

# Rapport du projet JAVA

Ecrit par Anojh Nageswaran et Rajibane Gunaratnam

# Table des matières

<b>I - Introduction.....</b>	<b>3</b>
<b>II - Le jeu.....</b>	<b>4</b>
<b>III - Les fonctionnalités du jeu.....</b>	<b>6</b>
<b>IV - UML.....</b>	<b>7</b>
A - Diagramme des cas d'utilisation.....	7
B - Diagramme de classe.....	8
C - Diagramme d'activité.....	10
<b>V - Organisation.....</b>	<b>11</b>
A - Gantt Project.....	11
B - Github.....	12
<b>VI - Les solutions.....</b>	<b>12</b>
A - Gestion du menu.....	12
B - Le déplacement.....	13
C - Collision.....	14
D - Le saut.....	14
E - Les fonctionnalités à faire.....	15
<b>VII - Conclusion.....</b>	<b>17</b>

## I - Introduction

Dans le cadre du module Algorithmique et Programmation du licence professionnelle ASCID, nous sommes assignés à réaliser un projet en JAVA. Dans lequel, nous allons être évalués sur la partie de la conceptualisation et de l'implémentation. Pour cela, nous suivons des cours d'UML pour la partie conceptualisation et nous suivons également les cours de JAVA pour la partie implémentation. Ce projet va également nous permettre de travailler en équipe et de nous mettre à rude épreuve dans les conditions et dans la peau d'un analyste, concepteur et développeur, qui est l'objectif du cursus.

Étant de grands amateurs de jeux vidéos, nous avons décidé de concevoir et de réaliser un jeu de plateforme assez connu type "Mario", en reprenant le même univers, les mêmes personnages. Nous avons déjà réalisé des jeux sur java, comme un MMORPG ou encore un jeu d'échecs ensemble. Ce nouveau projet va nous permettre de travailler ensemble, en méthode agile avec l'utilisation de git et de github pour une meilleure gestion de projet.

Tout au long de ce rapport, nous allons vous expliquer en détail les principes du jeu, puis la conception, l'organisation, la programmation et les difficultés rencontrés du projet.

## II - Le jeu

Ce projet consiste donc à réaliser un jeu de plateforme en 2D reprenant l'univers de Mario, où vous allez retrouver le célèbre personnage de Nintendo Mario pour des aventures au sein du Conservatoire National des Arts et Métiers. Le principe du jeu est très simple, il y aura plusieurs niveaux, dans lesquels, le personnage commence d'un point de départ et devra se rendre au point d'arrivée dans un temps imparti et en collectant toutes les pièces d'or. Il rencontrera cependant pas mal d'embûches sur son chemin.

\*photos de croquis à ajouter pour expliquer le principe.

Au-delà, de l'aspect jeu de plateforme avec un point de départ et point d'arrivée, nous souhaitons raconter une histoire à travers plusieurs niveaux. Effectivement, installer un scénario est une volonté de notre équipe, afin de faire vivre le jeu. Le scénario est un grand classique des jeux vidéo de ce type donc voici le synopsis :

*Dans le royaume enchanté d'Euphoria, la princesse Arielle est enlevée par le sorcier maléfique Anojh du royaume de CNAM. Le roi et la reine, désespérés, font appel au célèbre plombier Mario, pour sauver leur fille bien-aimée.*

*Mario est un plombier très courageux, et très expérimenté, qui a déjà combattu de vrais ennemis auparavant.. Il accepte donc cette mission dangereuse pour sauver la princesse Arielle et prouver sa bravoure au royaume d'Euphoria.*

Le joueur incarne donc Mario et doit naviguer à travers différents niveaux, chacun rempli de défis et d'ennemis qui lui barreront la route. Il doit vaincre les sbires d'Anojh, et trouver le moyen de sauver la princesse Arielle.

