Rapport du projet « Pacman »

Description du jeu

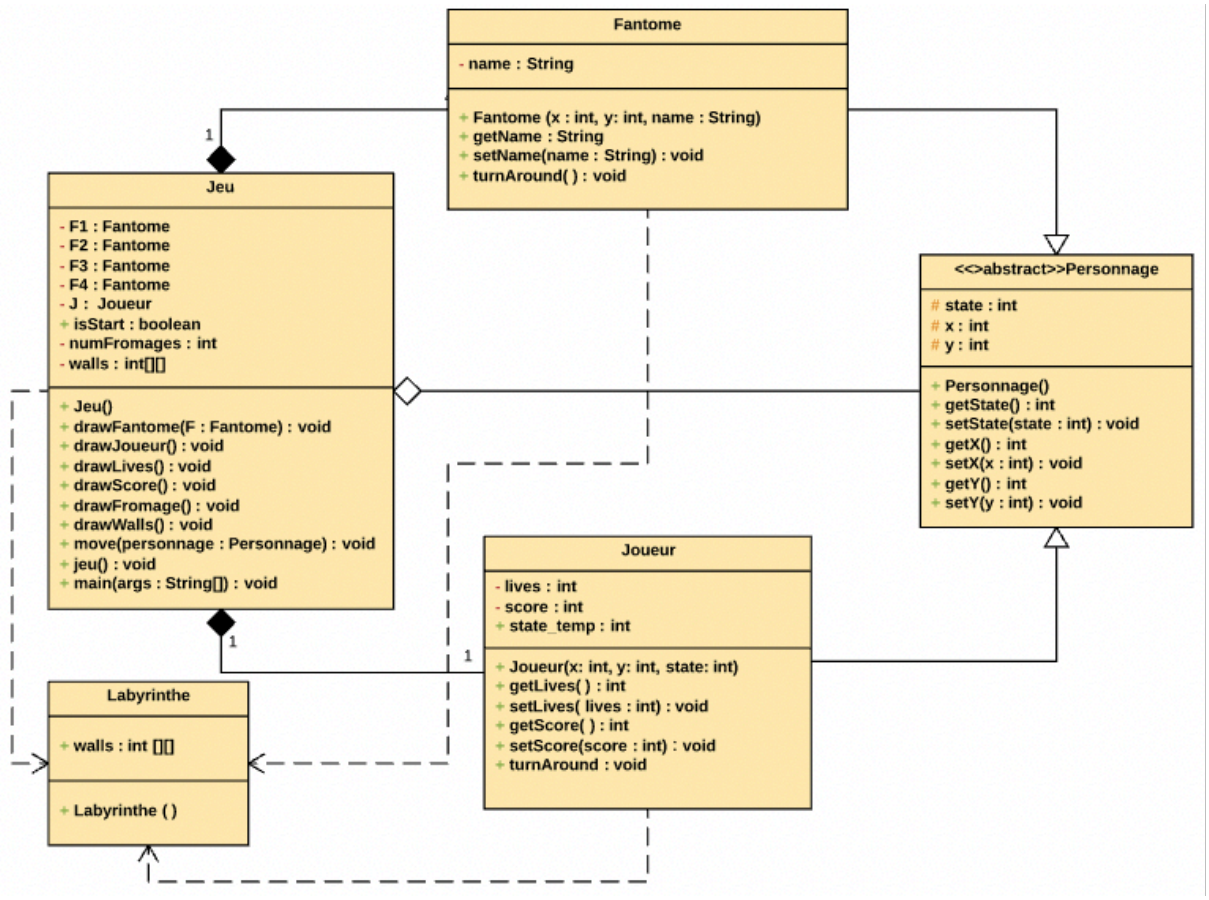
Le but du jeu est, pour le joueur dirigeant le pacman symbolisé par un rond jaune, de manger tous les fromages répartis sur les cases du labyrinthe. Les Fantômes sont des entités qui parcourent le labyrinthe au hasard. Le pacman a trois vies. Si le joueur se retrouve sur la même case qu'un fantôme, alors le joueur doit recommencer à son point de départ et une vie lui est enlevée. Le pacman avance automatiquement dans la direction choisie par le joueur avec les flèches du clavier. Lorsque le joueur choisit une direction, le pacman se tourne et avance dans la nouvelle direction. Lorsqu'un mur bloque le passage du pacman, il arrête d'avancer.

