Soutenance

Analyse et Conception de Logiciel

BEN KASDALLAH Fadoua - DUGAST Simon - HELBLING Hugo- PECASTAING Hugo

Introduction

- Jeu mono-utilisateur
- Interface graphique
- Héros dans labyrinthe
- Attraper un trésor
- Présence de monstres, pièges…
- Méthodes de conception de jeu

Annonce du plan

- Diagramme Use-Case
- Diagramme de classes
- Diagramme de séquences
- Organisation du projet
- Liste des fonctionnalités
- Bilan des compétences

Diagramme Use-Case

Diagramme Use-Case

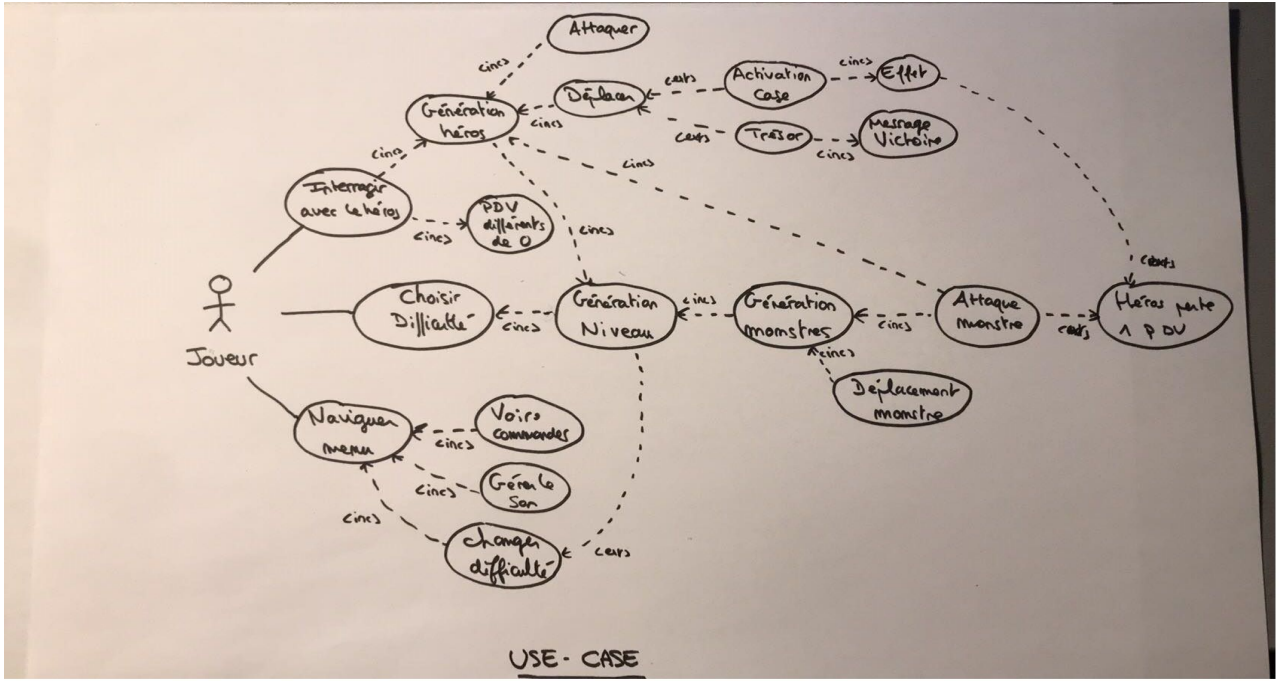


Diagramme de classes

Diagramme de classes

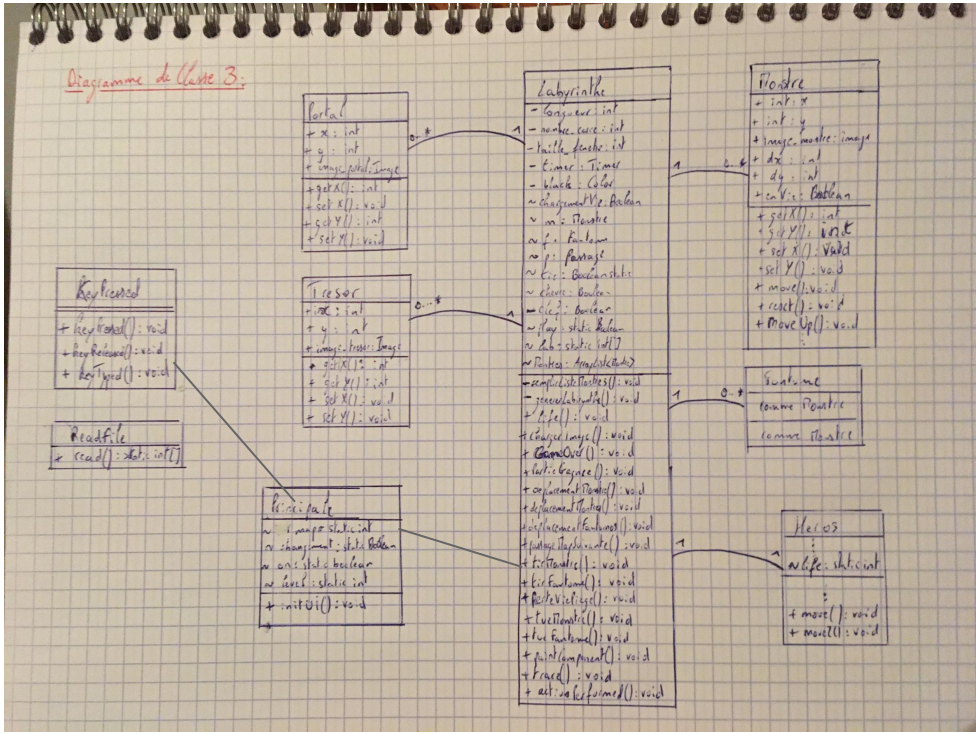
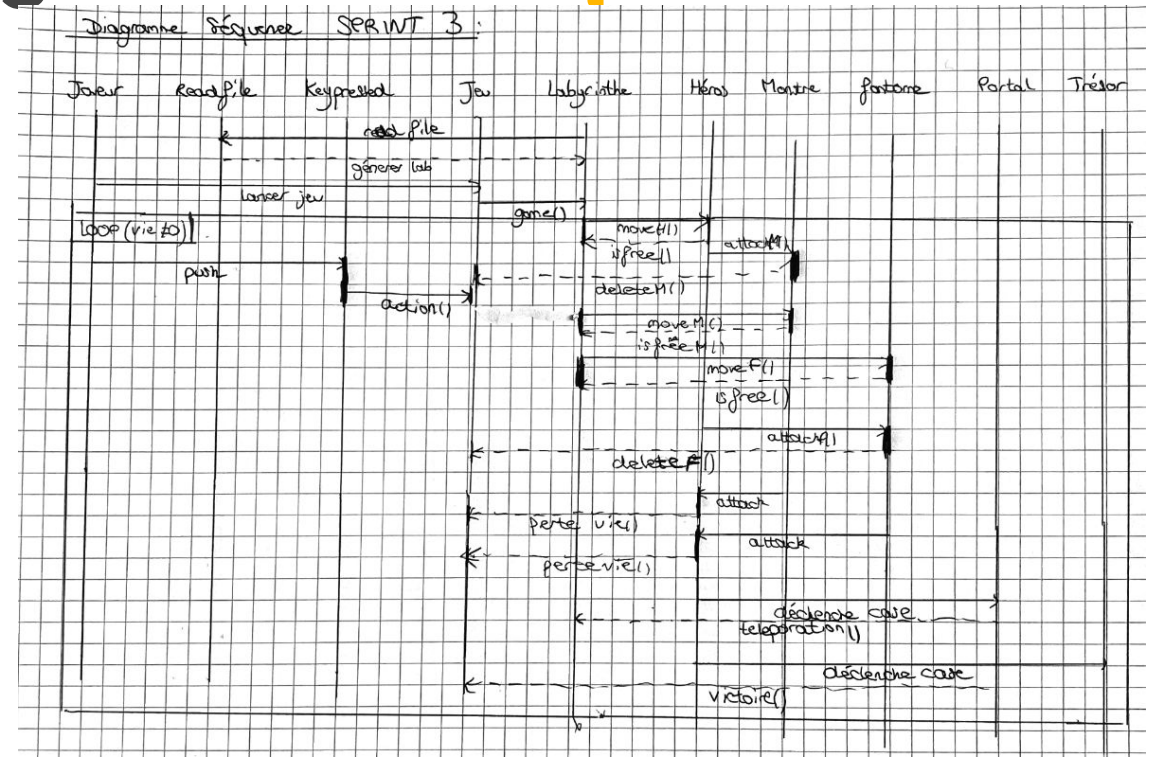