Il y aura donc des personnages secondaire et principale dans le jeu :

	<p>Mario est le personnage principal de cette page. Son objectif est d'éviter les obstacles et les monstres et récupérer des pièces trouvables sur son chemin pour atteindre le drapeau final qui désigne la fin du niveau. Il peut sauter sur des obstacles pour les éviter ou sur les montres pour les tuer. Mario peut aller à droite ou à gauche mais s'il touche un monstre alors il meurt.</p>
---	--

	<p>La princesse Arielle est le personnage secondaire de ce jeu car elle est le cœur de l'histoire. Son but est d'être sauvé par Mario.</p>
	<p>Le sorcier Anojh est le méchant final de ce jeu.</p>
	<p>Goomba, un monstre à la forme d'un champignon qui a été invoqué par le sorcier Anojh. Son objectif est de tuer Mario en le touchant. Il marche de droite à gauche sans arrêt.</p>
	<p>La tortue est un monstre invoqué par le méchant de ce jeu et il a pour but de tuer Mario. Il marche de droite à gauche sans arrêt mais s'il meurt, il laisse sa carapace pour qui peut tuer Mario et les autres monstres.</p>

Il y aura donc plusieurs niveaux, et plus on avance dans les niveaux, plus ces derniers seront durs, le but étant d'avoir du challenge et que le jeu a une bonne durée de vie. Notamment, avec des monstres plus laborieux à vaincre et un parcours de plus en plus complexe.

### III - Les fonctionnalités du jeu

Le jeu comporte plusieurs fonctionnalités qui répondent à des problématiques évidentes qui seront essentielles à la conception et au développement de ce dernier. Nous allons d'abord s'attarder sur la gestion des niveaux, qui est une fonctionnalité importante dans notre projet, car il sera le fil conducteur du scénario. Pour cette fonctionnalité, chaque niveau aura un environnement différent, des plateformes différentes, et des monstres différents, plus on avance dans le jeu, plus ce dernier devient compliqué à réussir.

Le joueur doit avoir la possibilité de revenir sur un jeu déjà entamé et retrouver sa progression, il faut donc gérer la sauvegarde. Pour cela, nous voulons réaliser des sauvegardes automatiques à chaque fin de niveau, une sauvegarde réalisée à partir des fichiers json ou bien .ini qui permettront de stocker la progression et l'utiliser.

Dans chaque niveau, le joueur doit récupérer un certain nombre de pièces dans un temps imparti afin de valider le niveau et débloquer le niveau suivant, le score sera affiché sur l'écran, il y aura donc une gestion du score.

Le jeu est une superposition de plusieurs images qu'on fait bouger ou non, et comme dans un film d'animation, c'est plusieurs images qui s'enchaînent sur 1 milliseconde. C'est exactement le même mécanisme qu'on va adopter pour notre projet, c'est rafraîchir l'image tous les 3ms, pour par exemple permettre de faire déplacer Mario.

Pour le déplacement, nous allons devoir faire de la gestion du clavier, en effet le joueur va pouvoir interagir Mario le personnage qu'il incarne avec les flèches directionnelles du clavier. Les déplacements sont les suivants : la flèche de droite pour diriger Mario vers la droite, La flèche gauche pour diriger Mario vers la gauche, et la flèche du haut pour faire sauter Mario. Mario est le seul personnage que le joueur pourra contrôler, les autres personnages sont des "png", autrement dit personnage non joueur. Ces personnages non joueurs (champignons, tortues), à défaut de ne pas être contrôlé par le joueur, auront un déplacement qui sera programmé et auront une position précise dans la map par rapport aux niveaux. Cependant, tous ces personnages qu'il soit non joueur ou pas, on devra gérer leurs collisions et les contacts entre eux et également entre les objets quelconques.

Tous ces éléments cités ci-dessus, sont importants pour passer à une partie centrale de la bonne conduite d'un projet, c'est la conception.

### IV - UML

Toutes les informations données dans le dernier point vont nous permettre de réaliser la gestion de projet, c'est-à-dire la conception du jeu. Pour cela, la réalisation du diagramme de cas d'utilisation est indispensable, pour une meilleure compréhension du sujet.

