer.parseInt(time[3]), 420, 60);
    text(player[4], 545, 60);
    text(player[1], 545, 40);
  } else if (usersLength-21*(actualPage-1)==1 && actualPage==nbPage) {
    line(95, 25, 105, 25);
    line(95, 25, 95, 35);
    line(220, 42, 220, 32);
    line(220, 42, 210, 42);
    player=split(playerBase[playerBase.length-1], "|");
    String[] time=split(player[3], ":");
    image(timer, 100, 45, 24, 24);
    image(deathPNG, 220, 45, 24, 24);
    image(coin, 225, 25, 16, 16);
    text(player[1], 245, 40);

text(Integer.parseInt(time[0])+":"+Integer.parseInt(time[1])+":"+Integer.parseInt(time[2])+":"+Integer.parseInt(time[3]), 120, 60);
    text(player[4], 245, 60);
  }


  stroke(0, 255, 0);
```

```
textSize(13);
fill(0, 255, 0, 100);
switch(column) {
case 1:
 switch(line) {
 case 1://passage à la page de gauche
  rect(10, 15, 50, 415, 20);
  line(20, 220, 50, 160);
  line(20, 220, 50, 280);
  break;
 case 8:
  rect(90, 450, 400, 100, 20);//rectangle de la case retour
  textSize(52);
  text("RETOUR", 130, 520);
  break;
 }
 break;
case 2:
 switch(line) {
 case 1://case 1
  rect(80, 20, 250, 50, 20);
  text(users[0+(21*(actualPage-1))], 100, 40);
  usernameNumber=0+(21*(actualPage-1));
  break;
 case 2://case 4
  rect(80, 80, 250, 50, 20);
  text(users[3+(21*(actualPage-1))], 100, 100);
  usernameNumber=3+(21*(actualPage-1));
  break;
 case 3://case 7
  rect(80, 140, 250, 50, 20);
```

```
  text(users[6+(21*(actualPage-1))], 100, 160);
  usernameNumber=6+(21*(actualPage-1));
  break;
 case 4://case 10
  rect(80, 200, 250, 50, 20);
  text(users[9+(21*(actualPage-1))], 100, 220);
  usernameNumber=9+(21*(actualPage-1));
  break;
 case 5://case 13
  rect(80, 260, 250, 50, 20);
  text(users[12+(21*(actualPage-1))], 100, 280);
  usernameNumber=12+(21*(actualPage-1));
  break;
 case 6://case 16
  rect(80, 320, 250, 50, 20);
  text(users[15+(21*(actualPage-1))], 100, 340);
  usernameNumber=15+(21*(actualPage-1));
  break;
 case 7://case 19
  rect(80, 380, 250, 50, 20);
  text(users[18+(21*(actualPage-1))], 100, 400);
  usernameNumber=18+(21*(actualPage-1));
  break;
 case 8:
  rect(515, 450, 400, 100, 20);//rectangle de la case import
  textSize(52);
  if (load==true) {
   text("IMPORT", 555, 520);
  } else if (save==true) {
   text("EXPORT", 555, 520);
  }
```

```
    break;
   }
   break;
  case 3:
   switch(line) {
   case 1://case 2
    rect(380, 20, 250, 50, 20);
    text(users[1+(21*(actualPage-1))], 400, 40);
    usernameNumber=1+(21*(actualPage-1));
    break;
   case 2://case 5
    rect(380, 80, 250, 50, 20);
    text(users[4+(21*(actualPage-1))], 400, 100);
    usernameNumber=4+(21*(actualPage-1));
    break;
   case 3://case 8
    rect(380, 140, 250, 50, 20);
    text(users[7+(21*(actualPage-1))], 400, 160);
    usernameNumber=7+(21*(actualPage-1));
    break;
   case 4://case 11
    rect(380, 200, 250, 50, 20);
    text(users[10+(21*(actualPage-1))], 400, 220);
    usernameNumber=10+(21*(actualPage-1));
    break;
   case 5://case 14
    rect(380, 260, 250, 50, 20);
    text(users[13+(21*(actualPage-1))], 400, 280);
    usernameNumber=13+(21*(actualPage-1));
    break;
   case 6://case 17
```

```
    rect(380, 320, 250, 50, 20);
    text(users[16+(21*(actualPage-1))], 400, 340);
    usernameNumber=16+(21*(actualPage-1));
    break;
   case 7://case 20
    rect(380, 380, 250, 50, 20);
    text(users[19+(21*(actualPage-1))], 400, 400);
    usernameNumber=19+(21*(actualPage-1));
    break;
   }
   break;
  case 4:
   switch(line) {
   case 1://case 3
    rect(680, 20, 250, 50, 20);
    text(users[2+(21*(actualPage-1))], 700, 40);
    usernameNumber=2+(21*(actualPage-1));
    break;
   case 2://case 6
    rect(680, 80, 250, 50, 20);
    text(users[5+(21*(actualPage-1))], 700, 100);
    usernameNumber=5+(21*(actualPage-1));
    break;
   case 3://case 9
    rect(680, 140, 250, 50, 20);
    text(users[8+(21*(actualPage-1))], 700, 160);
    usernameNumber=8+(21*(actualPage-1));
    break;
   case 4://case 12
    rect(680, 200, 250, 50, 20);
    text(users[11+(21*(actualPage-1))], 700, 220);
```

```
      usernameNumber=11+(21*(actualPage-1));
      break;
    case 5://case 15
      rect(680, 260, 250, 50, 20);
      text(users[14+(21*(actualPage-1))], 700, 280);
      usernameNumber=14+(21*(actualPage-1));
      break;
    case 6://case 18
      rect(680, 320, 250, 50, 20);
      text(users[17+(21*(actualPage-1))], 700, 340);
      usernameNumber=17+(21*(actualPage-1));
      break;
    case 7://case 21
      rect(680, 380, 250, 50, 20);
      text(users[20+(21*(actualPage-1))], 700, 400);
      usernameNumber=20+(21*(actualPage-1));
      break;
    }
    break;
  case 5:
    rect(964, 15, 50, 415, 20);
    line(1004, 220, 974, 160);
    line(1004, 220, 974, 280);
    break;
  }
}
```

```
String[] leaderboard() { //fonction qui retourne un tableau où le leaderboard est rangé selon 4 possibilités, le score, le timer, le nombre de mort, le nombre de pièces, dans cet ordre de préférence
  int[] numberTested=new int[0];
```

```
  String[] player;
  String[] timer;
  int time=0;
  for (int i=0; i<playerBase.length-1; i++) {
   player=split(playerBase[i+1], "|");
   if (testedValue==3) {//si la valeur est 3, on coupe aussi selon ":" pour récupérer les heures,minutes,seconde;
     timer=split(player[3], ":");
     numberTested=append(numberTested, Integer.parseInt(timer[0])*3600000+Integer.parseInt(timer[1])*60000+Integer.parseInt(timer[2])*1000+Integer.parseInt(timer[3]));
   } else {
     numberTested=append(numberTested, Integer.parseInt(player[testedValue]));
   }
  }
  numberTested=sort(numberTested);//ordonne le tableau
  if (testedValue==1 || testedValue==5) {
   numberTested=reverse(numberTested);//reverse le tableau pour ces valeurs car il s'agit du score et des pièces, et l'on souhaite le plus grand nombre dans le leaderboard pour ceux-là
  }
  String[] lessDeathPlayer=new String[0];
  while (lessDeathPlayer.length<playerBase.length-1) {
   for (int i=0; i<playerBase.length-1; i++) {
     player=split(playerBase[i+1], "|");

     if (testedValue==3) {//recoupe notre playerBase selon ":" pour ensuite réordonner le tableau par rapport à la liste que l'on a obtenu tout à l'heure pour remettre les "|" et les ":" dedans
       timer=split(player[3], ":");

time=Integer.parseInt(timer[0])*3600000+Integer.parseInt(timer[1])*60000+Integer.parseInt(timer[2])*1000+Integer.parseInt(timer[3]);
```

```
        if (lessDeathPlayer.length<numberTested.length &&
numberTested[lessDeathPlayer.length]==time) {//si la valeur à la position i du tableau numberTested
est la même que time alors on ordonne le tableau avec celle-ci

          lessDeathPlayer=append(lessDeathPlayer,
player[0]+"|"+player[1]+"|"+player[2]+"|"+player[3]+"|"+player[4]+"|"+player[5]);

        }

       } else {

        if (lessDeathPlayer.length<numberTested.length &&
numberTested[lessDeathPlayer.length]==Integer.parseInt(player[testedValue])) {

          lessDeathPlayer=append(lessDeathPlayer,
player[0]+"|"+player[1]+"|"+player[2]+"|"+player[3]+"|"+player[4]+"|"+player[5]);

        }

       }

     }

   }

   return lessDeathPlayer;

 }


  void visualLeaderboard() {//affichage visuel du leaderboard

   String[] coinsPlayer=leaderboard();

   fill(255, 0, 0);

   if (firstLeaderboard==true) {//quand pour la première fois on consulte le leaderboard, on ordonne
les pages par joueurs par page

     nbPage=(float)coinsPlayer.length/8;

     if (nbPage-(int)nbPage!=0) {

       nbPage=(int)nbPage+1;

     }

   }

   strokeWeight(4);

   for (int i=1; i<5; i++) {//tracé des lignes du tableau

     line(200*i, 0, 200*i, 390);

   }

   textSize(16);
```

```
  line(30, 0, 30, 390);
  line(0, 55, 1024, 55);
  if (testedValue==1) {//selon la valeur de tested value on place l'élément correspondant à testedValue puis score puis timer puis death puis coin selon l'ordre(4 éléments max)
    image(coin, 275, 5);
    text("SCORE", 475, 30);
    image(timer, 675, 5);
    image(deathPNG, 875, 5);
  } else if (testedValue==3) {
    image(coin, 875, 5);
    text("SCORE", 475, 30);
    image(timer, 275, 5);
    image(deathPNG, 675, 5);
  } else if (testedValue==4) {
    image(coin, 875, 5);
    image(timer, 675, 5);
    text("SCORE", 475, 30);
    image(deathPNG, 275, 5);
  } else if (testedValue==5) {
    text("SCORE", 275, 30);
    image(timer, 475, 5);
    image(deathPNG, 675, 5);
    image(coin, 875, 5);
  }
  text("PSEUDO", 65, 30);
  textSize(12);
  strokeWeight(2);
  stroke(80, 120, 200);
  for (int i=0; i<8 && i+8*(actualPage-1)<coinsPlayer.length; i++) {
    player=split(coinsPlayer[(actualPage-1)*8+i], "|");
    text(((actualPage-1)*8+i+1), 5, 40+40*(i+1));//position de la personne dans le leaderboard
```

```
 line(35, 65+40*i, 45, 65+40*i);//début des délimitations du pseudo
 line(35, 65+40*i, 35, 75+40*i);
 line(190, 85+40*i, 190, 75+40*i);
 line(190, 85+40*i, 180, 85+40*i);//fin des délimitaions du pseudo
 text(player[0], 40, 40+40*(i+1));//pseudo du joueur à la position i+1
 if (testedValue==1) {//coins
  text(player[1], 210, 40+40*(i+1));
  text(player[5], 410, 40+40*(i+1));
  text(player[3], 610, 40+40*(i+1));
  text(player[4], 810, 40+40*(i+1));
 } else if (testedValue==3) {//time
  text(player[3], 210, 40+40*(i+1));
  text(player[5], 410, 40+40*(i+1));
  text(player[4], 610, 40+40*(i+1));
  text(player[1], 810, 40+40*(i+1));
 } else if (testedValue==4) {//death
  text(player[4], 210, 40+40*(i+1));
  text(player[5], 410, 40+40*(i+1));
  text(player[3], 610, 40+40*(i+1));
  text(player[1], 810, 40+40*(i+1));
 } else if (testedValue==5) {//score
  text(player[5], 210, 40+40*(i+1));
  text(player[3], 410, 40+40*(i+1));
  text(player[4], 610, 40+40*(i+1));
  text(player[1], 810, 40+40*(i+1));
 }
}
strokeWeight(4);
textSize(40);
if (line==1) {
 switch(column) {
```

```
case 1://page précédente
  fill(0, 255, 0, 150);
  stroke(0, 255, 0);
  rect(200, 400, 50, 50, 20);
  line(205, 425, 240, 405);
  line(205, 425, 240, 445);
  fill(100, 100, 100, 150);
  stroke(100, 100, 100);
  rect(315, 400, 50, 50, 20);
  image(coin, 320, 410, 40, 30);
  rect(415, 400, 50, 50, 20);
  image(deathPNG, 420, 410, 40, 30);
  rect(515, 400, 50, 50, 20);
  image(timer, 520, 410, 40, 30);
  rect(615, 400, 50, 50, 20);
  fill(255, 0, 0);
  text("S", 623, 443);
  fill(100, 100, 100, 150);
  rect(750, 400, 50, 50, 20);
  line(795, 425, 755, 405);
  line(795, 425, 755, 445);
  rect(150, 480, 700, 100, 20);
  fill(255);
  text("RETURN", 400, 545);
  break;
case 2://pieces
  fill(0, 255, 0, 150);
  stroke(0, 255, 0);
  rect(315, 400, 50, 50, 20);
  fill(100, 100, 100, 150);
  stroke(100, 100, 100);
```

```
  rect(200, 400, 50, 50, 20);
  line(205, 425, 240, 405);
  line(205, 425, 240, 445);
  image(coin, 320, 410, 40, 30);
  rect(415, 400, 50, 50, 20);
  image(deathPNG, 420, 410, 40, 30);
  rect(515, 400, 50, 50, 20);
  image(timer, 520, 410, 40, 30);
  rect(615, 400, 50, 50, 20);
  fill(255, 0, 0);
  text("S", 623, 443);
  fill(100, 100, 100, 150);
  rect(750, 400, 50, 50, 20);
  line(795, 425, 755, 405);
  line(795, 425, 755, 445);
  rect(150, 480, 700, 100, 20);
  fill(255);
  text("RETURN", 400, 545);
  break;
 case 3://death
  fill(0, 255, 0, 150);
  stroke(0, 255, 0);
  rect(415, 400, 50, 50, 20);
  fill(100, 100, 100, 150);
  stroke(100, 100, 100);
  rect(315, 400, 50, 50, 20);
  rect(200, 400, 50, 50, 20);
  line(205, 425, 240, 405);
  line(205, 425, 240, 445);
  image(coin, 320, 410, 40, 30);
  image(deathPNG, 420, 410, 40, 30);
```

```
  rect(515, 400, 50, 50, 20);
  image(timer, 520, 410, 40, 30);
  rect(615, 400, 50, 50, 20);
  fill(255, 0, 0);
  text("S", 623, 443);
  fill(100, 100, 100, 150);
  rect(750, 400, 50, 50, 20);
  line(795, 425, 755, 405);
  line(795, 425, 755, 445);
  rect(150, 480, 700, 100, 20);
  fill(255);
  text("RETURN", 400, 545);
  break;
 case 4://timer
  fill(0, 255, 0, 150);
  stroke(0, 255, 0);
  rect(515, 400, 50, 50, 20);
  fill(100, 100, 100, 150);
  stroke(100, 100, 100);
  rect(200, 400, 50, 50, 20);
  line(205, 425, 240, 405);
  line(205, 425, 240, 445);
  rect(315, 400, 50, 50, 20);
  image(coin, 320, 410, 40, 30);
  rect(415, 400, 50, 50, 20);
  image(deathPNG, 420, 410, 40, 30);
  image(timer, 520, 410, 40, 30);
  rect(615, 400, 50, 50, 20);
  fill(255, 0, 0);
  text("S", 623, 443);
  fill(100, 100, 100, 150);
```

```
  rect(750, 400, 50, 50, 20);
  line(795, 425, 755, 405);
  line(795, 425, 755, 445);
  rect(150, 480, 700, 100, 20);
  fill(255);
  text("RETURN", 400, 545);
  break;
 case 5://score
  fill(0, 255, 0, 150);
  stroke(0, 255, 0);
  rect(615, 400, 50, 50, 20);
  fill(255, 0, 0);
  text("S", 623, 443);
  fill(100, 100, 100, 150);
  stroke(100, 100, 100);
  rect(200, 400, 50, 50, 20);
  line(205, 425, 240, 405);
  line(205, 425, 240, 445);
  rect(315, 400, 50, 50, 20);
  image(coin, 320, 410, 40, 30);
  rect(415, 400, 50, 50, 20);
  image(deathPNG, 420, 410, 40, 30);
  rect(515, 400, 50, 50, 20);
  image(timer, 520, 410, 40, 30);
  rect(750, 400, 50, 50, 20);
  line(795, 425, 755, 405);
  line(795, 425, 755, 445);
  rect(150, 480, 700, 100, 20);
  fill(255);
  text("RETURN", 400, 545);
  break;
```

```
  case 6://page suivante
    fill(0, 255, 0, 150);
    stroke(0, 255, 0);
    rect(750, 400, 50, 50, 20);
    line(795, 425, 755, 405);
    line(795, 425, 755, 445);
    fill(100, 100, 100, 150);
    stroke(100, 100, 100);
    rect(200, 400, 50, 50, 20);
    line(205, 425, 240, 405);
    line(205, 425, 240, 445);
    rect(315, 400, 50, 50, 20);
    image(coin, 320, 410, 40, 30);
    rect(415, 400, 50, 50, 20);
    image(deathPNG, 420, 410, 40, 30);
    rect(515, 400, 50, 50, 20);
    image(timer, 520, 410, 40, 30);
    rect(615, 400, 50, 50, 20);
    fill(255, 0, 0);
    text("S", 623, 443);
    fill(100, 100, 100, 150);
    rect(150, 480, 700, 100, 20);
    fill(255);
    text("RETURN", 400, 545);
    break;
  }
}
if (line==2 && column==1) {
  fill(255);
  text("RETURN", 400, 545);
  fill(0, 255, 0, 150);
```