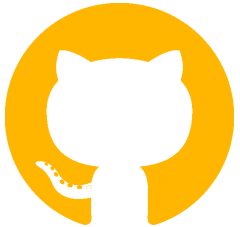
Diagramme de séquences

Diagramme de séquences



Organisation du projet

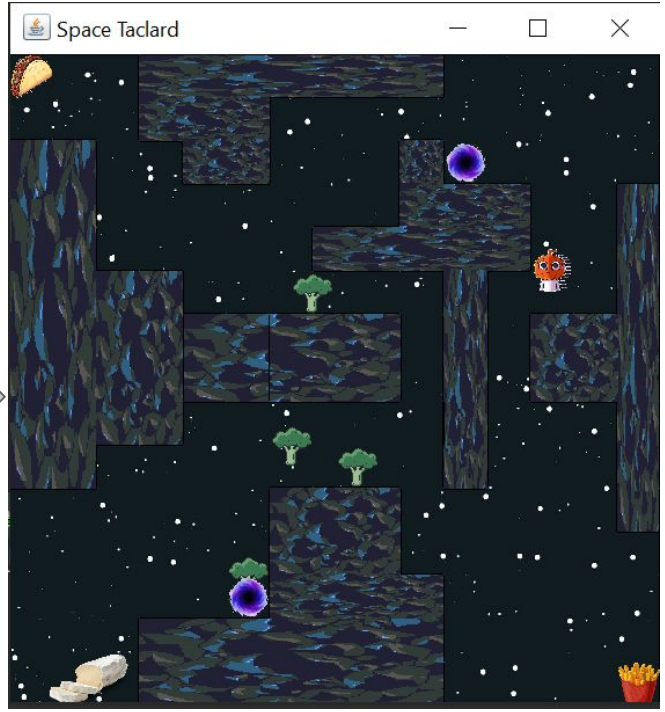
Liste des fonctionnalités par sprint



Sprint 1	Sprint 2	Sprint 3
• Générer un labyrinthe • Générer un monstre • Générer un trésor • Le héro peut se déplacer à l'aide des flèches du clavier • Le déplacement du monstre n'est pas aléatoire • Une collision avec le monstre entraîne la fin de la partie • Déplacer le héro sur la case trésor fait gagner la partie	• Ajout de textures et obstacles sur le labyrinthe • Ajout d'une case téléporteur • Ajout d'une clé pour déverrouiller le trésor • Ajout du fantôme • Le déplacement des monstres est aléatoire • Les monstres traversent encore les murs	• Seul le fantôme peut traverser les murs • Encore certain soucis de collision entre les monstres et le héro • Ajout d'une interface graphique : menu principal, d'un game over et d'un menu de victoire • Ajout d'un système de vie pour le héro • Ajout de la possibilité de tuer les monstres