### A - Diagramme des cas d'utilisation

Voici une première version du diagramme de cas d'utilisation

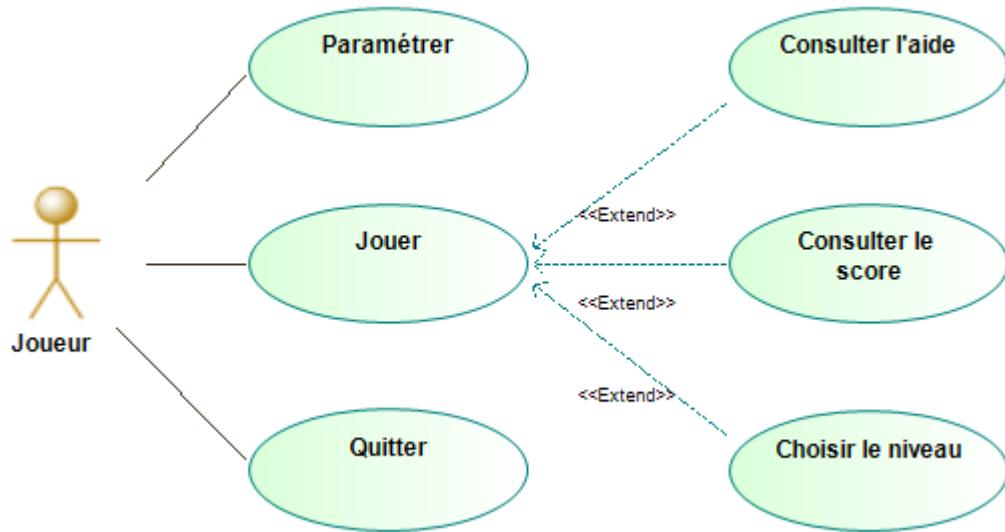


Figure 1.1 - Diagramme des cas d'utilisation

Dès le lancement du jeu, le joueur peut faire un choix entre jouer, paramétriser le jeu ou quitter le jeu mais s'il choisit de jouer alors il peut alors consulter le score qu'il avait effectué, consulter l'aide et choisir le niveau.