```
    stroke(0, 255, 0);
    rect(150, 480, 700, 100, 20);
    fill(100, 100, 100, 150);
    stroke(100, 100, 100);
    rect(200, 400, 50, 50, 20);
    line(205, 425, 240, 405);
    line(205, 425, 240, 445);
    rect(315, 400, 50, 50, 20);
    image(coin, 320, 410, 40, 30);
    rect(415, 400, 50, 50, 20);
    image(deathPNG, 420, 410, 40, 30);
    rect(515, 400, 50, 50, 20);
    image(timer, 520, 410, 40, 30);
    rect(615, 400, 50, 50, 20);
    fill(255, 0, 0);
    text("S", 623, 443);
    fill(100, 100, 100, 150);
    rect(750, 400, 50, 50, 20);
    line(795, 425, 755, 405);
    line(795, 425, 755, 445);
  }
}

void credits() {//affichage visuel de la page des crédits
  //gamedev
  //levelEditor dev
  //levelEditor design
  //sound design
  //character design
  //environment design
  //
```

```
  //level design for all levels
  //
  if (millis()-creditTime<=5000) {//change de page au bout de 5 secondes
    textSize(22);
    text("Game Development : Viale Stéphane", 50, 50);
    text("Level Editor development : Viale Stéphane", 50, 150);
    text("level Editor Design : Viale Stéphane", 50, 250);
    text("Sound Design : Coppé Vincent", 50, 350);
    text("Character  Design : Coppé Vincent", 50, 450);
    text("Environment Design : Coppé Vincent", 50, 550);
  } else if (millis()-creditTime<=10000) {//sur cette page et les suivantes il y a le nom de la personne qui a fait le niveau pendant 5secondes
    textSize(22);
    text("Levels Design", 350, 50);
    textSize(11);
    for (int i=1; i<=12; i++) {
      text("level "+i+" : Coppé Vincent", 50, 60+40*i);
      text("level "+(i+13)+" : Coppé Vincent", 350, 60+40*i);
    }
    text("level 13 : Viale Stéphane", 50, 60+40*13);
    text("level 26 : Coppé Vincent", 350, 60+40*13);
    text("level 27 : Viale Stéphane", 650, 100);
    for (int i=1; i<=5; i++) {
      text("level "+(i+27)+" : Coppé Vincent", 650, 60+40*(i+1));
    }
    for (int i=1; i<=7; i++) {
      text("level "+(i+32)+" : Viale Stéphane", 650, 60+40*(i+6));
    }
  } else if (millis()-creditTime<=15000) {
    textSize(22);
    text("Levels Design", 350, 50);
```

```
    textSize(11);
    for (int i=1; i<=13; i++) {
      text("level "+(i+39)+" : Viale Stéphane", 50, 60+40*i);
    }
    for (int i=1; i<=7; i++) {
      text("level "+(i+52)+" : Graulier Brice", 350, 60+40*i);
    }
  } else {//quitte les crédits et retourne à l'ecranTitre
    interfaces.credit=false;
    interfaces.ecranTitre=true;
  }
 }


 void pause() {
  image(menuEmpty, 0, 0);
  fill(#E4E823);
  textSize(22);
  image(timer, 350, 5);
  text(hour+":"+minute+":"+second+":"+(((millisPaused-initialTime)-
timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute
*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-
(minute*60000)-(hour*3600000)), 395, 35);//880
  image(deathPNG, 5, 5);
  text(hero.nbMort, 50, 35);
  image(coin, 5, 55);
  text(bonusPoints.nbPoints, 50, 88);
  stroke(100, 100, 200);
  fill(100, 100, 200, 50);
  rect(300, 50, 400, 500, 20);
  fill(255, 255, 255);
  textSize(52);
  text("PAUSE", 365, 120);
```

```
text("SAVE", 390, 220);
text("RESUME", 345, 340);
text("QUIT", 390, 460);

//fill(255, 102, 0, 50);
//stroke(255, 102, 0);
fill(100, 100, 100, 100);
rect(310, 150, 380, 100, 20);//save
rect(310, 270, 380, 100, 20);//resume
rect(310, 390, 380, 100, 20);//quit
fill(255, 102, 0, 50);
stroke(255, 102, 0);
switch(line) {
case 1://save
  rect(310, 150, 380, 100, 20);
  fill(255, 102, 0);
  text("SAVE", 390, 220);
  fill(100, 100, 100, 100);
  stroke(100, 100, 100);
  rect(310, 270, 380, 100, 20);//resume
  rect(310, 390, 380, 100, 20);//quit
  fill(255, 255, 255);
  text("RESUME", 345, 340);
  text("QUIT", 390, 460);
  break;
case 2://resume
  rect(310, 270, 380, 100, 20);//resume
  fill(255, 102, 0);
  text("RESUME", 345, 340);
  fill(100, 100, 100, 100);
  stroke(100, 100, 100);
```

```
    rect(310, 150, 380, 100, 20);
    rect(310, 390, 380, 100, 20);//quit
    fill(255, 255, 255);
    text("SAVE", 390, 220);
    text("QUIT", 390, 460);
    break;
  case 3://quit
    rect(310, 390, 380, 100, 20);//quit
    fill(255, 102, 0);
    text("QUIT", 390, 460);
    fill(100, 100, 100, 100);
    stroke(100, 100, 100);
    rect(310, 150, 380, 100, 20);
    rect(310, 270, 380, 100, 20);//resume
    fill(255, 255, 255);
    text("SAVE", 390, 220);
    text("RESUME", 345, 340);
    break;
  }
}

void importing() {
  image(menuEmpty, 0, 0);
  textSize(24);
  fill(0, 255, 0);
  text("Connectez-vous à la PI en SSH", 10, 50);
  text("Identifiant :pi", 10, 100);
  text("Mot de passe :colorPanic", 10, 150);
  text("entrez la commande suivante :", 10, 200);
  textSize(10);
  textFont(arial);
```

```
   text("\"scp -p ~/colorPanic/actualPlayer.txt
pi@colorPanic:/home/pi/colorPanic/data/actualPlayer.txt\"", 10, 250);
   fill(255, 102, 0, 50);
   stroke(255, 102, 0);
   textFont(font);
   textSize(42);
   switch(column) {
   case 1:
    rect(20, 450, 450, 100, 20);
    fill(255, 102, 0);
    text("RETURN", 100, 515);
    fill(100, 100, 100, 100);
    stroke(100, 100, 100);
    rect(520, 450, 450, 100, 20);
    fill(100, 100, 100);
    text("PLAY", 670, 515);
    break;
   case 2:
    rect(520, 450, 450, 100, 20);
    fill(255, 102, 0);
    text("PLAY", 670, 515);
    fill(100, 100, 100, 100);
    stroke(100, 100, 100);
    rect(20, 450, 450, 100, 20);
    fill(100, 100, 100);
    text("RETURN", 100, 515);
    break;
   }
  }

  void exporting() {
```

```
  image(menuEmpty, 0, 0);
  textSize(24);
  fill(0, 255, 0);
  text("Connectez-vous à la PI en SSH", 10, 50);
  text("Identifiant :pi", 10, 100);
  text("Mot de passe :colorPanic", 10, 150);
  text("entrez la commande suivante :", 10, 200);
  textSize(10);
  textFont(arial);
  text("\"scp -p pi@colorPanic:/home/pi/colorPanic/data/actualPlayer.txt
~/colorPanic/actualPlayer.txt\"", 10, 250);
  fill(255, 102, 0, 50);
  stroke(255, 102, 0);
  textFont(font);
  textSize(42);
  switch(column) {
  case 1:
    rect(20, 450, 450, 100, 20);
    fill(255, 102, 0);
    text("RETURN", 100, 515);
    fill(100, 100, 100, 100);
    stroke(100, 100, 100);
    rect(520, 450, 450, 100, 20);
    fill(100, 100, 100);
    text("SAVE", 670, 515);
    break;
  case 2:
    rect(520, 450, 450, 100, 20);
    fill(255, 102, 0);
    text("SAVE", 670, 515);
    fill(100, 100, 100, 100);
```

```
    stroke(100, 100, 100);
    rect(20, 450, 450, 100, 20);
    fill(100, 100, 100);
    text("RETURN", 100, 515);
    break;
   }
  }

  void endGame() {
   if (levelNumber==59) {
    sound.musicStop(actualMusic);
    sound.musicBegin("data/Sound/Music/musicBurn.mp3");
    interfaces.credit=true;
    levelNumber=0;
    creditTime=millis();
    int time=hour*3600000+minute*60000+second*1000+((millisPaused-initialTime)-
timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute
*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-
(minute*60000)-(hour*3600000);
    int timeDev=8*60000;
    float score=((0.5+(bonusPoints.nbPoints/19))*100000)*(exp(-
hero.nbMort/250)/log(1+(time/timeDev)));
    playerBase=append(playerBase,
pseudo+"|"+bonusPoints.nbPoints+"|59"+hour+":"+minute+":"+"+"+second+":"+str(((millisPaused-
initialTime)-
timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute
*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-
(minute*60000)-(hour*3600000))+"|"+hero.nbMort+"|"+score);
   }
  }

  void leftAction() {//actions faites dans les interfaces avec la touche GAUCHE
   if (interfaces.load==true || interfaces.save==true) {
    if (key==GAUCHE) {
```

```
    if (column==1 && line!=8) {
      column=5;
    } else if (line==8 && column==1) {
      column=2;
    } else if (column==2 && line!=8) {
      line=1;
      column=1;
    } else if (column==1 && line==8) {
      line=1;
      column=1;
    } else if (column==5) {
      column=4;
      line=4;
    } else {
      column--;
    }
  }
} else if (interfaces.leaderboard==true) {
  if (key==GAUCHE) {
    if (line==1) {
      if (column==1) {
        column=6;
      } else {
        column--;
      }
    } else {
      column=1;
      line=1;
    }
  }
} else if (interfaces.importing==true ||interfaces.exporting==true) {
```

```
    if (key==GAUCHE) {
      if (column==1) {
        column=2;
      } else {
        column--;
      }
    }
  } else if (interfaces.setUsername==true) {
    if (key==GAUCHE) {
      if ((line<=1 && column<=2)) {
        line=10;
      } else if (line<=1 && column==3) {
        line=6;
      } else if (line<=1 && column==4) {
        line=2;
      } else {
        line--;
      }
    }
  }
}
void rightAction() {//actions faites dans les interfaces avec la touche DROITE
  if (interfaces.load==true || interfaces.save==true) {
    if (key==DROITE) {
      if (column==5) {
        column=1;
      } else if (column==4) {
        column=5;
        line=1;
      } else if (column==1 && line!=8) {
        column=2;
```

```
      line=4;
    } else if (column==2 && line==8) {
      column=1;
    } else {
      column++;
    }
  }
} else if (interfaces.importing==true || interfaces.exporting==true) {
  if (key==DROITE) {
    if (column==2) {
      column=1;
    } else {
      column++;
    }
  }
} else if (interfaces.leaderboard==true) {
  if (key==DROITE) {
    if (line==1) {
      if (column==6) {
        column=1;
      } else {
        column++;
      }
    } else {
      column=6;
      line=1;
    }
  }
} else if (interfaces.setUsername==true) {
  if (key==DROITE) {
```

```
      if ((line>=10 && (column==1 || column==2))||(line>=6 && column==3) || line>=2 &&
column==4) {
        line=1;
       } else {
        line++;
       }
     }
   }
 }
  void topAction() {//actions faites dans les interfaces avec la touche HAUT
   if (interfaces.ecranTitre==true) {
     if (key==HAUT) {
       if (column==1) {
         column=5;
       } else {
         column--;
       }
     }
   } else if (interfaces.leaderboard==true) {
     if (key==HAUT) {
       if (line==1) {
         line=2;
         column=1;
       } else {
         line--;
         column=3;
       }
     }
   } else if (interfaces.load==true || interfaces.save==true) {
     if (key==HAUT) {
       if (line==1 && column<=3) {
```

```
      line=8;
      column=1;
    } else if (line==1) {
      line=8;
      column=2;
    } else if (line==1 && (column==5 || column==1)) {
      line=2;
      column=3;
    } else if (line==8 && column==1) {
      line=7;
      column=2;
    } else if (line==8 && column==2) {
      line=7;
      column=4;
    } else {
      line--;
    }
  }
} else if (interfaces.ecranTitre==false && interfaces.setUsername==true) {
  if (key==HAUT) {
    if (column==1 && line<=5) {
      line=1;
      column=4;
    } else if (column==1 && line>=6) {
      line=2;
      column=4;
    } else if (column==3) {
      line+=2;
      column=2;
    } else if (column==4 && line==1) {
      line=2;
```

```
      column=3;
    } else if (column==4 && line==2) {
      line=5;
      column=3;
    } else {
      column--;
    }
   }
  } else if (interfaces.pause==true) {
   if (key==HAUT) {
    if (line==1) {
      line=3;
    } else {
      line--;
    }
   }
  }
}
void bottomAction() {//actions faites sur les interfaces avec la touche BAS
  if (interfaces.ecranTitre==true) {
   if (key==BAS) {
    if (column==5) {
      column=1;
    } else {
      column++;
    }
   }
  } else if (interfaces.leaderboard==true) {
   if (key==BAS) {
    if (line==2) {
      line=1;
```

```
      column=3;
    } else if (line==1) {
      line++;
      column=1;
    }
  }
} else if (interfaces.load==true || interfaces.save==true) {
  if (key==BAS) {
    if (line==8 && column==1) {
      line=1;
      column=2;
    } else if (line==8) {
      line=1;
      column=4;
    } else if (column==1 && line==1) {
      line=8;
      column=1;
    } else if (column==5 && line==1) {
      line=8;
      column=2;
    } else if (line==7 && column<=3) {
      line=8;
      column=1;
    } else if (line==7) {
      line=8;
      column=2;
    } else {
      line++;
    }
  }
} else if (interfaces.ecranTitre==false && interfaces.setUsername==true) {
```

```
  if (key==BAS) {
    if (column==2 && line<=2) {
      line=1;
      column=3;
    } else if (column==2 && line>=9) {
      line=6;
      column=3;
    } else if (column==2) {
      line-=2;
      column=3;
    } else if (column==3 && line<=3) {
      line=1;
      column=4;
    } else if (column==3 && line>=4) {
      line=2;
      column=4;
    } else if (column==4 && line==1) {
      line=4;
      column=1;
    } else if (column==4 && line==2) {
      line=7;
      column=1;
    } else {
      column++;
    }
  }
} else if (interfaces.pause==true) {
  if (key==BAS) {
    if (line==3) {
      line=1;
    } else {
```

```
      line++;
     }
    }
  }
}
void confirmAction() {//validation de la case sur laquelle se trouve la selection
  if (interfaces.ecranTitre==true) {
    if (key==ACTION) {
      switch(column) {
      case 1://lance le choix du pseudo avant de joueur
        interfaces.ecranTitre=false;
        interfaces.setUsername=true;
        line=1;
        column=1;
        break;
      case 2://lance l'interface pour charger une partie
        interfaces.ecranTitre=false;
        interfaces.load=true;
        line=1;
        column=2;
        break;
      case 3://lance l'interface pour voir le leaderboard
        interfaces.ecranTitre=false;
        interfaces.leaderboard=true;
        column=2;
        line=1;
        break;
      case 4://lance les crédits
        interfaces.ecranTitre=false;
        interfaces.credit=true;
        creditTime=millis();
```

```
    break;
   case 5://quitte le jeu
    script.endGame();
    break;
   }
  }
} else if (interfaces.ecranTitre==false && interfaces.leaderboard==true) {
  if (key==ACTION) {
   if (line==1) {
    switch(column) {
    case 1:
     if (actualPage==1) {
      actualPage=(int)nbPage;
     } else {
      actualPage--;
     }
     break;
    case 2://affiche le leaderboard selon les pièces
     testedValue=1;
     break;
    case 3://affiche le leaderboard selon les morts
     testedValue=4;
     break;
    case 4://affiche le leaderboard selon le temps
     testedValue=3;
     break;
    case 5://affiche le leaderboard selon le score
     testedValue=5;
     break;
    case 6:
     if (actualPage==(int)nbPage) {
```

```
        actualPage=1;
       } else {
        actualPage++;
       }
       break;
     }
    } else if (line==2 && column==1) {//retourne à l'ecranTitre
     interfaces.ecranTitre=true;
     interfaces.leaderboard=false;
    }
   }
} else if (interfaces.importing==true) {
  if (key==ACTION) {
    if (column==1) {
     interfaces.load=true;
     interfaces.importing=false;
     line=1;
     column=2;
    } else if (column==2) {
     String[] joueur=loadStrings("data/actualPlayer.txt");
     String[] player=split(joueur[1], "|");
     levelNumber=Integer.parseInt(player[2])-1;//place le level sur lequel commencé
     bonusPoints.nbPoints=Integer.parseInt(player[1]);//redonne le nombre de pièces
     hero.nbMort=Integer.parseInt(player[4]);//set le nombre de mort
     String[] time=split(player[3], ":");
     loadedHour=Integer.parseInt(time[0]);//set les heures
     loadedMinute=Integer.parseInt(time[1]);//set les minutes
     loadedSecond=Integer.parseInt(time[2]);//set les secondes
     loadedMillis=Integer.parseInt(time[3]);//set les millisecondes
     hour+=loadedHour;
     minute+=loadedMinute;
```