Si le pacman mange un petit fromage, il gagne 10 points ; s'il mange un grand fromage, il gagne 50 points. La partie s'arrête lorsque le joueur a mangé tous les fromages du labyrinthe ou il n'a plus de vie.

Au début, on demande le joueur à presser ESPACE pour commencer. Lorsque la partie est terminée, on affiche le score du joueur et on demande si l'utilisateur souhaite rejouer.

Modélisation

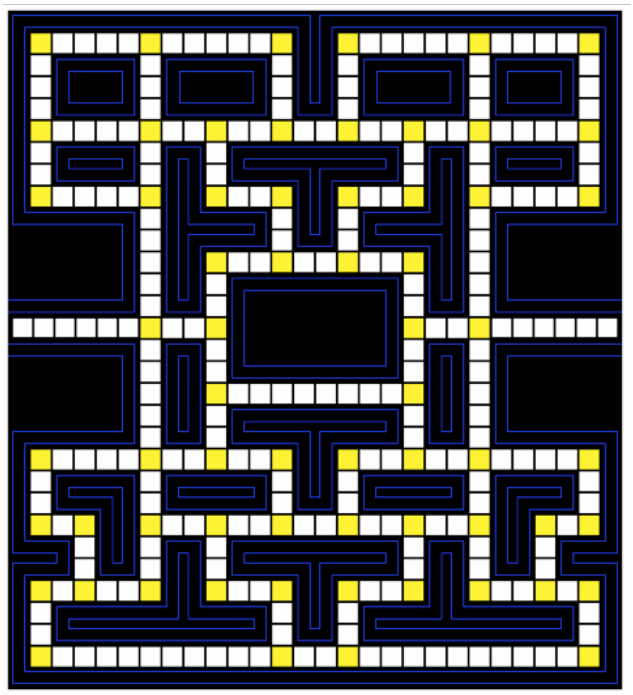
Notre projet a cinq classes : *Jeu*, *Fantome*, *Joueur*, *Labyrinthe* et *Personnage*. Personnage est une classe abstraite. Les classes *Fantome* et *Joueur* héritent la classes *Personnage*.



Description des fonctions

D’abord, nous décide à utiliser le même labyrinthe comme le figure suivant.

Nous trouvons que la carte pout être décomposée aux carreaux (31 lignes, 28 colonnes). Dans le figure suivant, les carreaux blancs et jaunes sont des chemins que les fantômes et le joueur peuvent passer.



Dans la classe *Labyrinthe*, il y a seulement une variable *walls* qui signifie le labyrinthe qu'on utilise. Un chiffre dans la matrice signifie un carreau. (0 = mur, 1 = petit fromage, 3 = porte, 4 = grand fromage)

```
public static int[][] walls = new int[][] {
    // 0 = wall, 1 = bean, 2 = empty, 3 = door,  4 = big bean
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0},
    {0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0},
    {0,4,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,4,0},
    {0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1,0},
    {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0},
    {0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0},
    {0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0},
    {0,1,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,0},
    {0,0,0,0,0,0,1,0,0,0,0,0,0,2,0,0,2,0,0,0,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,0,0,0,0,2,0,0,2,0,0,0,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,2,2,2,2,2,2,2,2,0,0,1,0,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,0,0,0,3,3,0,0,0,2,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,0,2,2,2,2,2,2,0,2,0,0,1,0,0,0,0,0},
    {2,2,2,2,2,2,1,2,2,2,0,2,2,2,2,2,2,0,2,2,2,1,2,2,2,2,2},
    {0,0,0,0,0,0,1,0,0,2,0,2,2,2,2,2,2,0,2,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,0,0,0,0,0,0,0,0,2,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,2,2,2,2,2,2,2,2,0,0,1,0,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,0,0,0,0,0,0,0,0,2,0,0,1,0,0,0,0,0},
    {0,0,0,0,0,0,1,0,0,2,0,0,0,0,0,0,0,0,2,0,0,1,0,0,0,0,0},
    {0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,0},
    {0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1,0},
    {0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,1,0},
    {0,4,1,1,0,0,1,1,1,1,1,1,2,2,1,1,1,1,1,1,1,0,0,1,1,4,0},
    {0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0},
    {0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0},
    {0,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,1,1,0},
    {0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,0},
    {0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,1,0},
    {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0},
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
};
```