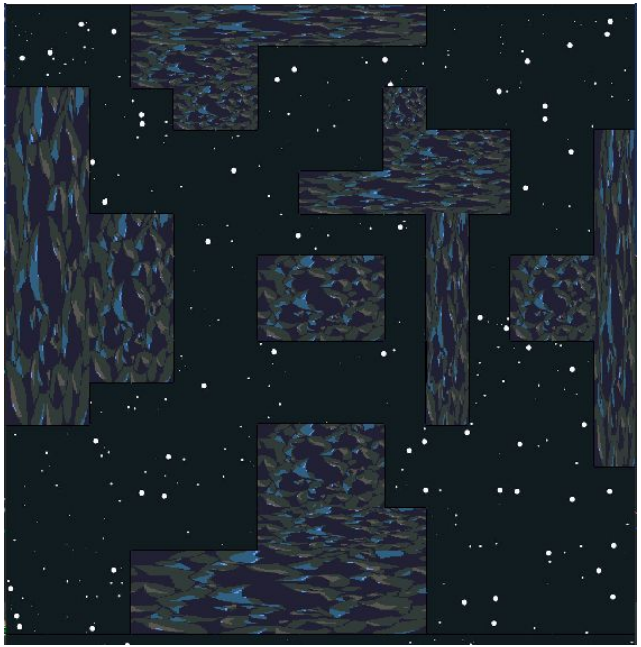
Liste des fonctionnalités

Liste des fonctionnalités



Liste des fonctionnalités

Labyrinthe:



- remplirListeMonstres() -> nb de monstres
- genererLabyrinthe(Graphics2D g2d,int map) -> trace les murs
- life() -> gère la vie du héros
- chargerImage(Graphics2D g2d,int dxx, int dyy) -> charge toutes les images
- GameOver(Graphics g2d)
- PartieGagnee(Graphics g2d)
- deplacementMonstre(Graphics g2d, Monstre M) -> +fantome
- passagePortal(Graphics g2d) -> téleport
- tirMonstre(Monstre m) -> +fantome
- tueMonstre(Graphics g2d)-> +fantome
- paintComponent(Graphics g) -> affiche
- trace(Graphics g) -> appel des fonctions

Liste des fonctionnalités

```
if (choix.size()!=0) {
    int nombreAleatoire = random.nextInt(choix.size());
    String direction = choix.get(nombreAleatoire);
    switch (direction) {
    case "Left":
        moveLeft();
        break;
    case "Right":
        moveRight();
        break;
    case "Up":
        moveUp();
        break;
    case "Down":
        moveDown();
        break;
    }
}
else {
    dx=-dx;
    dy=-dy;
}
```

```
void move() {
    Random random = new Random();
    ArrayList<String> choix = new ArrayList();
    if (x % 25 == 0 && y % 25 ==0) {
        int position = x/25 + 15 * (int)(y/25);
        if ((Labyrinthe.lab[position] & 1) ==0 && dx != 1) {
            choix.add("Left");
        }
        if ((Labyrinthe.lab[position] & 2) ==0 && dy != 1) {
            choix.add("Up");
        }
        if ((Labyrinthe.lab[position] & 4) ==0 && dx != -1) {
            choix.add("Right");
        }
        if ((Labyrinthe.lab[position] & 8) ==0 && dy != -1) {
            choix.add("Down");
        }
```

```
void moveRight() {
    // TODO Auto-generated method stub
    setX(x+1);
    dx=1;
    dy=0;
}
```

Monstres :

- Setters & getters
- Déplacements aléatoires
- Direction opposée choisis si arrive contre un mur

Liste des fonctionnalités

```java
import java.awt.Image;

public class Fantome {
    public int x=25*10;
    public int y=25*5;
    public Image image_fantome;
    static int dx=0;
    static int dy=0;
    public boolean enVie=true;
```

```java
void move() {
    if (x%25 == 0 && y%25 == 0) {
        Random random = new Random();
        ArrayList<String> choix = new ArrayList();
        choix.add("Left");
        choix.add("Right");
        choix.add("Up");
        choix.add("Down");
        String direction = choix.get(random.nextInt(choix.size()));
        switch (direction) {
        case "Left":
            moveLeft();
            break;
        case "Right":
            moveRight();
            break;
        case "Up":
            moveUp();
            break;
        case "Down":
            moveDown();
            break;
        }
```

```java
    if (x>25*14) {
        moveLeft();
    }
    if (x<0) {
        moveRight();
    }
    if (y<0) {
        moveDown();
    }
    if (y>25*14) {
        moveUp();
    }
```