```
        second+=loadedSecond;
        firstLoadedMillis=loadedMillis;
        pseudo=player[0];
        line=1;
        column=1;
        interfaces.importing=false;
       }
      }
   } else if (interfaces.ecranTitre==false && interfaces.setUsername==true) {//si l'on est dans le choix du pseudo
     if (key==ACTION) {
       if (charChosen=="DELETE" && pseudo.length()>=1) {//si la touche est delete et qu'on peut delete, delete le dernier caractère
         pseudo=pseudo.substring(0, pseudo.length()-1);
       } else if (charChosen=="DELETE" && pseudo.length()==0) {//sinon ne delete rien
       } else if (charChosen=="FINISH") {//si la touche est finish rajoute au fichier playerBase le joueur avec le pseudo choisi et créer un fichier actual player qui est envoyé sur un autre périphérique en ssh si voulu quand on sauvegarde sa partie
         line=1;
         column=1;
         String[] line=new String[2];
         boolean sameUsername=true;
         line[0]="username|coins|actualLevel|timer|death|score";
         line[1]=pseudo+"|0|1|00:00:00:0000|0|0000000";
         saveStrings("data/actualPlayer.txt", line);
         String[] username;//=split(playerBase[1],"|");
         //for(int i=0;i<username.length;i++){
         // println(username[i]);
         //}
         for (int i=0; i<playerBase.length-1; i++) {
           username=split(playerBase[i+1], "|");
           if (username[0]==pseudo) {
```

```
        sameUsername=false;
       }
      }
      if (sameUsername==true) {
       playerBase=append(playerBase, line[1]);
      }
      setUsername=false;
      saveStrings("data/playerBase.txt", playerBase);
      initialTime=millis();
     } else {//sinon ajoute la lettre au pseudo
      pseudo+=charChosen;
     }
    }
   } else if (interfaces.load==true) {//si l'on est dans l'interface de chargement
    if (key==ACTION) {
     if (line==1 && column==1 && actualPage==1) {
      actualPage=(int)nbPage;
     } else if (line==1 && column==1) {
      actualPage--;
     } else if (line==1 && column==5 && actualPage==nbPage) {
      actualPage=1;
     } else if (line==1 && column==5) {
      actualPage++;
     } else if (line==8 && column==1) {//retour à l'écran titre
      interfaces.load=false;
      interfaces.ecranTitre=true;
      line=1;
      column=1;
     } else if (line==8 && column==2) {
      line=1;
      column=1;
```

```
        interfaces.importing=true;
        interfaces.load=false;
       } else {
        if (playerBase.length-1>usernameNumber) {//quand on choisi une partie que l'on souhaite
jouer
          player=split(playerBase[usernameNumber+1], "|");
          levelNumber=Integer.parseInt(player[2])-1;//place le level sur lequel commencé
          bonusPoints.nbPoints=Integer.parseInt(player[1]);//redonne le nombre de pièces
          hero.nbMort=Integer.parseInt(player[4]);//set le nombre de mort
          String[] time=split(player[3], ":");
          loadedHour=Integer.parseInt(time[0]);//set les heures
          loadedMinute=Integer.parseInt(time[1]);//set les minutes
          loadedSecond=Integer.parseInt(time[2]);//set les secondes
          loadedMillis=Integer.parseInt(time[3]);//set les millisecondes
          hour+=loadedHour;
          minute+=loadedMinute;
          second+=loadedSecond;
          firstLoadedMillis=loadedMillis;
          pseudo=player[0];
          interfaces.load=false;//quitte l'interface load et lance le jeu avec les options choisies
          line=1;
          column=1;
        }
        initialTime=millis();
       }
      }
    } else if (interfaces.exporting==true) {
     if (key==ACTION) {
      if (column==1) {
        interfaces.save=true;
        interfaces.exporting=false;
```

```
        line=1;
        column=2;
       } else {
        interfaces.pause=true;
        interfaces.exporting=false;
        line=1;
        column=1;
       }
      }
    } else if (interfaces.pause==true) {
     if (key==ACTION) {
      if (line==1) {
        interfaces.pause=false;
        interfaces.save=true;
        interfaces.firstLoad=true;
      } else if (line==2) {
        interfaces.pause=false;
      } else {//retour au menu de démarrage
        playerBase=append(playerBase,
pseudo+"|"+bonusPoints.nbPoints+"|"+(levelNumber+1)+"|"+hour+":"+minute+":"+second+":"+str((
(millisPaused-initialTime)-
timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute
*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-
(minute*60000)-(hour*3600000))+"|"+hero.nbMort+"|0000000");
        saveStrings("data/playerBase.txt", playerBase);
        usersLength++;
        playerBase=loadStrings("data/playerBase.txt");
        bonusPoints.nbPoints=0;
        levelNumber=0;
        hour=0;
        minute=0;
        second=0;
        millisPaused=0;
```

```
    initialTime=0;
    timeStopped=0;
    interfaces.is1000=0;
    interfaces.loadedHour=0;
    interfaces.loadedMinute=0;
    interfaces.loadedSecond=0;
    interfaces.firstLoadedMillis=0;
    hero.nbMort=0;
    interfaces.ecranTitre=true;
    interfaces.pause=false;
   }
  }
 } else if (interfaces.save==true) {//si l'on est sur l'interface de sauvegarde
  if (key==ACTION) {
   if (line==1 && column==1 && actualPage==1) {
    actualPage=(int)nbPage;
   } else if (line==1 && column==1) {
    actualPage--;
   } else if (line==1 && column==5 && actualPage==nbPage) {
    actualPage=1;
   } else if (line==1 && column==5) {
    actualPage++;
   } else if (line==8 && column==1) {//quitte l'interface de sauvegarde
    interfaces.save=false;
    interfaces.pause=true;
    line=1;
    column=1;
   } else if (line==8 && column==2) {
    interfaces.save=false;
    interfaces.exporting=true;
    line=1;
```

```
        column=1;
       } else {
        if (playerBase.length-1>usernameNumber) {//sauvegarde les informations actuelles de la partie dans le fichier playerBase

playerBase[usernameNumber+1]=pseudo+"|"+bonusPoints.nbPoints+"|"+(levelNumber+1)+"|"+hour+":"+minute+":"+second+":"+str(((millisPaused-initialTime)-timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-(minute*60000)-(hour*3600000))+"|"+hero.nbMort+"|0000000";
          saveStrings("data/playerBase.txt", playerBase);
         } else {
          playerBase=append(playerBase, pseudo+"|"+bonusPoints.nbPoints+"|"+(levelNumber+1)+"|"+hour+":"+minute+":"+second+":"+str(((millisPaused-initialTime)-timeStopped+(1000*interfaces.is1000)+(interfaces.loadedHour*3600000)+(interfaces.loadedMinute*60000)+(interfaces.loadedSecond*1000)+(interfaces.firstLoadedMillis))-(second*1000)-(minute*60000)-(hour*3600000))+"|"+hero.nbMort+"|0000000");
          saveStrings("data/playerBase.txt", playerBase);
          usersLength++;
         }
        playerBase=loadStrings("data/playerBase.txt");
       }
     }
   }
 }
}
```

Script.pde :

```
class Script {


void Bashrun(String n) {


  String commandToRun = n;
```

```
File workingDir = new File(sketchPath(""));   // where to do it - should be full path
String returnedValues;                        // value to return any results

// give us some info:
println("Running command: " + commandToRun);
println("Location:       " + workingDir);
println("--------------------------------------------\n");

// run the command!
try {

 // complicated!  basically, we have to load the exec command within Java's Runtime
 // exec asks for 1. command to run, 2. null which essentially tells Processing to
 // inherit the environment settings from the current setup (I am a bit confused on
 // this so it seems best to leave it), and 3. location to work (full path is best)
 Process p = Runtime.getRuntime().exec(commandToRun, null, workingDir);

 // variable to check if we've received confirmation of the command
 int i = p.waitFor();

 // if we have an output, print to screen
 if (i == 0) {

  // BufferedReader used to get values back from the command
  BufferedReader stdInput = new BufferedReader(new InputStreamReader(p.getInputStream()));

  // read the output from the command
  while ( (returnedValues = stdInput.readLine ()) != null) {
   println(returnedValues);
  }
```

```
    }

    // if there are any error messages but we can still get an output, they print here
    else {
      BufferedReader stdErr = new BufferedReader(new InputStreamReader(p.getErrorStream()));

      // if something is returned (ie: not null) print the result
      while ( (returnedValues = stdErr.readLine ()) != null) {
        println(returnedValues);
      }
    }
  }

  // if there is an error, let us know
  catch (Exception e) {
    println("Error running command!");
    println(e);
    // e.printStackTrace();    // a more verbose debug, if needed
  }

  // when done running command, quit
  println("\n----------------------------------------------");
  println("DONE!");
  exit();
}



  public void endGame() {
    String os=System.getProperty("os.name");
    println(os);
```

```
   if (os.equals("Linux")) {
     Bashrun("./shutdown.sh");
   } else  {
     exit();
   }
  }

  public void saveload(Boolean s){
   if (s){
     Bashrun("data/Scripts/load.sh");
   } else {
     Bashrun("data/Scripts/save.sh");
   }
  }

}
```

Sound.pde :

```
class Sound {
  boolean firstMusic=true;
  int actualMusicTimer=0;
  void musicChange() {//fonction qui lance la musique du niveau suivant si elle est différente du niveau actuel (juste avant de passer un niveau)
   if (levelNumber!=59 && !hitboxLvl[levelNumber][128*38].equals(hitboxLvl[levelNumber-1][128*38])) {
     String nameMusic=hitboxLvl[levelNumber][128*38];
     actualMusic.close();
     actualMusic=musicBegin("data/Sound/Music/"+nameMusic+".mp3");
   }
  }

  void musicStop(AudioPlayer music) {//fonction qui arrête la musique courante et la rembobine
```

```
  music.skip(-music.length());
  music.pause();
}

AudioPlayer musicBegin(String musique) {//fonction qui permet de démarrer la musique courante
  AudioPlayer music=minim.loadFile(musique);
  music.play();
  music.loop();
  return music;
}

void musicLoop() {//fonction qui fait tourner en boucle la musique courante
  if (actualMusic.position()==actualMusic.length()) {
    actualMusic.rewind();
    actualMusic.play();
  }
}

void musicFirst() {//fonction qui démarre la première musique du jeu
  if (firstMusic==true) {
    musicStop(actualMusic);
    firstMusic=false;
    actualMusic.close();
    mvtInterface.close();
    validationInterface.close();
    death=minim.loadSample("data/Sound/SFX/Death.mp3");
    jump=minim.loadSample("data/Sound/SFX/Jump.mp3");
    powerup=minim.loadSample("data/Sound/SFX/Powerup.mp3");
    TP=minim.loadSample("data/Sound/SFX/TP.mp3");
    String nameMusic=hitboxLvl[levelNumber][128*38];
    actualMusic=musicBegin("data/Sound/Music/"+nameMusic+".mp3");
```

```
   }
  }
}
```

AppletV8.pde :

```
int x, y;
boolean isSaving=false;
String R="0", G="0", B="0";
boolean selectR=false, selectG=false, selectB=false, selectName=false, selectEnd=false, noClip=false;
String []type=new String[0];
String types="empty", lvl="lvl1", end="end", noClipValue="", musicChosen="musicColorPanic";
PImage[] wallRed=new PImage[26], wallDarkBlue=new PImage[26], wallLightBlue=new PImage[26],
wallGreen=new PImage[26], wallPurple=new PImage[26], spike=new PImage[20];
PImage img, save, door;
void setup() {
 size(1624, 900);
 for (int i=0; i<=25; i++) {
  wallRed[i]=loadImage("BLOCKS/Red/Red"+i+".png");
  wallDarkBlue[i]=loadImage("BLOCKS/DarkBlue/DarkBlue"+i+".png");
  wallLightBlue[i]=loadImage("BLOCKS/LightBlue/LightBlue"+i+".png");
  wallGreen[i]=loadImage("BLOCKS/Green/Green"+i+".png");
  wallPurple[i]=loadImage("BLOCKS/Purple/Purple"+i+".png");
 }
 for (int i=0; i<20; i++) {// BOTTOM/LEFT/TOP/RIGHT DB/G/LB/P/R
  spike[i]=loadImage("BLOCKS/spike/spike"+i+".png");
 }
 img=loadImage("BLOCKS/Empty.png");
 door=loadImage("BLOCKS/DOOR.png");
 frameRate(15);
 type=loadStrings("../../data/levels/lvlsHitbox/"+lvl+".txt");
 if (type.length!=128*38+1) {
```

```
   for (int i=0; i<128*38; i++) {
     type=append(type, types);
   }
   type=append(type, "musicRetroRide");
  }
  musicChosen=type[128*38];
}

void draw() {
  background(40, 10, 10);
  changeBackground();
  printImage();
}
void changeBackground() {
  if (R.length()==0 || G.length()==0 || B.length()==0) {
    fill(0, 0, 0);
  } else {
    fill(Integer.parseInt(R), Integer.parseInt(G), Integer.parseInt(B));
  }
  rect(50, 50, 1024, 600);
  fill(255, 255, 255, 100);
  stroke(255, 255, 255);
  fill(10, 10, 10);
  rect(80, 770, 170, 35, 5);
  rect(80, 810, 170, 35, 5);
  rect(80, 850, 170, 35, 5);
  fill(255, 255, 255);
  text("R : "+R, 100, 795);
  text("G : "+G, 100, 835);
  text("B : "+B, 100, 875);
  if (mousePressed==true && x>=80 && x<=250 && y>=770 && y<=805) {
```

```
  selectR=true;
  selectB=false;
  selectG=false;
  selectName=false;
  selectEnd=false;
}
if (mousePressed==true && x>=80 && x<=250 && y>=810 && y<=845) {
  selectG=true;
  selectR=false;
  selectB=false;
  selectName=false;
  selectEnd=false;
}
if (mousePressed==true && x>=80 && x<=250 && y>=850 && y<=885) {
  selectB=true;
  selectG=false;
  selectR=false;
  selectName=false;
  selectEnd=false;
}
if (selectR==true && keyPressed==true) {
  if (key==BACKSPACE && R.length()>0) {
    R=R.substring(0, R.length()-1);
  } else if (keyCode>=48 && keyCode<=57 && R.length()<3) {
    R+=key;
  }
}
if (selectG==true && keyPressed==true) {
  if (key==BACKSPACE && G.length()>0) {
    G=G.substring(0, G.length()-1);
  } else if (keyCode>=48 && keyCode<=57 && G.length()<3) {
```

```
    G+=key;
   }
  }
  if (selectB==true && keyPressed==true) {
   if (key==BACKSPACE && B.length()>0) {
    B=B.substring(0, B.length()-1);
   } else if (keyCode>=48 && keyCode<=57 && B.length()<3) {
    B+=key;
   }
  }
  if (R.length()==3 && Integer.parseInt(R)>255) {
   R="255";
  }
  if (G.length()==3 && Integer.parseInt(G)>255) {
   G="255";
  }
  if (B.length()==3 && Integer.parseInt(B)>255) {
   B="255";
  }
}
void hitbox() {
  x=mouseX;
  y=mouseY;
  if (noClip==true) {
   noClipValue="NoClip";
  } else {
   noClipValue="";
  }
  if (x>=1100 && x<=1132) {
   if (y>=50 && y<=82) {
    types="wall"+noClipValue+"BottomPurple";
```

```
}
if (y>=90 && y<=122) {
  types="wall"+noClipValue+"BottomLeftPurple";
}
if (y>=130 && y<=162) {
  types="wall"+noClipValue+"BottomLeftCornerPurple";
}
if (y>=170 && y<=202) {
  types="wall"+noClipValue+"BottomRightPurple";
}
if (y>=210 && y<=242) {
  types="wall"+noClipValue+"BottomRightCornerPurple";
}
if (y>=250 && y<=282) {
  types="wall"+noClipValue+"CenterPurple";
}
if (y>=290 && y<=322) {
  types="wall"+noClipValue+"LeftPurple";
}
if (y>=330 && y<=362) {
  types="emptywall"+noClipValue+"LineBottomPurple";
}
if (y>=370 && y<=402) {
  types="wall"+noClipValue+"LineBottomLeftPurple";
}
if (y>=410 && y<=442) {
  types="emptywall"+noClipValue+"LineBottomLeftCornerPurple";
}
if (y>=450 && y<=482) {
  types="emptywall"+noClipValue+"LineBottomRightPurple";
}
```

```
  if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineBottomRightCornerPurple";
  }
  if (y>=530 && y<=562) {
    types="wall"+noClipValue+"LineLeftPurple";
  }
 } else if (x>=1150 && x<=1182) {
  if (y>=530 && y<=562) {
    types="emptywall"+noClipValue+"LineTopRightCornerPurple";
  }
  if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineRightPurple";
  }
  if (y>=450 && y<=482) {
    types="wall"+noClipValue+"LineTopPurple";
  }
  if (y>=410 && y<=442) {
    types="wall"+noClipValue+"LineTopLeftPurple";
  }
  if (y>=370 && y<=402) {
    types="wall"+noClipValue+"LineTopLeftCornerPurple";
  }
  if (y>=330 && y<=362) {
    types="wall"+noClipValue+"LineTopRightPurple";
  }
  if (y>=290 && y<=322) {
    types="wall"+noClipValue+"ChiseledPurple";
  }
  if (y>=250 && y<=282) {
    types="wall"+noClipValue+"RightPurple";
  }
```