Dans la classe abstraite *Personnage*, il y a trois variables de type int. « x » et « y » signifie le coordonnée d'un fantôme ou d'un joueur. « state » signifie la direction du caractère.


```
public abstract class Personnage {
    protected int x;
    protected int y;
    protected int state = 1;

    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
    public int getState(){
        return state;
    }
    public void setState(int state){
        this.state = state;
    }
}
```

Dans la classes *Fantôme*, à part les getters et les setters, il y a une fonction principe « turnAround ». Il permet de changer la direction du fantôme. Comme les fantômes changent la direction aléatoirement quand ils arrivent dans les carreaux jaunes, on utilise la méthode « Math.random ».
Par exemple, quand state = 1 (le fantôme face à gauche), on doit définir les états des trois carreaux, précisément, Labyrinthe.walls[y][x-1] (le carreau à côté de gauche), Labyrinthe.walls[y-1][x] (le carreau au-dessus), et Labyrinthe.walls[y+1][x](le carreau au-dessous).

```
public void turnAround(){
    if((x==13||x==14)&&y==11){
        if(state !=1 || state !=3)
            state = 1;
        else if(state == 1)
            state = 1;
        else if (state == 3)
            state = 3;
    }
    else if((x==11||x==12) && (y==12||y==13||y==14||y==15)) // quand les fantômes sont dans le prison
        state = 3;
    else if((x==13||x==14) && (y==12||y==13||y==14||y==15)) // quand les fantômes sont dans le prison
        state = 2 ;
    else if((x==15||x==16) && (y==12||y==13||y==14||y==15)) // quand les fantômes sont dans le prison
        state = 1 ;
    else if(x==0 && state==1)
        state=1;
    else if(x==27 && state==3)
        state=3;
    else {
        switch (state){
        case 1:
            if(Labyrinthe.walls[y][x-1]==0 && Labyrinthe.walls[y-1][x]!=0 && Labyrinthe.walls[y+1][x]==0)
                state = 2;
            else if(Labyrinthe.walls[y][x-1]==0 && Labyrinthe.walls[y-1][x]==0 && Labyrinthe.walls[y+1][x]!=0)
                state = 4;
            else if(Labyrinthe.walls[y][x-1]!=0 && Labyrinthe.walls[y-1][x]==0 && Labyrinthe.walls[y+1][x]!=0){
                if(Math.random()<0.5)
                    state = 1;
                else
                    state = 4;

            }else if(Labyrinthe.walls[y][x-1]==0 && Labyrinthe.walls[y-1][x]!=0 && Labyrinthe.walls[y+1][x]!=0){
                if(Math.random()<0.5)
                    state = 2;
                else
                    state = 4;
            }else if(Labyrinthe.walls[y][x-1]!=0 && Labyrinthe.walls[y-1][x]!=0 && Labyrinthe.walls[y+1][x]==0){
                if(Math.random()<0.5)
                    state = 1;
                else
                    state = 2;
            }else if(Labyrinthe.walls[y][x-1]!=0 && Labyrinthe.walls[y-1][x]!=0 && Labyrinthe.walls[y+1][x]!=0){
                if(Math.random()<0.33)
                    state = 1;
                else if (Math.random() > 0.66)
                    state = 4;
                else
                    state = 2;
            }
            break;
```

... (case 2, 3, 4 ont le même principe avec case 1)

Dans la classe *Joueur*, il y a trois variables : « score » (les points que le joueur gagne), « lives » (nombre de vies du joueur, valeur initiale est 3) et « state_temp ». « state_temp » faire le joueur mémoriser la direction qu'on choit et changer la direction correspondante dès il arrive dans un carreau jaune.