Fantomes :

- Setters & getters
- Déplacement aléatoire
- Traverse les murs...
- ...mais les bords

Liste des fonctionnalités

```
public class Heros extends JPanel {
    public int x;
    public int y;
    public Image taco;
    static int dx=0;
    static int dy=0;
    static int life=2;

    public int getX() {

    public void setX(int x) {

    public int getY() {

    public void setY(int y) {
```

```
public void move() {
    if (dx==1) {
        setX(x+25);
    }
    if (dx==-1) {
        setX(x-25);
    }
    if (dy==-1) {
        setY(y+25);
    }
    if (dy==1) {
        setY(y-25);
    }
}
```

Héros :

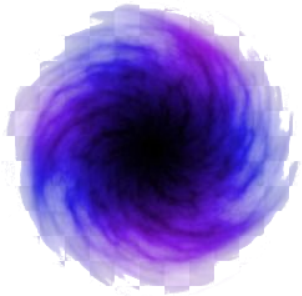
```
public void move2() {
    int pos = x/25+15*(int)y/25;
    int pos_ = Labyrinthe.lab[pos];
    if (dx==-1 && dy==0 && (pos_ & 1) != 0 || dx==1 && dy==0 && (pos_ & 4) != 0 ||
            dx==0 && dy==-1 && (pos_ & 8) != 0 || dx==0 && dy==1 && (pos_ & 2) != 0 ) {
        dx=0;
        dy=0;
    }
    else {
        move();
        dx=0;
        dy=0;
    }
}
```

- Setters & getters
- Deplacement case par case
- Impossibilité de rentrer dans les murs

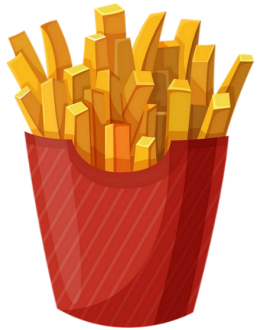
Liste des fonctionnalités

Cases :

Clef obligatoire à la victoire



Portail de téléportation



Trésor accessible que via la clef

- Setters & getters

Liste des fonctionnalités

ReadFile:

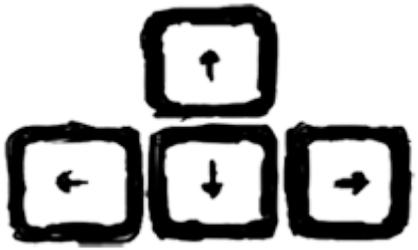
```java
public class ReadFile {

    public static int[] read(String fichier){
        int[] lab = new int[15*15];
        String adressedufichier = System.getProperty("user.dir") + "/" + "Ressources" + "/";
        try{
            InputStream ips=new FileInputStream(adressedufichier+fichier);
            InputStreamReader ipsr=new InputStreamReader(ips);
            BufferedReader br=new BufferedReader(ipsr);
            String ligne;
            int j=0;
            while ((ligne=br.readLine())!=null){
                for (int i=0; i<15; i++) {
                    lab[i+j]=Integer.parseInt(ligne.split(", ")[i]);
                }
                j+=15;
            }
            br.close();
        }
        catch (Exception e){
            System.out.println(e.toString());
        }
        return lab;
    }
}
```

```
19, 18, 22, 18, 18, 18, 18, 18, 18, 18, 19, 18, 18, 18, 22,
25, 24, 20, 0, 0, 0, 19, 18, 18, 26, 0, 0, 0, 0, 20,
17, 0, 17, 22, 0, 0, 17, 0, 20, 0, 25, 24, 0, 0, 28,
17, 0, 17, 0, 18, 18, 0, 24, 28, 0, 0, 0, 17, 20, 20,
17, 0, 25, 24, 0, 0, 20, 0, 0, 0, 0, 0, 17, 20, 20,
17, 0, 0, 0, 25, 24, 24, 26, 26, 22, 0, 19, 24, 28, 20,
17, 0, 0, 0, 17, 20, 0, 0, 0, 21, 0, 21, 0, 0, 20,
17, 0, 0, 0, 17, 20, 0, 0, 0, 21, 0, 21, 0, 0, 20,
17, 0, 0, 0, 19, 18, 18, 18, 18, 20, 17, 17, 18, 22, 20,
17, 0, 19, 18, 0, 0, 24, 24, 24, 20, 0, 17, 0, 20, 20,
19, 18, 0, 0, 0, 20, 0, 0, 0, 17, 18, 0, 0, 20, 20,
17, 0, 0, 0, 0, 20, 0, 0, 0, 25, 0, 0, 0, 0, 22,
17, 0, 0, 24, 24, 28, 0, 0, 0, 0, 17, 0, 0, 0, 20,
17, 0, 20, 0, 0, 0, 0, 0, 0, 0, 17, 0, 0, 0, 20,
25, 24, 28, 24, 24, 24, 24, 24, 24, 24, 25, 24, 24, 24, 28,
```