```
  if (y>=210 && y<=242) {
    types="wallTop"+noClipValue+"Purple";
  }
  if (y>=170 && y<=202) {
    types="wall"+noClipValue+"TopLeftPurple";
  }
  if (y>=130 && y<=162) {
    types="wall"+noClipValue+"TopLeftCornerPurple";
  }
  if (y>=90 && y<=122) {
    types="wall"+noClipValue+"TopRightPurple";
  }
  if (y>=50 && y<=82) {
    types="wall"+noClipValue+"TopRightCornerPurple";
  }
 } else if (x>=1200 && x<=1232) {
  if (y>=50 && y<=82) {
    types="wall"+noClipValue+"BottomRed";
  }
  if (y>=90 && y<=122) {
    types="wall"+noClipValue+"BottomLeftRed";
  }
  if (y>=130 && y<=162) {
    types="wall"+noClipValue+"BottomLeftCornerRed";
  }
  if (y>=170 && y<=202) {
    types="wall"+noClipValue+"BottomRightRed";
  }
  if (y>=210 && y<=242) {
    types="wall"+noClipValue+"BottomRightCornerRed";
  }
```

```
  if (y>=250 && y<=282) {
    types="wall"+noClipValue+"CenterRed";
  }
  if (y>=290 && y<=322) {
    types="wall"+noClipValue+"LeftRed";
  }
  if (y>=330 && y<=362) {
    types="emptywall"+noClipValue+"LineBottomRed";
  }
  if (y>=370 && y<=402) {
    types="wall"+noClipValue+"LineBottomLeftRed";
  }
  if (y>=410 && y<=442) {
    types="emptywall"+noClipValue+"LineBottomLeftCornerRed";
  }
  if (y>=450 && y<=482) {
    types="emptywall"+noClipValue+"LineBottomRightRed";
  }
  if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineBottomRightCornerRed";
  }
  if (y>=530 && y<=562) {
    types="wall"+noClipValue+"LineLeftRed";
  }
 } else if (x>=1250 && x<=1282) {
  if (y>=530 && y<=562) {
    types="emptywall"+noClipValue+"LineTopRightCornerRed";
  }
  if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineRightRed";
  }
```

```
if (y>=450 && y<=482) {
  types="wall"+noClipValue+"LineTopRed";
}
if (y>=410 && y<=442) {
  types="wall"+noClipValue+"LineTopLeftRed";
}
if (y>=370 && y<=402) {
  types="wall"+noClipValue+"LineTopLeftCornerRed";
}
if (y>=330 && y<=362) {
  types="wall"+noClipValue+"LineTopRightRed";
}
if (y>=290 && y<=322) {
  types="wall"+noClipValue+"ChiseledRed";
}
if (y>=250 && y<=282) {
  types="wall"+noClipValue+"RightRed";
}
if (y>=210 && y<=242) {
  types="wall"+noClipValue+"TopRed";
}
if (y>=170 && y<=202) {
  types="wall"+noClipValue+"TopLeftRed";
}
if (y>=130 && y<=162) {
  types="wall"+noClipValue+"TopLeftCornerRed";
}
if (y>=90 && y<=122) {
  types="wall"+noClipValue+"TopRightRed";
}
if (y>=50 && y<=82) {
```

```
   types="wall"+noClipValue+"TopRightCornerRed";
  }
 } else if (x>=1300 && x<=1332) {
  if (y>=50 && y<=82) {
   types="wall"+noClipValue+"BottomDarkBlue";
  }
  if (y>=90 && y<=122) {
   types="wall"+noClipValue+"BottomLeftDarkBlue";
  }
  if (y>=130 && y<=162) {
   types="wall"+noClipValue+"BottomLeftCornerDarkBlue";
  }
  if (y>=170 && y<=202) {
   types="wall"+noClipValue+"BottomRightDarkBlue";
  }
  if (y>=210 && y<=242) {
   types="wall"+noClipValue+"BottomRightCornerDarkBlue";
  }
  if (y>=250 && y<=282) {
   types="wall"+noClipValue+"CenterDarkBlue";
  }
  if (y>=290 && y<=322) {
   types="wall"+noClipValue+"LeftDarkBlue";
  }
  if (y>=330 && y<=362) {
   types="emptywall"+noClipValue+"LineBottomDarkBlue";
  }
  if (y>=370 && y<=402) {
   types="wall"+noClipValue+"LineBottomLeftDarkBlue";
  }
  if (y>=410 && y<=442) {
```

```
    types="emptywall"+noClipValue+"LineBottomLeftCornerDarkBlue";
   }
   if (y>=450 && y<=482) {
    types="emptywall"+noClipValue+"LineBottomRightDarkBlue";
   }
   if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineBottomRightCornerDarkBlue";
   }
   if (y>=530 && y<=562) {
    types="wall"+noClipValue+"LineLeftDarkBlue";
   }
  } else if (x>=1350 && x<=1382) {
   if (y>=530 && y<=562) {
    types="emptywall"+noClipValue+"LineTopRightCornerDarkBlue";
   }
   if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineRightDarkBlue";
   }
   if (y>=450 && y<=482) {
    types="wall"+noClipValue+"LineTopDarkBlue";
   }
   if (y>=410 && y<=442) {
    types="wall"+noClipValue+"LineTopLeftDarkBlue";
   }
   if (y>=370 && y<=402) {
    types="wall"+noClipValue+"LineTopLeftCornerDarkBlue";
   }
   if (y>=330 && y<=362) {
    types="wall"+noClipValue+"LineTopRightDarkBlue";
   }
   if (y>=290 && y<=322) {
```

```
  types="wall"+noClipValue+"ChiseledDarkBlue";
 }
 if (y>=250 && y<=282) {
  types="wall"+noClipValue+"RightDarkBlue";
 }
 if (y>=210 && y<=242) {
  types="wall"+noClipValue+"TopDarkBlue";
 }
 if (y>=170 && y<=202) {
  types="wall"+noClipValue+"TopLeftDarkBlue";
 }
 if (y>=130 && y<=162) {
  types="wall"+noClipValue+"TopLeftCornerDarkBlue";
 }
 if (y>=90 && y<=122) {
  types="wall"+noClipValue+"TopRightDarkBlue";
 }
 if (y>=50 && y<=82) {
  types="wall"+noClipValue+"TopRightCornerDarkBlue";
 }
} else if (x>=1400 && x<=1432) {
 if (y>=50 && y<=82) {
  types="wall"+noClipValue+"BottomLightBlue";
 }
 if (y>=90 && y<=122) {
  types="wall"+noClipValue+"BottomLeftLightBlue";
 }
 if (y>=130 && y<=162) {
  types="wall"+noClipValue+"BottomLeftCornerLightBlue";
 }
 if (y>=170 && y<=202) {
```

```
  types="wall"+noClipValue+"BottomRightLightBlue";
 }
 if (y>=210 && y<=242) {
  types="wall"+noClipValue+"BottomRightCornerLightBlue";
 }
 if (y>=250 && y<=282) {
  types="wall"+noClipValue+"CenterLightBlue";
 }
 if (y>=290 && y<=322) {
  types="wall"+noClipValue+"LeftLightBlue";
 }
 if (y>=330 && y<=362) {
  types="emptywall"+noClipValue+"LineBottomLightBlue";
 }
 if (y>=370 && y<=402) {
  types="wall"+noClipValue+"LineBottomLeftLightBlue";
 }
 if (y>=410 && y<=442) {
  types="emptywall"+noClipValue+"LineBottomLeftCornerLightBlue";
 }
 if (y>=450 && y<=482) {
  types="emptywall"+noClipValue+"LineBottomRightLightBlue";
 }
 if (y>=490 && y<=522) {
  types="emptywall"+noClipValue+"LineBottomRightCornerLightBlue";
 }
 if (y>=530 && y<=562) {
  types="wall"+noClipValue+"LineLeftLightBlue";
 }
} else if (x>=1450 && x<=1482) {
 if (y>=530 && y<=562) {
```

```
 types="emptywall"+noClipValue+"LineTopRightCornerLightBlue";
}
if (y>=490 && y<=522) {
 types="emptywall"+noClipValue+"LineRightLightBlue";
}
if (y>=450 && y<=482) {
 types="wall"+noClipValue+"LineTopLightBlue";
}
if (y>=410 && y<=442) {
 types="wall"+noClipValue+"LineTopLeftLightBlue";
}
if (y>=370 && y<=402) {
 types="wall"+noClipValue+"LineTopLeftCornerLightBlue";
}
if (y>=330 && y<=362) {
 types="wall"+noClipValue+"LineTopRightLightBlue";
}
if (y>=290 && y<=322) {
 types="wall"+noClipValue+"ChiseledLightBlue";
}
if (y>=250 && y<=282) {
 types="wall"+noClipValue+"RightLightBlue";
}
if (y>=210 && y<=242) {
 types="wall"+noClipValue+"TopLightBlue";
}
if (y>=170 && y<=202) {
 types="wall"+noClipValue+"TopLeftLightBlue";
}
if (y>=130 && y<=162) {
 types="wall"+noClipValue+"TopLeftCornerLightBlue";
```

```
  }
  if (y>=90 && y<=122) {
    types="wall"+noClipValue+"TopRightLightBlue";
  }
  if (y>=50 && y<=82) {
    types="wall"+noClipValue+"TopRightCornerLightBlue";
  }
 } else if (x>=1500 && x<=1532) {
  if (y>=50 && y<=82) {
    types="wall"+noClipValue+"BottomGreen";
  }
  if (y>=90 && y<=122) {
    types="wall"+noClipValue+"BottomLeftGreen";
  }
  if (y>=130 && y<=162) {
    types="wall"+noClipValue+"BottomLeftCornerGreen";
  }
  if (y>=170 && y<=202) {
    types="wall"+noClipValue+"BottomRightGreen";
  }
  if (y>=210 && y<=242) {
    types="wall"+noClipValue+"BottomRightCornerGreen";
  }
  if (y>=250 && y<=282) {
    types="wall"+noClipValue+"GreenCenter";
  }
  if (y>=290 && y<=322) {
    types="wall"+noClipValue+"LeftGreen";
  }
  if (y>=330 && y<=362) {
    types="emptywall"+noClipValue+"LineBottomGreen";
```

```
  }
  if (y>=370 && y<=402) {
    types="wall"+noClipValue+"LineBottomLeftGreen";
  }
  if (y>=410 && y<=442) {
    types="emptywall"+noClipValue+"LineBottomLeftCornerGreen";
  }
  if (y>=450 && y<=482) {
    types="emptywall"+noClipValue+"LineBottomRightGreen";
  }
  if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineBottomRightCornerGreen";
  }
  if (y>=530 && y<=562) {
    types="wall"+noClipValue+"LineLeftGreen";
  }
 } else if (x>=1550 && x<=1582) {
  if (y>=530 && y<=562) {
    types="emptywall"+noClipValue+"LineTopRightCornerGreen";
  }
  if (y>=490 && y<=522) {
    types="emptywall"+noClipValue+"LineRightGreen";
  }
  if (y>=450 && y<=482) {
    types="wall"+noClipValue+"LineTopGreen";
  }
  if (y>=410 && y<=442) {
    types="wall"+noClipValue+"LineTopLeftGreen";
  }
  if (y>=370 && y<=402) {
    types="wall"+noClipValue+"LineTopLeftCornerGreen";
```

```
 }
 if (y>=330 && y<=362) {
  types="wall"+noClipValue+"LineTopRightGreen";
 }
 if (y>=290 && y<=322) {
  types="wall"+noClipValue+"ChiseledGreen";
 }
 if (y>=250 && y<=282) {
  types="wall"+noClipValue+"RightGreen";
 }
 if (y>=210 && y<=242) {
  types="wall"+noClipValue+"TopGreen";
 }
 if (y>=170 && y<=202) {
  types="wall"+noClipValue+"TopLeftGreen";
 }
 if (y>=130 && y<=162) {
  types="wall"+noClipValue+"TopLeftCornerGreen";
 }
 if (y>=90 && y<=122) {
  types="wall"+noClipValue+"TopRightGreen";
 }
 if (y>=50 && y<=82) {
  types="wall"+noClipValue+"TopRightCornerGreen";
 }
 if (y>=290 && y<=322) {
  types="wall"+noClipValue+"ChiseledGreen";
 }
}
//////////////////////////////////////////////////////////////
```

```
if (y>=630 && y<=662) {
  if (x>=1100 && x<=1132) {
    types="emptyspikeDarkBlueBottom";
  }
  if (x>=1140 && x<=1172) {
    types="spikeDarkBlueLeft";
  }
  if (x>=1180 && x<=1212) {
    types="spikeDarkBlueTop";
  }
  if (x>=1220 && x<=1252) {
    types="emptyspikeDarkBlueRight";
  }
  if (x>=1260 && x<=1292) {
    types="emptyspikeGreenBottom";
  }
  if (x>=1300 && x<=1332) {
    types="spikeGreenLeft";
  }
  if (x>=1340 && x<=1372) {
    types="spikeGreenTop";
  }
  if (x>=1380 && x<=1412) {
    types="emptyspikeGreenRight";
  }
  if (x>=1420 && x<=1452) {
    types="emptyspikeLightBlueBottom";
  }
  if (x>=1460 && x<=1492) {
    types="spikeLightBlueLeft";
  }
```

```
  if (x>=1500 && x<=1532) {
    types="spikeLightBlueTop";
  }
  if (x>=1540 && x<=1572) {
    types="emptyspikeLightBlueRight";
  }
 } else if (y>=670 && y<=702) {
  if (x>=1180 && x<=1212) {
    types="emptyspikePurpleBottom";
  }
  if (x>=1220 && x<=1252) {
    types="spikePurpleLeft";
  }
  if (x>=1260 && x<=1292) {
    types="spikePurpleTop";
  }
  if (x>=1300 && x<=1332) {
    types="emptyspikePurpleRight";
  }
  if (x>=1340 && x<=1372) {
    types="emptyspikeRedBottom";
  }
  if (x>=1380 && x<=1412) {
    types="spikeRedLeft";
  }
  if (x>=1420 && x<=1452) {
    types="spikeRedTop";
  }
  if (x>=1460 && x<=1492) {
    types="emptyspikeRedRight";
  }
```

```
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
if (y>=573 && y<=618) {
  if (x>=1080 && x<=1125) {
    types="BJump";
  }
  if (x>=1135 && x<=1180) {
    types="empty";
  }
  if (x>=1190 && x<=1235) {
    types="BTP";
  }
  if (x>=1245 && x<=1290 && noClip==false) {
    noClip=true;
  } else if (x>=1245 && x<=1290 && noClip==true) {
    noClip=false;
  }
  if (x>=1300 && x<=1345) {
    types="BnoClip";
  }
  if (x>=1355 && x<=1400) {
    types="hero";
  }
  if (x>=1410 && x<=1455) {
    types="BDash";
  }
  if (x>=1465 && x<=1510) {
    types=end;
  }
  if (x>=1520 && x<=1565) {
```

```
    types="BGSwap";
  }
  if (x>=1575 && x<=1620) {
    types="BPoints";
  }
}
if (x>=1100 && x<=1164 && y>=720 && y<=784) {
  types="door"+end.substring(3, end.length());
  println(types);
}
if (x>=1170 && x<=1335) {
  if (y>=730 && y<=760) {
    type[128*38]="musicBurn";
    musicChosen="musicBurn";
  } else if (y>=765 && y<=795) {
    type[128*38]="musicColorPanic";
    musicChosen="musicColorPanic";
  } else if (y>=800 && y<=830) {
    type[128*38]="musicJourneyBegin";
    musicChosen="musicJourneyBegin";
  } else if (y>=835 && y<=865) {
    type[128*38]="musicNewPower";
    musicChosen="musicNewPower";
  }
}
if (x>=1340 && x<=1505) {
  if (y>=730 && y<=760) {
    type[128*38]="musicRetroRide";
    musicChosen="musicRetroRide";
  } else if (y>=765 && y<=795) {
    type[128*38]="musicRise";
```

```
    musicChosen="musicRise";
  } else if (y>=800 && y<=830) {
    type[128*38]="musicTheLastBattle";
    musicChosen="musicTheLastBattle";
  } else if (y>=835 && y<=865) {
    type[128*38]="musicTheOne";
    musicChosen="musicTheOne";
  }
}
if (x>=1510 && x<=1610 && y>=730 && y<=760) {
  type[128*38]="musicValk";
  musicChosen="musicValk";
}

////////////////////////////////////////////////////////////////////////////////////////////////////

if (x>=300 && x<=470 && y>=700 && y<=750) {
  saveStrings("../../data/levels/lvlsHitbox/"+lvl+".txt", type);
  isSaving=true;
}
if (x>=100 && x<=250 && y>=700 && y<=750) {
  File file = new File(dataPath("../../../data/levels/lvlsHitbox"));
  boolean doesLvlExist=false;
  String[] lvls = file.list(), empty=new String[128*38+1];
  for (int i=0; i<lvls.length; i++) {
    if (lvls[i].equals(lvl+".txt")) {
      doesLvlExist=true;
    }
  }
  if (doesLvlExist==false) {
    saveStrings("../../data/levels/lvlsHitbox/"+lvl+".txt", empty);
```