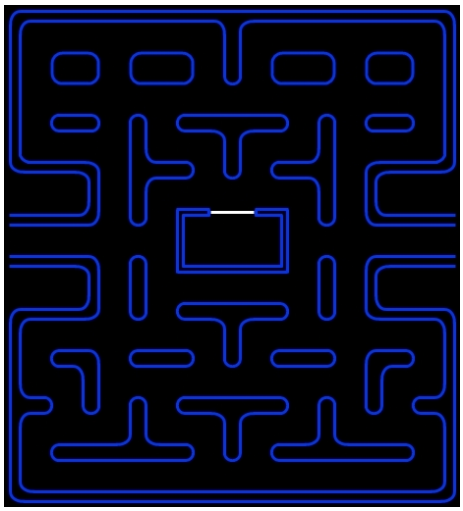
```
public void turnAround(){
    if (StdDraw.isKeyPressed(KeyEvent.VK_DOWN)) {
        state_temp = 4;
    }else if (StdDraw.isKeyPressed(KeyEvent.VK_LEFT)) {
        state_temp = 1;
    }else if (StdDraw.isKeyPressed(KeyEvent.VK_UP)) {
        state_temp = 2;
    }
    else if (StdDraw.isKeyPressed(KeyEvent.VK_RIGHT)) {
        state_temp = 3;
    }

    switch(state_temp){
    case 1:
        if(x!=0){
            if(Labyrinthe.walls[y][x-1]!=0)
                state = 1;
        }else
                state = 1;
        break;
    case 2:
        if(Labyrinthe.walls[y-1][x]!=0){
            state = 2;
        }
        break;
    case 3:
        if (x!=27){
            if(Labyrinthe.walls[y][x+1]!=0)
                state = 3;
        }else
            state = 3;
        break;
    case 4:
        if(Labyrinthe.walls[y+1][x]!=0 && Labyrinthe.walls[y+1][x]!=3)
            state = 4;
        break;
    }
}
```

Dans la classe *Jeu*, nous utilisons beaucoup de fonctions qui permettent de dessiner les caractères, les fromages, les murs, le chiffre des points gagnés et le chiffre de vies.

Dans la fonction « drawWalls », on utilise la méthode « StdDraw.picture » avec cinq paramètres. *SPACE* est une valeur fixe qui représente la longueur d'un côté.

```
public static void drawWalls(){
    StdDraw.picture(SPACE_BODER+SPACE*14,SPACE_SCORE+SPACE*16, "images/maze.jpg",SPACE*29,SPACE*32);
}
```



Dans la fonction « drawFromage », on utilise une boucle for pour changer l'état de la position où le fromage est mangé ou pour dessiner des fromages restes. On utilise la méthode « StdDraw.filledCircle » pour dessiner les fromages représentant par des petits points blancs ronds.
Remarque : il y a deux types de fromage (grand et petit), si le joueur mange un petit fromage, il gagne 10 points ; s'il mange un grand fromage, il gagne 50 points. Dans le jeu Pacman officiel, si le joueur mange un grand fromage, tous les fantômes deviennent vulnérables, mais nous n'avons pas réalisé ce fonctionnement.

```
public static void drawFromage(){
    for (int i=0; i<walls.length; i++){
        for(int j=0; j<walls[0].length;j++){
            if(walls[i][j]==1){
                if(i==J.getY() && j==J.getX()){
                    walls[i][j]=2; // Si les positions du joueur et d'un petit fromage sont même,
                                   // l'état de cette position est changé à 2 (= vide),
                    numFromages -=1; // et le nombre des fromages restes est moins 1

                    J.setScore(J.getScore()+10); // Si le joueur mange un petit fromage, il gagne 10 points.
                }else{
                    StdDraw.setPenColor(StdDraw.WHITE);
                    StdDraw.filledCircle(j*SPACE+SPACE/2+SPACE_BODER, SPACE_SCORE+(walls.length-i)*SPACE, 2);
                }
            }else if(walls[i][j]==4){
                if(i==J.getY() && j==J.getX()){
                    walls[i][j]=2; // même principe avec le petit fromage
                    numFromages -=1;
                    J.setScore(J.getScore()+50); // Si le joueur mange un grand fromage, il gagne 50 points.

                }else{
                    StdDraw.setPenColor(StdDraw.WHITE);
                    StdDraw.filledCircle(j*SPACE+SPACE/2+SPACE_BODER, SPACE_SCORE+(walls.length-i)*SPACE, 7.5);
                }
            }
        }
    }
}
```