- Va lire le fichier donnant les murs du labyrinthe

Liste des fonctionnalités

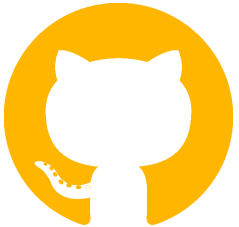
keyPressed::



```
public class keyPressed implements KeyListener {

    @Override
    public void keyPressed(KeyEvent arg0) {
        int touche = arg0.getKeyCode();
        switch (touche)
        {
        case KeyEvent.VK_UP:
            Heros.dy=1;
            break;
        case KeyEvent.VK_LEFT:
            Heros.dx=-1;
            break;
        case KeyEvent.VK_RIGHT:
            Heros.dx=1;
            break;
        case KeyEvent.VK_DOWN:
            Heros.dy=-1;
            break;

        case KeyEvent.VK_SPACE:
            if (Labyrinthe.play==false) {
                Labyrinthe.play=true;
            }
            if (Labyrinthe.play==true) {
                Labyrinthe.tir = true;
            }
            if (Labyrinthe.replay=true) {
                Labyrinthe.play=true;
            }
```

- Reconnaît quand le joueur appuie sur une touche
- Reconnaît quand il la relâche

Bilan des compétences



Gestion de projet

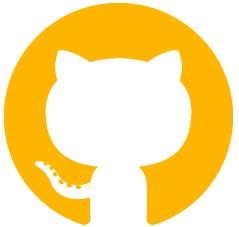
Travail en groupe

Programmer en JAVA

Modélisation du problème en UML

Travail en autonomie

Conclusion



1. Héros

1) Le héros est placé sur le plateau de jeu et peut s'y déplacer à l'intérieur. ✓

2.2. Labyrinthe

1) Le labyrinthe est généré par défaut - le héros et les monstres ne peuvent pas traverser les murs. ✓

2) Le labyrinthe est généré à partir d'un fichier. ✓

3) Le labyrinthe est généré en fonction du niveau sélectionné.

4) Certains cases du labyrinthe sont spéciales : • trésor : si le héros arrive sur la case il a gagné le jeu ✓ • pièges : quand un personnage arrive sur la case il subit des dégâts ✗ • magiques : si un personnage arrive sur la case un effet est déclenché ✓ • passages : un personnage qui arrive sur la case est téléporté à un autre endroit 2. ✓

3. Monstres

1) Des monstres sont placés de manière aléatoire dans le labyrinthe. ✗

2) Les monstres se déplacent de manière aléatoire. ✓

3) Les monstres se déplacent de manière intelligente en essayant d'attraper le héros. ✗

4) Les fantômes sont des monstres qui peuvent traverser les murs. ✓

4. Attaques

1) Le héros est tué au contact d'un monstre. ✓

2) Le héros peut attaquer les montres avec lequel il est en contact. ✓

3) Le héros peut attaquer les montres sur la case adjacente ✓

Merci de votre attention

Test du jeu