```
    isSaving=true;
   }
   type=loadStrings("../../data/levels/lvlsHitbox/"+lvl+".txt");
  }
  if (x>=300 && x<=470 && y>=770 && y<=820) {
   for (int i=0; i<128*38; i++) {
    type[i]="empty";
   }
  }
}
void printImage() {
  fill(0, 0, 0);
  rect(500, 700, 500, 50, 5);
  rect(500, 770, 500, 50, 5);
  fill(255, 255, 255);
  text("nom du fichier: "+lvl, 510, 730);
  text("niveau suivant: "+end, 510, 800);
  if (mousePressed==true && x>=500 && x<=1000 && y>=700 && y<=750) {
   selectName=true;
   selectR=false;
   selectG=false;
   selectB=false;
   selectEnd=false;
  }
  if (mousePressed==true && x>=500 && x<=1000 && y>=770 && y<=820) {
   selectName=false;
   selectR=false;
   selectG=false;
   selectB=false;
   selectEnd=true;
  }
```

```
 if (keyPressed==true && selectEnd==true) {
  if (key==BACKSPACE && end.length()>3) {
   end=end.substring(0, end.length()-1);
  } else if (end.length()<20 && ((keyCode>=48 && keyCode<=57) || (keyCode>=65 && keyCode<=90)
||(keyCode>=97 && keyCode<=122))) {
   end+=key;
  }
 }
 if (keyPressed==true && selectName==true) {
  if (key==BACKSPACE && lvl.length()>3) {
   lvl=lvl.substring(0, lvl.length()-1);
  } else if (end.length()<20 && ((keyCode>=48 && keyCode<=57) || (keyCode>=65 && keyCode<=90)
||(keyCode>=97 && keyCode<=122))) {
   lvl+=key;
  }
 }
 fill(0);
 stroke(0);
 for (int i=0; i<13; i++) {
  for (int j=0; j<10; j++) {
   rect(1100+50*j, i*40+50, 32, 32);
  }
  image(wallPurple[i], 1100, (i*40)+50);
  image(wallPurple[25-i], 1150, i*40+50);
  image(wallRed[i], 1200, 50+i*40);
  image(wallRed[25-i], 1250, 50+i*40);
  image(wallDarkBlue[i], 1300, (i*40)+50);
  image(wallDarkBlue[25-i], 1350, (i*40)+50);
  image(wallLightBlue[i], 1400, (i*40)+50);
  image(wallLightBlue[25-i], 1450, (i*40)+50);
  image(wallGreen[i], 1500, (i*40)+50);
  image(wallGreen[25-i], 1550, (i*40)+50);
```

```
}
for (int i=0; i<12; i++) {
  stroke(0, 0, 0);
  rect(1100+(i*40), 630, 32, 32);
  image(spike[i], 1100+(i*40), 630);
}
for (int i=0; i<8; i++) {
  rect(1180+(i*40), 670, 32, 32);
  image(spike[12+i], 1180+(i*40), 670);
}
image(door, 1100, 720);
fill(255, 255, 255);
text("x="+x, 110, 685);
text("y="+y, 210, 685);
text("sauvegarde", 320, 730);
text("actualiser", 120, 730);
text("réinitialiser", 320, 800);
text(types, 290, 685);
textSize(13);
text("BJump", 1083, 600);
text("empty", 1139, 600);
text("BTP", 1203, 600);
if (noClip==true) {
  fill(0, 255, 0);
} else {
  fill(255, 0, 0);
}
text("noClip", 1248, 600);
fill(255, 255, 255);
textSize(12);
text("BnoClip", 1300, 600);
```

```
textSize(13);
text("hero", 1363, 600);
text("BDash", 1412, 600);
text("end", 1475, 600);
textSize(11);
text("BGSwap", 1523, 600);
textSize(13);
text("BPoints", 1576, 600);
stroke(255, 255, 0);
fill(255, 255, 255, 20);
rect(1300, 573, 45, 45);  //5 BnoClip

stroke(0, 255, 0);
fill(0, 255, 0, 100);
rect(1080, 573, 45, 45); //1 bonusDoubleJump
stroke(153, 0, 0);
fill(153, 0, 0, 100);
rect(1410, 573, 45, 45); //7 BDash
stroke(142, 64, 30);
fill(142, 64, 30, 100);
rect(1135, 573, 45, 45); //2 empty
fill(140, 24, 202, 100);
stroke(140, 24, 202);
rect(1190, 573, 45, 45); //3 BTP
stroke(255, 255, 255);
fill(100, 100, 100, 100);
rect(1245, 573, 45, 45); //4 noClip
stroke(255, 20, 147);
fill(255, 20, 147, 100);
rect(1355, 573, 45, 45); //6 hero
stroke(0, 255, 255);
```

```
fill(0, 255, 255, 100);
rect(1465, 573, 45, 45); //8 end
fill(30, 87, 142, 100);
stroke(30, 87, 142);
rect(1520, 573, 45, 45); //9 BGSwap
stroke(255, 255, 0);
fill(255, 255, 0, 100);
rect(1575, 573, 45, 45);   //10 BPoints
stroke(255, 255, 255);
fill(255, 255, 255, 100);
rect(300, 700, 170, 50, 8);
rect(100, 700, 150, 50, 8);
rect(300, 770, 170, 50, 5);
//---------------------------------------------------------//
textSize(20);
fill(0, 0, 0);
switch(type[128*38]) {
case "musicBurn":
  fill(0, 255, 0);
  text("Burn", 1230, 750);
  fill(0, 0, 0);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
case "musicColorPanic":
```

```
  text("Burn", 1230, 750);
  fill(0, 255, 0);
  text("Color Panic", 1200, 785);
  fill(0, 0, 0);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
 case"musicJourneyBegin":
  text("Burn", 1230, 750);
  text("Color Panic", 1200, 785);
  fill(0, 255, 0);
  text("Journey Begins", 1185, 820);
  fill(0, 0, 0);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
 case "musicNewPower":
  text("Burn", 1230, 750);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  fill(0, 255, 0);
  text("New Power", 1200, 855);
```

```
  fill(0, 0, 0);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
 case "musicRetroRide":
  text("Burn", 1230, 750);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  fill(0, 255, 0);
  text("Retro Ride", 1370, 750);
  fill(0, 0, 0);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
 case "musicRise":
  text("Burn", 1230, 750);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  fill(0, 255, 0);
  text("Rise", 1400, 785);
  fill(0, 0, 0);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
```

```
  text("Valk", 1535, 750);
  break;
 case"musicTheLastBattle":
  text("Burn", 1230, 750);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  fill(0, 255, 0);
  text("The Last Battle", 1350, 820);
  fill(0, 0, 0);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
 case"musicTheOne":
  text("Burn", 1230, 750);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  fill(0, 255, 0);
  text("The One", 1380, 855);
  fill(0, 0, 0);
  text("Valk", 1535, 750);
  break;
 case"musicValk":
  fill(0, 0, 0);
  text("Burn", 1230, 750);
```

```
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  fill(0, 255, 0);
  text("Valk", 1535, 750);
  break;
 default:
  fill(0, 255, 0);
  text("Burn", 1230, 750);
  fill(0, 0, 0);
  text("Color Panic", 1200, 785);
  text("Journey Begins", 1185, 820);
  text("New Power", 1200, 855);
  text("Retro Ride", 1370, 750);
  text("Rise", 1400, 785);
  text("The Last Battle", 1350, 820);
  text("The One", 1380, 855);
  text("Valk", 1535, 750);
  break;
 }
 fill(100, 100, 100, 100);
 stroke(255, 255, 255);
 rect(1170, 730, 165, 30);
 rect(1170, 765, 165, 30);
 rect(1170, 800, 165, 30);
 rect(1170, 835, 165, 30);
 rect(1340, 730, 165, 30);
```

```
rect(1340, 765, 165, 30);
rect(1340, 800, 165, 30);
rect(1340, 835, 165, 30);
rect(1510, 730, 100, 30);
textSize(22);



for (int i=0; i<128; i++) {
  for (int j=0; j<38; j++) {
    stroke(0);
    fill(255, 255, 255, 0);
    if (j<37 && i<125) {
      if (type[i+128*j].startsWith("emptywallLineTopRightCorner")) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="wall";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="empty";
      } else if (type[i+128*j].startsWith("emptywallNoClipLineTopRightCorner")) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="wallNoClip";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="empty";
```

```
      } else if (type[i+128*j].startsWith("wallLineTopLeftCorner") ||
type[i+128*j].startsWith("wallNoClipLineTopLeftCorner")) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="empty";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="empty";
      } else if (type[i+128*j].startsWith("emptywallLineBottomRightCorner")) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="empty";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="wall";
      } else if (type[i+128*j].startsWith("emptywallNoClipLineBottomRightCorner")) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="empty";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="wallNoClip";
      } else if (type[i+128*j].startsWith("wallLineTopRight")) {
        type[i+1+128*j]="wall";
        type[i+2+128*j]="wall";
        type[i+3+128*j]="wall";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
```

```
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="wall";
 } else if (type[i+128*j].startsWith("wallNoClipLineTopRight")) {
  type[i+1+128*j]="wallNoClip";
  type[i+2+128*j]="wallNoClip";
  type[i+3+128*j]="wallNoClip";
  type[i+128*(j+1)]="empty";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="wallNoClip";
 } else if (type[i+128*j].startsWith("wallLineTopLeft")) {
  type[i+1+128*j]="wall";
  type[i+2+128*j]="wall";
  type[i+3+128*j]="wall";
  type[i+128*(j+1)]="wall";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("wallNoClipLineTopLeft")) {
  type[i+1+128*j]="wallNoClip";
  type[i+2+128*j]="wallNoClip";
  type[i+3+128*j]="wallNoClip";
  type[i+128*(j+1)]="wallNoClip";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("wallLineTop")) {
  type[i+1+128*j]="wall";
  type[i+2+128*j]="wall";
  type[i+3+128*j]="wall";
  type[i+128*(j+1)]="empty";
```

```
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("wallNoClipLineTop")) {
  type[i+1+128*j]="wallNoClip";
  type[i+2+128*j]="wallNoClip";
  type[i+3+128*j]="wallNoClip";
  type[i+128*(j+1)]="empty";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("emptywallLineBottomLeftCorner")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wall";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("emptywallNoClipLineBottomLeftCorner")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wallNoClip";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("wallLineBottomLeft")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
```

```
  type[i+128*(j+1)]="wall";
  type[i+1+128*(j+1)]="wall";
  type[i+2+128*(j+1)]="wall";
  type[i+3+128*(j+1)]="wall";
 } else if (type[i+128*j].startsWith("wallNoClipLineBottomLeft")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wallNoClip";
  type[i+1+128*(j+1)]="wallNoClip";
  type[i+2+128*(j+1)]="wallNoClip";
  type[i+3+128*(j+1)]="wallNoClip";
 } else if (type[i+128*j].startsWith("emptywallLineBottomRight")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="wall";
  type[i+128*(j+1)]="wall";
  type[i+1+128*(j+1)]="wall";
  type[i+2+128*(j+1)]="wall";
  type[i+3+128*(j+1)]="wall";
 } else if (type[i+128*j].startsWith("emptywallNoClipLineBottomRight")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="wallNoClip";
  type[i+128*(j+1)]="wallNoClip";
  type[i+1+128*(j+1)]="wallNoClip";
  type[i+2+128*(j+1)]="wallNoClip";
  type[i+3+128*(j+1)]="wallNoClip";
 } else if (type[i+128*j].startsWith("emptywallLineBottom")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
```

```
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wall";
  type[i+1+128*(j+1)]="wall";
  type[i+2+128*(j+1)]="wall";
  type[i+3+128*(j+1)]="wall";
 } else if (type[i+128*j].startsWith("emptywallNoClipLineBottom")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wallNoClip";
  type[i+1+128*(j+1)]="wallNoClip";
  type[i+2+128*(j+1)]="wallNoClip";
  type[i+3+128*(j+1)]="wallNoClip";
 } else if (type[i+128*j].startsWith("wallLineLeft")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wall";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("wallNoClipLineLeft")) {
  type[i+1+128*j]="empty";
  type[i+2+128*j]="empty";
  type[i+3+128*j]="empty";
  type[i+128*(j+1)]="wallNoClip";
  type[i+1+128*(j+1)]="empty";
  type[i+2+128*(j+1)]="empty";
  type[i+3+128*(j+1)]="empty";
 } else if (type[i+128*j].startsWith("emptywallLineRight")) {
  type[i+1+128*j]="empty";
```

```
        type[i+2+128*j]="empty";
        type[i+3+128*j]="wall";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="wall";
      } else if (type[i+128*j].startsWith("emptywallNoClipLineRight")) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="wallNoClip";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="wallNoClip";
      } else if (i%4==0 && j%2==0 &&j<37 && type[i+128*j].startsWith("wall") &&
!type[i+128*j].startsWith("wallNoClip")) {
        type[i+1+128*j]="wall";
        type[i+2+128*j]="wall";
        type[i+3+128*j]="wall";
        type[i+128*(j+1)]="wall";
        type[i+1+128*(j+1)]="wall";
        type[i+2+128*(j+1)]="wall";
        type[i+3+128*(j+1)]="wall";
      } else if (i%4==0 && j%2==0 && j<37 && type[i+128*j].startsWith("wallNoClip")) {//&&
type[i%4+128*j].startsWith("wall") && j<37){
        type[i+1+128*j]="wallNoClip";
        type[i+2+128*j]="wallNoClip";
        type[i+3+128*j]="wallNoClip";
        type[i+128*(j+1)]="wallNoClip";
        type[i+1+128*(j+1)]="wallNoClip";
        type[i+2+128*(j+1)]="wallNoClip";
        type[i+3+128*(j+1)]="wallNoClip";
```

```
    } else if (i%4==0 && type[i+128*j].startsWith("spike") && type[i+128*j].endsWith("Top")) {
      type[i+1+128*j]="spike";
      type[i+2+128*j]="spike";
      type[i+3+128*j]="spike";
      type[i+128*(j+1)]="empty";
      type[i+1+128*(j+1)]="empty";
      type[i+2+128*(j+1)]="empty";
      type[i+3+128*(j+1)]="empty";
    } else if (i%4==0 && j%2==0 && j<37 && type[i+128*j].startsWith("emptyspike") &&
type[i+128*j].endsWith("Bottom")) {
      type[i+1+128*j]="empty";
      type[i+2+128*j]="empty";
      type[i+3+128*j]="empty";
      type[i+128*(j+1)]="spike";
      type[i+1+128*(j+1)]="spike";
      type[i+2+128*(j+1)]="spike";
      type[i+3+128*(j+1)]="spike";
    } else if (j%2==0 && j<37 && type[i+128*j].startsWith("spike") &&
type[i+128*j].endsWith("Left")) {
      type[i+1+128*j]="spike";
      type[i+2+128*j]="empty";
      type[i+3+128*j]="empty";
      type[i+128*(j+1)]="spike";
      type[i+1+128*(j+1)]="spike";
      type[i+2+128*(j+1)]="empty";
      type[i+3+128*(j+1)]="empty";
    } else if (j%2==0 && j<37 && type[i+128*j].startsWith("emptyspike") &&
type[i+128*j].endsWith("Right")) {
      type[i+1+128*j]="empty";
      type[i+2+128*j]="spike";
      type[i+3+128*j]="spike";
      type[i+128*(j+1)]="empty";
```

```
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="spike";
        type[i+3+128*(j+1)]="spike";
      } else if (i%4==0 && j%2==0 && j<37 && i<125 && type[i+128*j].equals("empty") &&
(type[i+1+128*j].startsWith("wall") || type[i+1+128*j].startsWith("emptywall") ||
type[i+1+128*j].startsWith("spike") ||type[i+1+128*j].startsWith("emptyspike")
||type[i+2+128*j].startsWith("wall") || type[i+2+128*j].startsWith("emptywall") ||
type[i+2+128*j].startsWith("spike") ||type[i+2+128*j].startsWith("emptyspike")
||type[i+3+128*j].startsWith("wall") || type[i+3+128*j].startsWith("emptywall")||
type[i+3+128*j].startsWith("spike") ||type[i+3+128*j].startsWith("emptyspike")
||type[i+128*(j+1)].startsWith("wall") ||type[i+128*(j+1)].startsWith("emptywall") ||
type[i+128*(j+1)].startsWith("spike") ||type[i+128*(j+1)].startsWith("emptyspike")
||type[i+1+128*(j+1)].startsWith("wall") ||type[i+1+128*(j+1)].startsWith("emptywall") ||
type[i+1+128*(j+1)].startsWith("spike") ||type[i+1+128*(j+1)].startsWith("emptyspike")
||type[i+2+128*(j+1)].startsWith("wall") || type[i+2+128*(j+1)].startsWith("emptywall") ||
type[i+2+128*(j+1)].startsWith("spike") ||type[i+2+128*(j+1)].startsWith("emptyspike") ||
type[i+3+128*(j+1)].startsWith("wall") ||type[i+3+128*(j+1)].startsWith("emptywall") ||
type[i+3+128*(j+1)].startsWith("spike") ||type[i+3+128*(j+1)].startsWith("emptyspike"))) {
        type[i+1+128*j]="empty";
        type[i+2+128*j]="empty";
        type[i+3+128*j]="empty";
        type[i+128*(j+1)]="empty";
        type[i+1+128*(j+1)]="empty";
        type[i+2+128*(j+1)]="empty";
        type[i+3+128*(j+1)]="empty";
      } else if (i%8==0 && j%4==0 && j<35 && i<121 && type[i+128*j].startsWith("door")) {
        for (int k=1; k<8; k++) {
          for (int l=0; l<4; l++) {
            type[i+k+128*(j+l)]="end"+type[i+128*j].substring(4, type[i+128*j].length());
          }
        }
        type[i+128*(j+1)]="end"+type[i+128*j].substring(4, type[i+128*j].length());
        type[i+128*(j+2)]="end"+type[i+128*j].substring(4, type[i+128*j].length());
        type[i+128*(j+3)]="end"+type[i+128*j].substring(4, type[i+128*j].length());
      }
    }
```