Dans la fonction « drawJoueur », on utilise « StdDraw.picture » à dessiner le joueur ayant la direction correspondante.
L'état du joueur :


```
public static void drawJoueur(){
    int x = J.getX();
    int y = J.getY();
    int degre = 0; // Ce paramètre permet de choisir le degré de rotation de l'image.
    switch(J.getState()){ // Le joueur a quatre états : 1 = face à gauche, 2 = face en haut, 3 = face à droite, 4 = face en b
    case 1: degre = 180; break;
    case 2: degre = 90; break;
    case 3: degre = 0; break; // L'état initial de l'image est à gauche.
    case 4: degre = 270; break;

    }
    StdDraw.picture(x*SPACE+SPACE/2+SPACE_BODER,
            SPACE_SCORE+SPACE_SCORE+(walls.length-y-1)*SPACE-SPACE/2-10,
            "images/joueur.png",  SPACE*1.5, SPACE*1.5, degre);
}
```

Pour la fonction « drawFantome », on utilise la structure *if ... else* pour déterminer la bonne image du fantôme.


```
public static void drawFantome(Fantome F){
    int x = F.getX();
    int y = F.getY();
    String image;
    if (F.getName()=="Blinky"){
        image = "images/blinky.png";
    }else if(F.getName()=="Pinky"){
        image = "images/pinky.png";
    }else if(F.getName()=="Inky"){
        image = "images/inky.png";
    }else{
        image = "images/clyde.png";
    }
    StdDraw.picture(x*SPACE+SPACE/2+SPACE_BODER,
            SPACE_SCORE+SPACE_SCORE+(walls.length-y-1)*SPACE-SPACE/2-10,
            image, SPACE*1.5, SPACE*1.5, 0);
}
```

Pour les fonctions « drawScore » et « drawLives », l'interface graphique est :

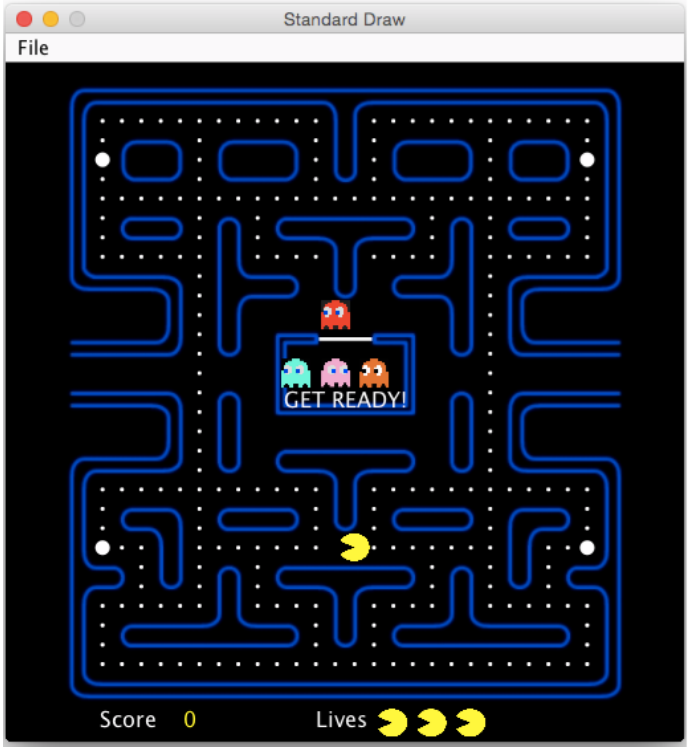

```
public static void drawScore(){
    StdDraw.setPenColor(StdDraw.WHITE);
    StdDraw.text(80, 20, "Score " );
    StdDraw.setPenColor(StdDraw.YELLOW);
    StdDraw.text(140, 20, ""+J.getScore());
}

public static void drawLives(){
    StdDraw.setPenColor(StdDraw.WHITE);
    StdDraw.text(300, 20, "Lives " );
    if (J.getLives()>0){ // Si le joueur a au moin une vie, on dessine une vie.
        StdDraw.picture(350, 20,"images/joueur.png", 30, 30, 180);
        if(J.getLives()>1){ // S'il a au moin deux vies, on dessine une deuxième vie.
            StdDraw.picture(390, 20,"images/joueur.png", 30, 30, 180);
            if(J.getLives()>2) // S'il a trois vies, on dessine une dernière vie.
                StdDraw.picture(430, 20,"images/joueur.png", 30, 30, 180);
        }
    }
}
```

La fonction « move » permet d'avancer d'un carreau un fantôme ou le joueur automatiquement. On utilise la structure switch case pour définir la direction.

```
public static void move(Personnage personnage){
    switch (personnage.getState()){ // 4 états
    case 1 :
        if(personnage.getX()==0)
            personnage.setX(27); // Dans ce cas, le personnage peut traverser de gauche à droit
        else if(Labyrinthe.walls[personnage.getY()][personnage.getX()-1] != 0){
            personnage.setX(personnage.getX()-1);
        }
        break;
    case 2 :
        if(Labyrinthe.walls[personnage.getY()-1][personnage.getX()] != 0){
            personnage.setY(personnage.getY()-1);
        }
        break;
    case 3 :
        if(personnage.getX()==27)
            personnage.setX(0); // Dans ce cas, le personnage peut traverser de droit à gauche
        else if(Labyrinthe.walls[personnage.getY()][personnage.getX()+1] != 0){
            personnage.setX(personnage.getX()+1);
        }
        break;
    case 4 :
        if(Labyrinthe.walls[personnage.getY()+1][personnage.getX()] != 0){
            personnage.setY(personnage.getY()+1);
        }
        break;
    }
}
```

Dans la première partie de la fonction « jeu », on dessine les murs, les fromages, « score », « lives » et « GET READY ». De plus, le joueur et les quatre fantômes se tournent si nécessaire et s'avancent.



```
public static void jeu(){
    StdDraw.clear(StdDraw.BLACK);
    drawWalls();
    drawFromage();
    drawScore();
    drawLives();
    if(isStart){
        StdDraw.setPenColor(StdDraw.WHITE);
        StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2, "GET READY!");
        isStart = false;
    }

    J.turnAround();
    move(J);
    drawJoueur();

    /*
     * Quand le joueur est mangé par un fantôme
     */
    if ((J.getX()==F1.getX()&&J.getY()==F1.getY())||
            (J.getX()==F2.getX()&&J.getY()==F2.getY())||
            (J.getX()==F3.getX()&&J.getY()==F3.getY())||
            (J.getX()==F4.getX()&&J.getY()==F4.getY())){
        J.setLives(J.getLives()-1);

        isStart = true;
        /*
         * initialiser les positions des fantômes et du joueur
         */
        J.setX(14);
        J.setY(23);
        J.setState(1);
        F1.setX(13);
        F1.setY(12);
        F2.setX(14);
        F2.setY(15);
        F3.setX(12);
        F3.setY(15);
        F4.setX(16);
        F4.setY(15);
    }
```

```
    F1.turnAround();
    move(F1);
    drawFantome(F1);

    F2.turnAround();
    move(F2);
    drawFantome(F2);

    F3.turnAround();
    move(F3);
    drawFantome(F3);

    F4.turnAround();
    move(F4);
    drawFantome(F4);
```

```
    if ((J.getX()==F1.getX()&&J.getY()==F1.getY())||
            (J.getX()==F2.getX()&&J.getY()==F2.getY())||
            (J.getX()==F3.getX()&&J.getY()==F3.getY())||
            (J.getX()==F4.getX()&&J.getY()==F4.getY())){
        J.setLives(J.getLives()-1);
        isStart = true;

        J.setX(14);
        J.setY(23);
        J.setState(1);
        F1.setX(13);
        F1.setY(12);
        F2.setX(14);
        F2.setY(15);
        F3.setX(12);
        F3.setY(15);
        F4.setX(16);
        F4.setY(15);

        J.state_temp = 1;

    }

    StdDraw.show(120); // Les personnages s'avancent chaque 120 ms.
```