```
    if (x>50+8*i && x<50+(8*i+8) && y>50+16*j && y<50+(16*j+16) && x<1074 && y<=650) {
      type[i+128*j]=types;
    }
    if (type[i+128*j].equals("null") || type[i+128*j]=="null") {
      type[i+128*j]="empty";
    }

    ////////////////////////////////////////////////////////////////////////
    if (type[i+128*j].equals("emptyspikeDarkBlueBottom") ||
type[i+128*j]=="emptyspikeDarkBlueBottom") {
      img=spike[0];
    } else if (type[i+128*j].equals("spikeDarkBlueLeft") || type[i+128*j]=="spikeDarkBlueLeft") {
      img=spike[1];
    } else if (type[i+128*j].equals("spikeDarkBlueTop") || type[i+128*j]=="spikeDarkBlueTop") {
      img=spike[2];
    } else if (type[i+128*j].equals("emptyspikeDarkBlueRight") ||
type[i+128*j]=="emptyspikeDarkBlueRight") {
      img=spike[3];
    } else if (type[i+128*j].equals("emptyspikeGreenBottom") ||
type[i+128*j]=="emptyspikeGreenBottom") {
      img=spike[4];
    } else if (type[i+128*j].equals("spikeGreenLeft") || type[i+128*j]=="spikeGreenLeft") {
      img=spike[5];
    } else if (type[i+128*j].equals("spikeGreenTop") || type[i+128*j]=="spikeGreenTop") {
      img=spike[6];
    } else if (type[i+128*j].equals("emptyspikeGreenRight")
||type[i+128*j]=="emptyspikeGreenRight") {
      img=spike[7];
    } else if (type[i+128*j].equals("emptyspikeLightBlueBottom")
||type[i+128*j]=="emptyspikeLightBlueBottom") {
      img=spike[8];
    } else if (type[i+128*j].equals("spikeLightBlueLeft") || type[i+128*j]=="spikeLightBlueLeft") {
      img=spike[9];
```

```
    } else if (type[i+128*j].equals("spikeLightBlueTop") || type[i+128*j]=="spikeLightBlueTop") {

      img=spike[10];

    } else if (type[i+128*j].equals("emptyspikeLightBlueRight") ||
type[i+128*j]=="emptyspikeLightBlueRight") {

      img=spike[11];

    } else if (type[i+128*j].equals("emptyspikePurpleBottom") ||
type[i+128*j]=="emptyspikePurpleBottom") {

      img=spike[12];

    } else if (type[i+128*j].equals("spikePurpleLeft") || type[i+128*j]=="spikePurpleLeft") {

      img=spike[13];

    } else if (type[i+128*j].equals("spikePurpleTop") || type[i+128*j]=="spikePurpleTop") {

      img=spike[14];

    } else if (type[i+128*j].equals("emptyspikePurpleRight") ||
type[i+128*j]=="emptyspikePurpleRight") {

      img=spike[15];

    } else if (type[i+128*j].equals("emptyspikeRedBottom") ||
type[i+128*j]=="emptyspikeRedBottom") {

      img=spike[16];

    } else if (type[i+128*j].equals("spikeRedLeft") || type[i+128*j]=="spikeRedLeft") {

      img=spike[17];

    } else if (type[i+128*j].equals("spikeRedTop") || type[i+128*j]=="spikeRedTop") {

      img=spike[18];

    } else if (type[i+128*j].equals("emptyspikeRedRight") || type[i+128*j]=="emptyspikeRedRight") {

      img=spike[19];

      ////////////////////////////////////////////////////////////////////////////////

    } else if (type[i+128*j].equals("wallBottomPurple") || type[i+128*j]=="wallBottomPurple") {

      img=wallPurple[0];

    } else if (type[i+128*j].equals("wallBottomLeftPurple") ||
type[i+128*j]=="wallBottomLeftPurple") {

      img=wallPurple[1];

    } else if (type[i+128*j].equals("wallBottomLeftCornerPurple") ||
type[i+128*j]=="wallBottomLeftCornerPurple") {

      img=wallPurple[2];
```

```
    } else if (type[i+128*j].equals("wallBottomRightPurple") ||
type[i+128*j]=="wallBottomRightPurple") {

      img=wallPurple[3];

    } else if (type[i+128*j].equals("wallBottomRightCornerPurple") ||
type[i+128*j]=="wallBottomRightCornerPurple") {

      img=wallPurple[4];

    } else if (type[i+128*j].equals("wallCenterPurple") || type[i+128*j]=="wallCenterPurple") {

      img=wallPurple[5];

    } else if (type[i+128*j].equals("wallLeftPurple") || type[i+128*j]=="wallLeftPurple") {

      img=wallPurple[6];

    } else if (type[i+128*j].equals("emptywallLineBottomPurple") ||
type[i+128*j]=="emptywallLineBottomPurple") {

      img=wallPurple[7];

    } else if (type[i+128*j].equals("wallLineBottomLeftPurple") ||
type[i+128*j]=="wallLineBottomLeftPurple") {

      img=wallPurple[8];

    } else if (type[i+128*j].equals("emptywallLineBottomLeftCornerPurple") ||
type[i+128*j]=="emptywallLineBottomLeftCornerPurple") {

      img=wallPurple[9];

    } else if (type[i+128*j].equals("emptywallLineBottomRightPurple")
||type[i+128*j]=="emptywallLineBottomRightPurple") {

      img=wallPurple[10];

    } else if (type[i+128*j].equals("emptywallLineBottomRightCornerPurple") ||
type[i+128*j]=="emptywallLineBottomRightCornerPurple") {

      img=wallPurple[11];

    } else if (type[i+128*j].equals("wallLineLeftPurple") || type[i+128*j]=="wallLineLeftPurple") {

      img=wallPurple[12];

    } else if (type[i+128*j].equals("emptywallLineRightPurple") ||
type[i+128*j]=="emptywallLineRightPurple") {

      img=wallPurple[14];

    } else if (type[i+128*j].equals("wallLineTopPurple") || type[i+128*j]=="wallLineTopPurple") {

      img=wallPurple[15];

    } else if (type[i+128*j].equals("wallLineTopLeftPurple") ||
type[i+128*j]=="wallLineTopLeftPurple") {
```

```
        img=wallPurple[16];

    } else if (type[i+128*j].equals("wallLineTopLeftCornerPurple") ||
type[i+128*j]=="wallLineTopLeftCornerPurple") {

        img=wallPurple[17];

    } else if (type[i+128*j].equals("wallLineTopRightPurple") ||
type[i+128*j]=="wallLineTopRightPurple") {

        img=wallPurple[18];

    } else if (type[i+128*j].equals("emptywallLineTopRightCornerPurple") ||
type[i+128*j]=="emptywallLineTopRightCornerPurple") {

        img=wallPurple[13];

    } else if (type[i+128*j].equals("wallRightPurple") ||type[i+128*j]=="wallRightPurple") {

        img=wallPurple[20];

    } else if (type[i+128*j].equals("wallTopPurple") ||type[i+128*j]=="wallTopPurple") {

        img=wallPurple[21];

    } else if (type[i+128*j].equals("wallTopLeftPurple") || type[i+128*j]=="wallTopLeftPurple") {

        img=wallPurple[22];

    } else if (type[i+128*j].equals("wallTopLeftCornerPurple") ||
type[i+128*j]=="wallTopLeftCornerPurple") {

        img=wallPurple[23];

    } else if (type[i+128*j].equals("wallTopRightPurple") || type[i+128*j]=="wallTopRightPurple") {

        img=wallPurple[24];

    } else if (type[i+128*j].equals("wallTopRightCornerPurple") ||
type[i+128*j]=="wallTopRightCornerPurple") {

        img=wallPurple[25];

    } else if (type[i+128*j].equals("wallChiseledPurple") || type[i+128*j]=="wallCiseledPurple") {

        img=wallPurple[19];

    } else if (type[i+128*j].equals("wallBottomRed") || type[i+128*j]=="wallBottomRed") {

        img=wallRed[0];

    } else if (type[i+128*j].equals("wallBottomLeftRed") || type[i+128*j]=="wallBottomLeftRed") {

        img=wallRed[1];

    } else if (type[i+128*j].equals("wallBottomLeftCornerRed") ||
type[i+128*j]=="wallBottomLeftCornerRed") {

        img=wallRed[2];
```

```
    } else if (type[i+128*j].equals("wallBottomRightRed") || type[i+128*j]=="wallBottomRightRed") {

      img=wallRed[3];

    } else if (type[i+128*j].equals("wallBottomRightCornerRed") ||
type[i+128*j]=="wallBottomRightCornerRed") {

      img=wallRed[4];

    } else if (type[i+128*j].equals("wallCenterRed") || type[i+128*j]=="wallCenterRed") {

      img=wallRed[5];

    } else if (type[i+128*j].equals("wallLeftRed") || type[i+128*j]=="wallLeftRed") {

      img=wallRed[6];

    } else if (type[i+128*j].equals("emptywallLineBottomRed") ||
type[i+128*j]=="emptywallLineBottomRed") {

      img=wallRed[7];

    } else if (type[i+128*j].equals("wallLineBottomLeftRed") ||
type[i+128*j]=="wallLineBottomLeftRed") {

      img=wallRed[8];

    } else if (type[i+128*j].equals("emptywallLineBottomLeftCornerRed") ||
type[i+128*j]=="emptywallLineBottomLeftCornerRed") {

      img=wallRed[9];

    } else if (type[i+128*j].equals("emptywallLineBottomRightRed")
||type[i+128*j]=="emptywallLineBottomRightRed") {

      img=wallRed[10];

    } else if (type[i+128*j].equals("emptywallLineBottomRightCornerRed") ||
type[i+128*j]=="emptywallLineBottomRightCornerRed") {

      img=wallRed[11];

    } else if (type[i+128*j].equals("wallLineLeftRed") || type[i+128*j]=="wallLineLeftRed") {

      img=wallRed[12];

    } else if (type[i+128*j].equals("emptywallLineRightRed") ||
type[i+128*j]=="emptywallLineRightRed") {

      img=wallRed[14];

    } else if (type[i+128*j].equals("wallLineTopRed") || type[i+128*j]=="wallLineTopRed") {

      img=wallRed[15];

    } else if (type[i+128*j].equals("wallLineTopLeftRed") || type[i+128*j]=="wallLineTopLeftRed") {

      img=wallRed[16];
```

```
    } else if (type[i+128*j].equals("wallLineTopLeftCornerRed") ||
type[i+128*j]=="wallLineTopLeftCornerRed") {

      img=wallRed[17];

    } else if (type[i+128*j].equals("wallLineTopRightRed") || type[i+128*j]=="wallLineTopRightRed") {

      img=wallRed[18];

    } else if (type[i+128*j].equals("emptywallLineTopRightCornerRed") ||
type[i+128*j]=="emptywallLineTopRightCornerRed") {

      img=wallRed[13];

    } else if (type[i+128*j].equals("wallRightRed") ||type[i+128*j]=="wallRightRed") {

      img=wallRed[20];

    } else if (type[i+128*j].equals("wallTopRed") ||type[i+128*j]=="wallTopRed") {

      img=wallRed[21];

    } else if (type[i+128*j].equals("wallTopLeftRed") || type[i+128*j]=="wallTopLeftRed") {

      img=wallRed[22];

    } else if (type[i+128*j].equals("wallTopLeftCornerRed") ||
type[i+128*j]=="wallTopLeftCornerRed") {

      img=wallRed[23];

    } else if (type[i+128*j].equals("wallTopRightRed") || type[i+128*j]=="wallTopRightRed") {

      img=wallRed[24];

    } else if (type[i+128*j].equals("wallTopRightCornerRed") ||
type[i+128*j]=="wallTopRightCornerRed") {

      img=wallRed[25];

    } else if (type[i+128*j].equals("wallChiseledRed") ||type[i+128*j]=="wallChiseledRed") {

      img=wallRed[19];

    } else if (type[i+128*j].equals("wallBottomDarkBlue") || type[i+128*j]=="wallBottomDarkBlue") {

      img=wallDarkBlue[0];

    } else if (type[i+128*j].equals("wallBottomLeftDarkBlue") ||
type[i+128*j]=="wallBottomLeftDarkBlue") {

      img=wallDarkBlue[1];

    } else if (type[i+128*j].equals("wallBottomLeftCornerDarkBlue") ||
type[i+128*j]=="wallBottomLeftCornerDarkBlue") {

      img=wallDarkBlue[2];

    } else if (type[i+128*j].equals("wallBottomRightDarkBlue") ||
type[i+128*j]=="wallBottomRightDarkBlue") {
```

```
        img=wallDarkBlue[3];

    } else if (type[i+128*j].equals("wallBottomRightCornerDarkBlue") ||
type[i+128*j]=="wallBottomRightCornerDarkBlue") {

        img=wallDarkBlue[4];

    } else if (type[i+128*j].equals("wallCenterDarkBlue") || type[i+128*j]=="wallCenterDarkBlue") {

        img=wallDarkBlue[5];

    } else if (type[i+128*j].equals("wallLeftDarkBlue") || type[i+128*j]=="wallLeftDarkBlue") {

        img=wallDarkBlue[6];

    } else if (type[i+128*j].equals("emptywallLineBottomDarkBlue") ||
type[i+128*j]=="emptywallLineBottomDarkBlue") {

        img=wallDarkBlue[7];

    } else if (type[i+128*j].equals("wallLineBottomLeftDarkBlue") ||
type[i+128*j]=="wallLineBottomLeftDarkBlue") {

        img=wallDarkBlue[8];

    } else if (type[i+128*j].equals("emptywallLineBottomLeftCornerDarkBlue") ||
type[i+128*j]=="emptywallLineBottomLeftCornerDarkBlue") {

        img=wallDarkBlue[9];

    } else if (type[i+128*j].equals("emptywallLineBottomRightDarkBlue")
||type[i+128*j]=="emptywallLineBottomRightDarkBlue") {

        img=wallDarkBlue[10];

    } else if (type[i+128*j].equals("emptywallLineBottomRightCornerDarkBlue") ||
type[i+128*j]=="emptywallLineBottomRightCornerDarkBlue") {

        img=wallDarkBlue[11];

    } else if (type[i+128*j].equals("wallLineLeftDarkBlue") || type[i+128*j]=="wallLineLeftDarkBlue") {

        img=wallDarkBlue[12];

    } else if (type[i+128*j].equals("emptywallLineRightDarkBlue") ||
type[i+128*j]=="emptywallLineRightDarkBlue") {

        img=wallDarkBlue[14];

    } else if (type[i+128*j].equals("wallLineTopDarkBlue") || type[i+128*j]=="wallLineTopDarkBlue") {

        img=wallDarkBlue[15];

    } else if (type[i+128*j].equals("wallLineTopLeftDarkBlue") ||
type[i+128*j]=="wallLineTopLeftDarkBlue") {

        img=wallDarkBlue[16];
```

```
    } else if (type[i+128*j].equals("wallLineTopLeftCornerDarkBlue") ||
type[i+128*j]=="wallLineTopLeftCornerDarkBlue") {

      img=wallDarkBlue[17];

    } else if (type[i+128*j].equals("wallLineTopRightDarkBlue") ||
type[i+128*j]=="wallLineTopRightDarkBlue") {

      img=wallDarkBlue[18];

    } else if (type[i+128*j].equals("emptywallLineTopRightCornerDarkBlue") ||
type[i+128*j]=="emptywallDLineTopRightCornerarkBlue") {

      img=wallDarkBlue[13];

    } else if (type[i+128*j].equals("wallRightDarkBlue") ||type[i+128*j]=="wallRightDarkBlue") {

      img=wallDarkBlue[20];

    } else if (type[i+128*j].equals("wallTopDarkBlue") ||type[i+128*j]=="wallTopDarkBlue") {

      img=wallDarkBlue[21];

    } else if (type[i+128*j].equals("wallTopLeftDarkBlue") || type[i+128*j]=="wallTopLeftDarkBlue") {

      img=wallDarkBlue[22];

    } else if (type[i+128*j].equals("wallTopLeftCornerDarkBlue") ||
type[i+128*j]=="wallTopLeftCornerDarkBlue") {

      img=wallDarkBlue[23];

    } else if (type[i+128*j].equals("wallTopRightDarkBlue") ||
type[i+128*j]=="wallTopRightDarkBlue") {

      img=wallDarkBlue[24];

    } else if (type[i+128*j].equals("wallTopRightCornerDarkBlue") ||
type[i+128*j]=="wallTopRightCornerDarkBlue") {

      img=wallDarkBlue[25];

    } else if (type[i+128*j].equals("wallChiseledDarkBlue") ||type[i+128*j]=="wallChiseledDarkBlue")
{

      img=wallDarkBlue[19];

    } else if (type[i+128*j].equals("wallBottomLightBlue") || type[i+128*j]=="wallBottomLightBlue") {

      img=wallLightBlue[0];

    } else if (type[i+128*j].equals("wallBottomLeftLightBlue") ||
type[i+128*j]=="wallBottomLeftLightBlue") {

      img=wallLightBlue[1];

    } else if (type[i+128*j].equals("wallBottomLeftCornerLightBlue") ||
type[i+128*j]=="wallBottomLeftCornerLightBlue") {
```

```
        img=wallLightBlue[2];

    } else if (type[i+128*j].equals("wallBottomRightLightBlue") ||
type[i+128*j]=="wallBottomRightLightBlue") {

        img=wallLightBlue[3];

    } else if (type[i+128*j].equals("wallBottomRightCornerLightBlue") ||
type[i+128*j]=="wallBottomRightCornerLightBlue") {

        img=wallLightBlue[4];

    } else if (type[i+128*j].equals("wallCenterLightBlue") || type[i+128*j]=="wallCenterLightBlue") {

        img=wallLightBlue[5];

    } else if (type[i+128*j].equals("wallLeftLightBlue") || type[i+128*j]=="wallLeftLightBlue") {

        img=wallLightBlue[6];

    } else if (type[i+128*j].equals("emptywallLineBottomLightBlue") ||
type[i+128*j]=="emptywallLineBottomLightBlue") {

        img=wallLightBlue[7];

    } else if (type[i+128*j].equals("wallLineBottomLeftLightBlue") ||
type[i+128*j]=="wallLineBottomLeftLightBlue") {

        img=wallLightBlue[8];

    } else if (type[i+128*j].equals("emptywallLineBottomLeftCornerLightBlue") ||
type[i+128*j]=="emptywallLineBottomLeftCornerLightBlue") {

        img=wallLightBlue[9];

    } else if (type[i+128*j].equals("emptywallLineBottomRightLightBlue")
||type[i+128*j]=="emptywallLineBottomRightLightBlue") {

        img=wallLightBlue[10];

    } else if (type[i+128*j].equals("emptywallLineBottomRightCornerLightBlue") ||
type[i+128*j]=="emptywallLineBottomRightCornerLightBlue") {

        img=wallLightBlue[11];

    } else if (type[i+128*j].equals("wallLineLeftLightBlue") || type[i+128*j]=="wallLineLeftLightBlue")
{

        img=wallLightBlue[12];

    } else if (type[i+128*j].equals("emptywallLineRightLightBlue") ||
type[i+128*j]=="emptywallLineRightLightBlue") {

        img=wallLightBlue[14];

    } else if (type[i+128*j].equals("wallLineTopLightBlue") || type[i+128*j]=="wallLineTopLightBlue") {

        img=wallLightBlue[15];
```

```
    } else if (type[i+128*j].equals("wallLineTopLeftLightBlue") ||
type[i+128*j]=="wallLineTopLeftLightBlue") {

      img=wallLightBlue[16];

    } else if (type[i+128*j].equals("wallLineTopLeftCornerLightBlue") ||
type[i+128*j]=="wallLineTopLeftCornerLightBlue") {

      img=wallLightBlue[17];

    } else if (type[i+128*j].equals("wallLineTopRightLightBlue") ||
type[i+128*j]=="wallLineTopRightLightBlue") {

      img=wallLightBlue[18];

    } else if (type[i+128*j].equals("emptywallLineTopRightCornerLightBlue") ||
type[i+128*j]=="emptywallLineTopRightCornerLightBlue") {

      img=wallLightBlue[13];

    } else if (type[i+128*j].equals("wallRightLightBlue") ||type[i+128*j]=="wallRightLightBlue") {

      img=wallLightBlue[20];

    } else if (type[i+128*j].equals("wallTopLightBlue") ||type[i+128*j]=="wallTopLightBlue") {

      img=wallLightBlue[21];

    } else if (type[i+128*j].equals("wallTopLeftLightBlue") || type[i+128*j]=="wallTopLeftLightBlue") {

      img=wallLightBlue[22];

    } else if (type[i+128*j].equals("wallTopLeftCornerLightBlue") ||
type[i+128*j]=="wallTopLeftCornerLightBlue") {

      img=wallLightBlue[23];

    } else if (type[i+128*j].equals("wallTopRightLightBlue") ||
type[i+128*j]=="wallTopRightLightBlue") {

      img=wallLightBlue[24];

    } else if (type[i+128*j].equals("wallTopRightCornerLightBlue") ||
type[i+128*j]=="wallTopRightCornerLightBlue") {

      img=wallLightBlue[25];

    } else if (type[i+128*j].equals("wallChiseledLightBlue") ||type[i+128*j]=="wallChiseledLightBlue")
{

      img=wallLightBlue[19];

    } else if (type[i+128*j].equals("wallBottomGreen") || type[i+128*j]=="wallBottomGreen") {

      img=wallGreen[0];

    } else if (type[i+128*j].equals("wallBottomLeftGreen") || type[i+128*j]=="wallBottomLeftGreen")
{
```

```
        img=wallGreen[1];

    } else if (type[i+128*j].equals("wallBottomLeftCornerGreen") ||
type[i+128*j]=="wallBottomLeftCornerGreen") {

        img=wallGreen[2];

    } else if (type[i+128*j].equals("wallBottomRightGreen") ||
type[i+128*j]=="wallBottomRightGreen") {

        img=wallGreen[3];

    } else if (type[i+128*j].equals("wallBottomRightCornerGreen") ||
type[i+128*j]=="wallBottomRightCornerGreen") {

        img=wallGreen[4];

    } else if (type[i+128*j].equals("wallCenterGreen") || type[i+128*j]=="wallCenterGreen") {

        img=wallGreen[5];

    } else if (type[i+128*j].equals("wallLeftGreen") || type[i+128*j]=="wallLeftGreen") {

        img=wallGreen[6];

    } else if (type[i+128*j].equals("emptywallLineBottomGreen") ||
type[i+128*j]=="emptywallLineBottomGreen") {

        img=wallGreen[7];

    } else if (type[i+128*j].equals("wallLineBottomLeftGreen") ||
type[i+128*j]=="wallLineBottomLeftGreen") {

        img=wallGreen[8];

    } else if (type[i+128*j].equals("emptywallLineBottomLeftCornerGreen") ||
type[i+128*j]=="emptywallLineBottomLeftCornerGreen") {

        img=wallGreen[9];

    } else if (type[i+128*j].equals("emptywallLineBottomRightGreen")
||type[i+128*j]=="emptywallLineBottomRightGreen") {

        img=wallGreen[10];

    } else if (type[i+128*j].equals("emptywallLineBottomRightCornerGreen") ||
type[i+128*j]=="emptywallLineBottomRightCornerGreen") {

        img=wallGreen[11];

    } else if (type[i+128*j].equals("wallLineLeftGreen") || type[i+128*j]=="wallLineLeftGreen") {

        img=wallGreen[12];

    } else if (type[i+128*j].equals("emptywallLineRightGreen") ||
type[i+128*j]=="emptywallLineRightGreen") {

        img=wallGreen[14];
```

```
    } else if (type[i+128*j].equals("wallLineTopGreen") || type[i+128*j]=="wallLineTopGreen") {

      img=wallGreen[15];

    } else if (type[i+128*j].equals("wallLineTopLeftGreen") ||
type[i+128*j]=="wallLineTopLeftGreen") {

      img=wallGreen[16];

    } else if (type[i+128*j].equals("wallLineTopLeftCornerGreen") ||
type[i+128*j]=="wallLineTopLeftCornerGreen") {

      img=wallGreen[17];

    } else if (type[i+128*j].equals("wallLineTopRightGreen") ||
type[i+128*j]=="wallLineTopRightGreen") {

      img=wallGreen[18];

    } else if (type[i+128*j].equals("emptywallLineTopRightCornerGreen") ||
type[i+128*j]=="emptywallLineTopRightCornerGreen") {

      img=wallGreen[13];

    } else if (type[i+128*j].equals("wallRightGreen") ||type[i+128*j]=="wallRightGreen") {

      img=wallGreen[20];

    } else if (type[i+128*j].equals("wallTopGreen") ||type[i+128*j]=="wallTopGreen") {

      img=wallGreen[21];

    } else if (type[i+128*j].equals("wallTopLeftGreen") || type[i+128*j]=="wallTopLeftGreen") {

      img=wallGreen[22];

    } else if (type[i+128*j].equals("wallTopLeftCornerGreen") ||
type[i+128*j]=="wallTopLeftCornerGreen") {

      img=wallGreen[23];

    } else if (type[i+128*j].equals("wallTopRightGreen") || type[i+128*j]=="wallTopRightGreen") {

      img=wallGreen[24];

    } else if (type[i+128*j].equals("wallTopRightCornerGreen") ||
type[i+128*j]=="wallTopRightCornerGreen") {

      img=wallGreen[25];

    } else if (type[i+128*j].equals("wallChiseledGreen") ||type[i+128*j]=="wallChiseledGreen") {

      img=wallGreen[19];

    } else if (type[i+128*j].equals("wallNoClipBottomPurple") ||
type[i+128*j]=="wallNoClipBottomPurple") {

      img=wallPurple[0];
```

```
    } else if (type[i+128*j].equals("wallNoClipBottomLeftPurple") ||
type[i+128*j]=="wallNoClipBottomLeftPurple") {

      img=wallPurple[1];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftCornerPurple") ||
type[i+128*j]=="wallNoClipBottomLeftCornerPurple") {

      img=wallPurple[2];

    } else if (type[i+128*j].equals("wallNoClipBottomRightPurple") ||
type[i+128*j]=="wallNoClipBottomRightPurple") {

      img=wallPurple[3];

    } else if (type[i+128*j].equals("wallNoClipBottomRightCornerPurple") ||
type[i+128*j]=="wallNoClipBottomRightCornerPurple") {

      img=wallPurple[4];

    } else if (type[i+128*j].equals("wallNoClipCenterPurple") ||
type[i+128*j]=="wallNoClipCenterPurple") {

      img=wallPurple[5];

    } else if (type[i+128*j].equals("wallNoClipLeftPurple") || type[i+128*j]=="wallNoClipLeftPurple") {

      img=wallPurple[6];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomPurple") ||
type[i+128*j]=="emptywallNoClipLineBottomPurple") {

      img=wallPurple[7];

    } else if (type[i+128*j].equals("wallNoClipLineBottomLeftPurple") ||
type[i+128*j]=="wallNoClipLineBottomLeftPurple") {

      img=wallPurple[8];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomLeftCornerPurple") ||
type[i+128*j]=="emptywallNoClipLineBottomLeftCornerPurple") {

      img=wallPurple[9];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightPurple")
||type[i+128*j]=="emptywallNoClipLineBottomRightPurple") {

      img=wallPurple[10];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightCornerPurple") ||
type[i+128*j]=="emptywallNoClipLineBottomRightCornerPurple") {

      img=wallPurple[11];

    } else if (type[i+128*j].equals("wallNoClipLineLeftPurple") ||
type[i+128*j]=="wallNoClipLineLeftPurple") {

      img=wallPurple[12];
```

```
    } else if (type[i+128*j].equals("emptywallNoClipLineRightPurple") ||
type[i+128*j]=="emptywallNoClipLineRightPurple") {

      img=wallPurple[14];

    } else if (type[i+128*j].equals("wallNoClipLineTopPurple") ||
type[i+128*j]=="wallNoClipLineTopPurple") {

      img=wallPurple[15];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftPurple") ||
type[i+128*j]=="wallNoClipLineTopLeftPurple") {

      img=wallPurple[16];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftCornerPurple") ||
type[i+128*j]=="wallNoClipLineTopLeftCornerPurple") {

      img=wallPurple[17];

    } else if (type[i+128*j].equals("wallNoClipLineTopRightPurple") ||
type[i+128*j]=="wallNoClipLineTopRightPurple") {

      img=wallPurple[18];

    } else if (type[i+128*j].equals("emptywallNoClipLineTopRightCornerPurple") ||
type[i+128*j]=="emptywallNoClipLineTopRightCornerPurple") {

      img=wallPurple[13];

    } else if (type[i+128*j].equals("wallNoClipRightPurple")
||type[i+128*j]=="wallNoClipRightPurple") {

      img=wallPurple[20];

    } else if (type[i+128*j].equals("wallNoClipTopPurple") ||type[i+128*j]=="wallNoClipTopPurple") {

      img=wallPurple[21];

    } else if (type[i+128*j].equals("wallNoClipTopLeftPurple") ||
type[i+128*j]=="wallNoClipTopLeftPurple") {

      img=wallPurple[22];

    } else if (type[i+128*j].equals("wallNoClipTopLeftCornerPurple") ||
type[i+128*j]=="wallNoClipTopLeftCornerPurple") {

      img=wallPurple[23];

    } else if (type[i+128*j].equals("wallNoClipTopRightPurple") ||
type[i+128*j]=="wallNoClipTopRightPurple") {

      img=wallPurple[24];

    } else if (type[i+128*j].equals("wallNoClipTopRightCornerPurple") ||
type[i+128*j]=="wallNoClipTopRightCornerPurple") {

      img=wallPurple[25];
```

```
    } else if (type[i+128*j].equals("wallNoClipChiseledPurple") ||
type[i+128*j]=="wallNoClipChiseledPurple") {

      img=wallPurple[19];

    } else if (type[i+128*j].equals("wallNoClipBottomRed") ||
type[i+128*j]=="wallNoClipBottomRed") {

      img=wallRed[0];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftRed") ||
type[i+128*j]=="wallNoClipBottomLeftRed") {

      img=wallRed[1];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftCornerRed") ||
type[i+128*j]=="wallNoClipBottomLeftCornerRed") {

      img=wallRed[2];

    } else if (type[i+128*j].equals("wallNoClipBottomRightRed") ||
type[i+128*j]=="wallNoClipBottomRightRed") {

      img=wallRed[3];

    } else if (type[i+128*j].equals("wallNoClipBottomRightCornerRed") ||
type[i+128*j]=="wallNoClipBottomRightCornerRed") {

      img=wallRed[4];

    } else if (type[i+128*j].equals("wallNoClipCenterRed") || type[i+128*j]=="wallNoClipCenterRed")
{

      img=wallRed[5];

    } else if (type[i+128*j].equals("wallNoClipLeftRed") || type[i+128*j]=="wallNoClipLeftRed") {

      img=wallRed[6];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRed") ||
type[i+128*j]=="emptywallNoClipLineBottomRed") {

      img=wallRed[7];

    } else if (type[i+128*j].equals("wallNoClipLineBottomLeftRed") ||
type[i+128*j]=="wallNoClipLineBottomLeftRed") {

      img=wallRed[8];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomLeftCornerRed") ||
type[i+128*j]=="emptywallNoClipLineBottomLeftCornerRed") {

      img=wallRed[9];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightRed")
||type[i+128*j]=="emptywallNoClipLineBottomRightRed") {

      img=wallRed[10];
```

```
    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightCornerRed") ||
type[i+128*j]=="emptywallNoClipLineBottomRightCornerRed") {

      img=wallRed[11];

    } else if (type[i+128*j].equals("wallNoClipLineLeftRed") ||
type[i+128*j]=="wallNoClipLineLeftRed") {

      img=wallRed[12];

    } else if (type[i+128*j].equals("emptywallNoClipLineRightRed") ||
type[i+128*j]=="emptywallNoClipLineRightRed") {

      img=wallRed[14];

    } else if (type[i+128*j].equals("wallNoClipLineTopRed") ||
type[i+128*j]=="wallNoClipLineTopRed") {

      img=wallRed[15];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftRed") ||
type[i+128*j]=="wallNoClipLineTopLeftRed") {

      img=wallRed[16];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftCornerRed") ||
type[i+128*j]=="wallNoClipLineTopLeftCornerRed") {

      img=wallRed[17];

    } else if (type[i+128*j].equals("wallNoClipLineTopRightRed") ||
type[i+128*j]=="wallNoClipLineTopRightRed") {

      img=wallRed[18];

    } else if (type[i+128*j].equals("emptywallNoClipLineTopRightCornerRed") ||
type[i+128*j]=="emptywallNoClipLineTopRightCornerRed") {

      img=wallRed[13];

    } else if (type[i+128*j].equals("wallNoClipRightRed") ||type[i+128*j]=="wallNoClipRightRed") {

      img=wallRed[20];

    } else if (type[i+128*j].equals("wallNoClipTopRed") ||type[i+128*j]=="wallNoClipTopRed") {

      img=wallRed[21];

    } else if (type[i+128*j].equals("wallNoClipTopLeftRed") ||
type[i+128*j]=="wallNoClipTopLeftRed") {

      img=wallRed[22];

    } else if (type[i+128*j].equals("wallNoClipTopLeftCornerRed") ||
type[i+128*j]=="wallNoClipTopLeftCornerRed") {

      img=wallRed[23];
```

```
    } else if (type[i+128*j].equals("wallNoClipTopRightRed") ||
type[i+128*j]=="wallNoClipTopRightRed") {

      img=wallRed[24];

    } else if (type[i+128*j].equals("wallNoClipTopRightCornerRed") ||
type[i+128*j]=="wallNoClipTopRightCornerRed") {

      img=wallRed[25];