Pour la partie suivante, si le joueur mange tous les fromages, le jeu est terminé, le score est affiché et on est demandé à presser ESPACE pour redémarrer le jeu.


```
if(numFromages==0){ // s'il n'y a plus de fromage, le joueur gagne.
    do{
        StdDraw.clear(StdDraw.BLACK);
        StdDraw.setPenColor(StdDraw.WHITE);
        StdDraw.picture(WIN_WIDTH/2,  WIN_HEIGHT/2+80, "images/youWin.jpeg");
        StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2-20, "Your score : " + J.getScore());
        StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2-60, "Press SPACE to try again !" );
        StdDraw.show(20);
    }while(!StdDraw.isKeyPressed(KeyEvent.VK_SPACE));
    J.state_temp = 1;
    numFromages = 244;
    for (int i=0; i<Labyrinthe.walls[0].length;i++){
        for (int j=0; j<Labyrinthe.walls.length; j++){
            walls[j][i] = Labyrinthe.walls[j][i];
        }
    }
    new Jeu();
}
```

Pour la partie suivante, si le joueur n'a plus de vie, le jeu est terminé, le score est affiché et on est demandé à presser ESPACE pour redémarrer le jeu.


```
    if(J.getLives()==0){ // si le joueur n'a plus de vie, il mort.
        do{
            StdDraw.clear(StdDraw.BLACK);
            StdDraw.setPenColor(StdDraw.WHITE);
            StdDraw.picture(WIN_WIDTH/2,  WIN_HEIGHT/2+80, "images/gameOver.jpeg");
            StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2-50, "Your score : " + J.getScore());
            StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2-90, "Press SPACE to try again !" );
            StdDraw.show(20);
        }while(!StdDraw.isKeyPressed(KeyEvent.VK_SPACE));
        J.state_temp = 1;
        numFromages = 244;
        for (int i=0; i<Labyrinthe.walls[0].length;i++){
            for (int j=0; j<Labyrinthe.walls.length; j++){
                walls[j][i] = Labyrinthe.walls[j][i];
            }
        }
        new Jeu();
    }

}
```

Pour la fonction main, on définir la première interface graphique. Le joueur est demandé à presser ESPACE pour commencer le jeu.


```
public static void main(String [] args) {
    StdDraw.setXscale(-50, WIN_WIDTH+50);
    StdDraw.setYscale(0, WIN_HEIGHT);
    do{
        StdDraw.clear(StdDraw.BLACK);
        StdDraw.setPenColor(StdDraw.WHITE);
        StdDraw.picture(WIN_WIDTH/2,  WIN_HEIGHT/2+80, "images/logo.png");
        StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2-40, "Press SPACE to start !" );
        StdDraw.show(20);
    }while(!StdDraw.isKeyPressed(KeyEvent.VK_SPACE));

    for (int i=0; i<Labyrinthe.walls[0].length;i++){
        for (int j=0; j<Labyrinthe.walls.length; j++){
            walls[j][i] = Labyrinthe.walls[j][i];
        }
    }
    StdDraw.clear(StdDraw.BLACK);
    drawWalls();
    drawFromage();
    drawScore();
    drawLives();
    if(isStart){
        StdDraw.setPenColor(StdDraw.WHITE);
        StdDraw.text(WIN_WIDTH/2, WIN_HEIGHT/2, "GET READY!");
        isStart = false;
    }
    drawFantome(F1);
    drawFantome(F2);
    drawFantome(F3);
    drawFantome(F4);
    drawJoueur();
    StdDraw.show(20);
    StdDraw.pause(3000);
    while (true){
        jeu();
    }
}
```

Conclusion

Ce projet nous permet de comprendre comment utiliser StdDraw en faisant des recherches sur internet et augmenter le niveau de nos compétences d'utiliser le Java.

Nous avons bien réalisé le cahier de charge et ajouté plusieurs fonctionnements supplémentaires.

Nous avons essayé d'ajouter la musique d'ambiance et faire les fantômes vulnérables quand le joueur mange le grand fromage, mais nous n'avons pas réussi.