    } else if (type[i+128*j].equals("wallNoClipChiseledRed")
||type[i+128*j]=="wallNoClipChiseledRed") {

      img=wallRed[19];

    } else if (type[i+128*j].equals("wallNoClipBottomDarkBlue") ||
type[i+128*j]=="wallNoClipBottomDarkBlue") {

      img=wallDarkBlue[0];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftDarkBlue") ||
type[i+128*j]=="wallNoClipBottomLeftDarkBlue") {

      img=wallDarkBlue[1];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftCornerDarkBlue") ||
type[i+128*j]=="wallNoClipBottomLeftCornerDarkBlue") {

      img=wallDarkBlue[2];

    } else if (type[i+128*j].equals("wallNoClipBottomRightDarkBlue") ||
type[i+128*j]=="wallNoClipBottomRightDarkBlue") {

      img=wallDarkBlue[3];

    } else if (type[i+128*j].equals("wallNoClipBottomRightCornerDarkBlue") ||
type[i+128*j]=="wallBottomRightCornerDarkBlue") {

      img=wallDarkBlue[4];

    } else if (type[i+128*j].equals("wallNoClipCenterDarkBlue") ||
type[i+128*j]=="wallNoClipCenterDarkBlue") {

      img=wallDarkBlue[5];

    } else if (type[i+128*j].equals("wallNoClipLeftDarkBlue") ||
type[i+128*j]=="wallNoClipLeftDarkBlue") {

      img=wallDarkBlue[6];

    } else if (type[i+128*j].equals("wallNoClipLineBottomDarkBlue") ||
type[i+128*j]=="wallNoClipLineBottomDarkBlue") {

      img=wallDarkBlue[7];

    } else if (type[i+128*j].equals("wallNoClipLineBottomLeftDarkBlue") ||
type[i+128*j]=="wallNoClipLineBottomLeftDarkBlue") {
```

```
        img=wallDarkBlue[8];

    } else if (type[i+128*j].equals("wallNoClipLineBottomLeftCornerDarkBlue") ||
type[i+128*j]=="wallNoClipLineBottomLeftCornerDarkBlue") {

        img=wallDarkBlue[9];

    } else if (type[i+128*j].equals("wallNoClipLineBottomRightDarkBlue")
||type[i+128*j]=="wallNoClipLineBottomRightDarkBlue") {

        img=wallDarkBlue[10];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightCornerDarkBlue") ||
type[i+128*j]=="emptywallNoClipLineBottomRightCornerDarkBlue") {

        img=wallDarkBlue[11];

    } else if (type[i+128*j].equals("wallNoClipLineLeftDarkBlue") ||
type[i+128*j]=="wallNoClipLineLeftDarkBlue") {

        img=wallDarkBlue[12];

    } else if (type[i+128*j].equals("emptywallNoClipLineRightDarkBlue") ||
type[i+128*j]=="emptywallNoClipLineRightDarkBlue") {

        img=wallDarkBlue[14];

    } else if (type[i+128*j].equals("wallNoClipLineTopDarkBlue") ||
type[i+128*j]=="wallNoClipLineTopDarkBlue") {

        img=wallDarkBlue[15];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftDarkBlue") ||
type[i+128*j]=="wallNoClipLineTopLeftDarkBlue") {

        img=wallDarkBlue[16];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftCornerDarkBlue") ||
type[i+128*j]=="wallNoClipLineTopLeftCornerDarkBlue") {

        img=wallDarkBlue[17];

    } else if (type[i+128*j].equals("wallNoClipLineTopRightDarkBlue") ||
type[i+128*j]=="wallNoClipLineTopRightDarkBlue") {

        img=wallDarkBlue[18];

    } else if (type[i+128*j].equals("emptywallNoClipLineTopRightCornerDarkBlue") ||
type[i+128*j]=="emptywallNoClipLineTopRightCornerDarkBlue") {

        img=wallDarkBlue[13];

    } else if (type[i+128*j].equals("wallNoClipRightDarkBlue")
||type[i+128*j]=="wallNoClipRightDarkBlue") {

        img=wallDarkBlue[20];
```

```
    } else if (type[i+128*j].equals("wallNoClipTopDarkBlue")
||type[i+128*j]=="wallNoClipTopDarkBlue") {

      img=wallDarkBlue[21];

    } else if (type[i+128*j].equals("wallNoClipTopLeftDarkBlue") ||
type[i+128*j]=="wallNoClipTopLeftDarkBlue") {

      img=wallDarkBlue[22];

    } else if (type[i+128*j].equals("wallNoClipTopLeftCornerDarkBlue") ||
type[i+128*j]=="wallNoClipTopLeftCornerDarkBlue") {

      img=wallDarkBlue[23];

    } else if (type[i+128*j].equals("wallNoClipTopRightDarkBlue") ||
type[i+128*j]=="wallNoClipTopRightDarkBlue") {

      img=wallDarkBlue[24];

    } else if (type[i+128*j].equals("wallNoClipTopRightCornerDarkBlue") ||
type[i+128*j]=="wallNoClipTopRightCornerDarkBlue") {

      img=wallDarkBlue[25];

    } else if (type[i+128*j].equals("wallNoClipChiseledDarkBlue")
||type[i+128*j]=="wallNoClipChiseledDarkBlue") {

      img=wallDarkBlue[19];

    } else if (type[i+128*j].equals("wallNoClipBottomLightBlue") ||
type[i+128*j]=="wallNoClipBottomLightBlue") {

      img=wallLightBlue[0];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftLightBlue") ||
type[i+128*j]=="wallNoClipBottomLeftLightBlue") {

      img=wallLightBlue[1];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftCornerLightBlue") ||
type[i+128*j]=="wallNoClipBottomLeftCornerLightBlue") {

      img=wallLightBlue[2];

    } else if (type[i+128*j].equals("wallNoClipBottomRightLightBlue") ||
type[i+128*j]=="wallNoClipBottomRightLightBlue") {

      img=wallLightBlue[3];

    } else if (type[i+128*j].equals("wallNoClipBottomRightCornerLightBlue") ||
type[i+128*j]=="wallNoClipBottomRightCornerLightBlue") {

      img=wallLightBlue[4];

    } else if (type[i+128*j].equals("wallNoClipCenterLightBlue") ||
type[i+128*j]=="wallNoClipCenterLightBlue") {
```

```
        img=wallLightBlue[5];

    } else if (type[i+128*j].equals("wallNoClipLeftLightBlue") ||
type[i+128*j]=="wallNoClipLeftLightBlue") {

        img=wallLightBlue[6];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomLightBlue") ||
type[i+128*j]=="emptywallNoClipLineBottomLightBlue") {

        img=wallLightBlue[7];

    } else if (type[i+128*j].equals("wallNoClipLineBottomLeftLightBlue") ||
type[i+128*j]=="wallNoClipLineBottomLeftLightBlue") {

        img=wallLightBlue[8];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomLeftCornerLightBlue") ||
type[i+128*j]=="emptywallNoClipLineBottomLeftCornerLightBlue") {

        img=wallLightBlue[9];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightLightBlue")
||type[i+128*j]=="emptywallLineBottomRightLightBlue") {

        img=wallLightBlue[10];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightCornerLightBlue") ||
type[i+128*j]=="emptywallNoClipLineBottomRightCornerLightBlue") {

        img=wallLightBlue[11];

    } else if (type[i+128*j].equals("wallNoClipLineLeftLightBlue") ||
type[i+128*j]=="wallNoClipLineLeftLightBlue") {

        img=wallLightBlue[12];

    } else if (type[i+128*j].equals("emptywallNoClipLineRightLightBlue") ||
type[i+128*j]=="emptywallNoClipLineRightLightBlue") {

        img=wallLightBlue[14];

    } else if (type[i+128*j].equals("wallNoClipLineTopLightBlue") ||
type[i+128*j]=="wallNoClipLineTopLightBlue") {

        img=wallLightBlue[15];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftLightBlue") ||
type[i+128*j]=="wallNoClipLineTopLeftLightBlue") {

        img=wallLightBlue[16];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftCornerLightBlue") ||
type[i+128*j]=="wallNoClipLineTopLeftCornerLightBlue") {

        img=wallLightBlue[17];
```

```
    } else if (type[i+128*j].equals("wallNoClipLineTopRightLightBlue") ||
type[i+128*j]=="wallNoClipLineTopRightLightBlue") {

      img=wallLightBlue[18];

    } else if (type[i+128*j].equals("emptywallNoClipLineTopRightCornerLightBlue") ||
type[i+128*j]=="emptywallNoClipLineTopRightCornerLightBlue") {

      img=wallLightBlue[13];

    } else if (type[i+128*j].equals("wallNoClipRightLightBlue")
||type[i+128*j]=="wallNoClipRightLightBlue") {

      img=wallLightBlue[20];

    } else if (type[i+128*j].equals("wallNoClipTopLightBlue")
||type[i+128*j]=="wallNoClipTopLightBlue") {

      img=wallLightBlue[21];

    } else if (type[i+128*j].equals("wallNoClipTopLeftLightBlue") ||
type[i+128*j]=="wallNoClipTopLeftLightBlue") {

      img=wallLightBlue[22];

    } else if (type[i+128*j].equals("wallNoClipTopLeftCornerLightBlue") ||
type[i+128*j]=="wallNoClipTopLeftCornerLightBlue") {

      img=wallLightBlue[23];

    } else if (type[i+128*j].equals("wallNoClipTopRightLightBlue") ||
type[i+128*j]=="wallNoClipTopRightLightBlue") {

      img=wallLightBlue[24];

    } else if (type[i+128*j].equals("wallNoClipTopRightCornerLightBlue") ||
type[i+128*j]=="wallNoClipTopRightCornerLightBlue") {

      img=wallLightBlue[25];

    } else if (type[i+128*j].equals("wallNoClipChiseledLightBlue")
||type[i+128*j]=="wallNoClipChiseledLightBlue") {

      img=wallLightBlue[19];

    } else if (type[i+128*j].equals("wallNoClipBottomGreen") ||
type[i+128*j]=="wallNoClipBottomGreen") {

      img=wallGreen[0];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftGreen") ||
type[i+128*j]=="wallNoClipBottomLeftGreen") {

      img=wallGreen[1];

    } else if (type[i+128*j].equals("wallNoClipBottomLeftCornerGreen") ||
type[i+128*j]=="wallNoClipBottomLeftCornerGreen") {
```

```
        img=wallGreen[2];

    } else if (type[i+128*j].equals("wallNoClipBottomRightGreen") ||
type[i+128*j]=="wallNoClipBottomRightGreen") {

        img=wallGreen[3];

    } else if (type[i+128*j].equals("wallNoClipBottomRightCornerGreen") ||
type[i+128*j]=="wallNoClipBottomRightCornerGreen") {

        img=wallGreen[4];

    } else if (type[i+128*j].equals("wallNoClipCenterGreen") ||
type[i+128*j]=="wallNoClipCenterGreen") {

        img=wallGreen[5];

    } else if (type[i+128*j].equals("wallNoClipLeftGreen") || type[i+128*j]=="wallNoClipLeftGreen") {

        img=wallGreen[6];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomGreen") ||
type[i+128*j]=="emptywallNoClipLineBottomGreen") {

        img=wallGreen[7];

    } else if (type[i+128*j].equals("wallNoClipLineBottomLeftGreen") ||
type[i+128*j]=="wallNoClipLineBottomLeftGreen") {

        img=wallGreen[8];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomLeftCornerGreen") ||
type[i+128*j]=="emptywallNoClipLineBottomLeftCornerGreen") {

        img=wallGreen[9];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightGreen")
||type[i+128*j]=="emptywallNoClipLineBottomRightGreen") {

        img=wallGreen[10];

    } else if (type[i+128*j].equals("emptywallNoClipLineBottomRightCornerGreen") ||
type[i+128*j]=="emptywallNoClipLineBottomRightCornerGreen") {

        img=wallGreen[11];

    } else if (type[i+128*j].equals("wallNoClipLineLeftGreen") ||
type[i+128*j]=="wallNoClipLineLeftGreen") {

        img=wallGreen[12];

    } else if (type[i+128*j].equals("emptywallNoClipLineRightGreen") ||
type[i+128*j]=="emptywallNoClipLineRightGreen") {

        img=wallGreen[14];

    } else if (type[i+128*j].equals("wallNoClipLineTopGreen") ||
type[i+128*j]=="wallNoClipLineTopGreen") {
```

```
      img=wallGreen[15];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftGreen") ||
type[i+128*j]=="wallNoClipLineTopLeftGreen") {

      img=wallGreen[16];

    } else if (type[i+128*j].equals("wallNoClipLineTopLeftCornerGreen") ||
type[i+128*j]=="wallNoClipLineTopLeftCornerGreen") {

      img=wallGreen[17];

    } else if (type[i+128*j].equals("wallNoClipLineTopRightGreen") ||
type[i+128*j]=="wallNoClipLineTopRightGreen") {

      img=wallGreen[18];

    } else if (type[i+128*j].equals("emptywallNoClipLineTopRightCornerGreen") ||
type[i+128*j]=="emptywallNoClipLineTopRightCornerGreen") {

      img=wallGreen[13];

    } else if (type[i+128*j].equals("wallNoClipRightGreen") ||type[i+128*j]=="wallNoClipRightGreen")
{

      img=wallGreen[20];

    } else if (type[i+128*j].equals("wallNoClipTopGreen") ||type[i+128*j]=="wallNoClipTopGreen") {

      img=wallGreen[21];

    } else if (type[i+128*j].equals("wallNoClipTopLeftGreen") ||
type[i+128*j]=="wallNoClipTopLeftGreen") {

      img=wallGreen[22];

    } else if (type[i+128*j].equals("wallNoClipTopLeftCornerGreen") ||
type[i+128*j]=="wallNoClipTopLeftCornerGreen") {

      img=wallGreen[23];

    } else if (type[i+128*j].equals("wallNoClipTopRightGreen") ||
type[i+128*j]=="wallNoClipTopRightGreen") {

      img=wallGreen[24];

    } else if (type[i+128*j].equals("wallNoClipTopRightCornerGreen") ||
type[i+128*j]=="wallNoClipTopRightCornerGreen") {

      img=wallGreen[25];

    } else if (type[i+128*j].equals("wallNoClipChiseledGreen")
||type[i+128*j]=="wallNoClipChiseledGreen") {

      img=wallGreen[19];

    } else if (type[i+128*j].startsWith("door") ||type[i+128*j].startsWith("door")) {

      img=door;
```

```
    }
    if (type[i+128*j].equals("BJump") || type[i+128*j]=="BJump") {
      fill(0, 255, 0, 150);
    } else if (type[i+128*j].equals("BnoClip") || type[i+128*j]=="BnoClip") {
      fill(255, 255, 255);
    } else if (type[i+128*j].equals("BTP") || type[i+128*j]=="BTP") {
      fill(140, 24, 202, 150);
    } else if (type[i+128*j].equals("BDash") || type[i+128*j]=="BDash") {
      fill(255, 0, 0, 150);
    } else if (type[i+128*j].startsWith("empty") || type[i+128*j]=="empty") {//||){
type[i+128*j]=="emptyspikeRedRight" || type[i+128*j]=="emptyspikePurpleBottom") {
      if (R.length()==0 || G.length()==0 || B.length()==0) {
      } else {
        fill(Integer.parseInt(R), Integer.parseInt(G), Integer.parseInt(B));//fill(152,64,30);
      }
      fill(120, 120, 120, 150);
    } else if (type[i+128*j].equals("noClip") || type[i+128*j]=="noClip") {
      fill(100, 100, 100, 150);
    } else if (type[i+128*j].equals("hero") || type[i+128*j]=="hero") {
      fill(255, 20, 147, 150);
    } else if (type[i+128*j].startsWith("end") || type[i+128*j].startsWith("end")) {
      fill(0, 255, 255, 150);
    } else if (type[i+128*j].equals("BGSwap") || type[i+128*j]=="BGSwap") {
      fill(30, 87, 142, 150);
    } else if (type[i+128*j].equals("BPoints") ||type[i+128*j]=="BPoints") {
      fill(255, 255, 0, 150);
    }
    if ((!type[i+128*j].equals("wall") && !type[i+128*j].equals("wallNoClip") &&
type[i+128*j].startsWith("wall")) || (!type[i+128*j].equals("spike") &&
type[i+128*j].startsWith("spike")) || (!type[i+128*j].equals("spike") &&
type[i+128*j].startsWith("emptyspike")) ||type[i+128*j].startsWith("door") ||
type[i+128*j].startsWith("emptywall")) {
      image(img, 50+8*i, 50+16*j);
```

```
    }
    if (isSaving==false) {
     if (R.length()!=0 && G.length()!=0 && B.length()!=0 && Integer.parseInt(R)!=120 &&
Integer.parseInt(G)!=100 && Integer.parseInt(B)!=100) {
       stroke(120, 100, 100);
     } else {
       stroke(150, 130, 130);
     }
     if ( type[i+128*j].startsWith("wallNoClip")) {
       fill(100, 100, 100, 150);
     }
     rect(50+8*i, 50+16*j, 8, 16);
     stroke(0, 0, 0);
    }
   }
  }
  if (isSaving==true) {
   isSaving=false;
   save=get(50, 50, 1024, 599);
   save.save("../../data/levels/lvlsPrint/"+lvl+".png");
  }
}




void mousePressed() {
  //click();
  hitbox();
}

void mouseDragged() {
```

```
  hitbox();
  //click();
}
```