### B - Diagramme de classe

Nous avons ensuite le diagramme de classe



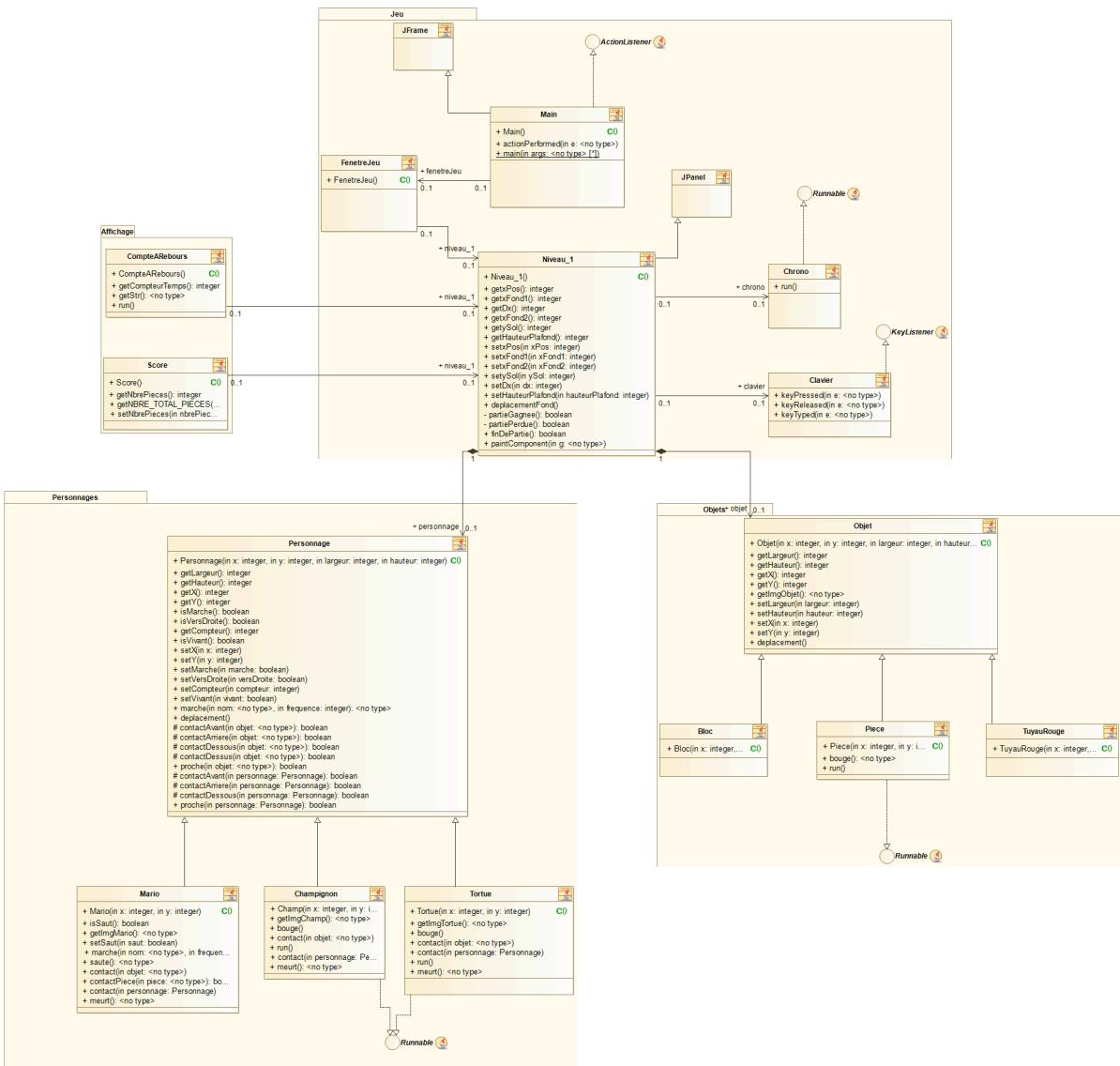
Figure 1.2 - Diagramme de classe

Nous avons créé 4 packages qui sont :

- Le jeu
  - Personnages
  - Objet
  - Score/Timer

Dans chaque package, nous avons une super-classe qui sont soit composés ou agréés de classe sauf pour le package Score/timer, il n'y a pas de super-classe mais que des petites classes.

Voici la version final du diagramme des classe :



On a ajouté trois classes en plus :

- La classe JFrame
  - La classe JPanel
  - La classe Sauvegarde

### C - Diagramme d'activité



Figure 1.3 - Diagramme d'activité

Le diagramme d'activité permet de voir comment le système réagit en fonction des choix qu'on effectue.

## V - Organisation

L'IDE que nous avons utilisé pour ce projet est Eclipse et nous avons programmé le jeu avec la librairie JFrame.

### A - Gantt Project

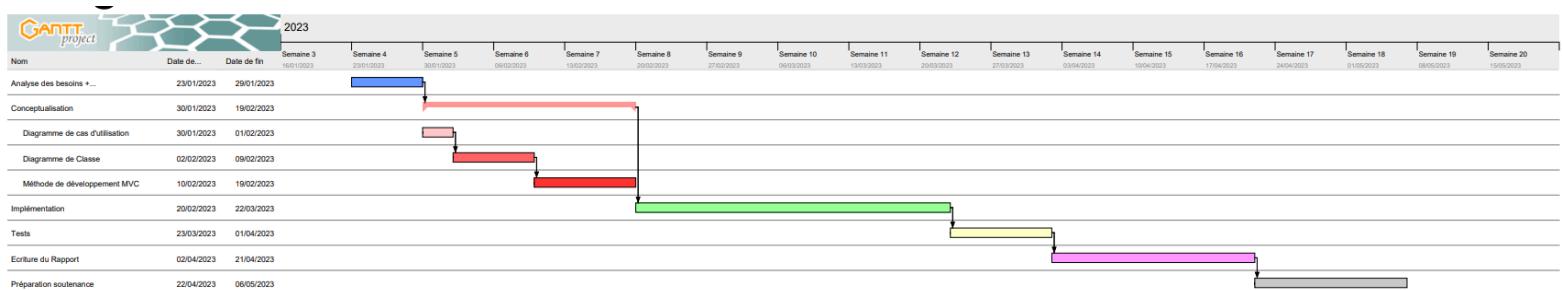


Figure 2.1 - Diagramme de Gantt

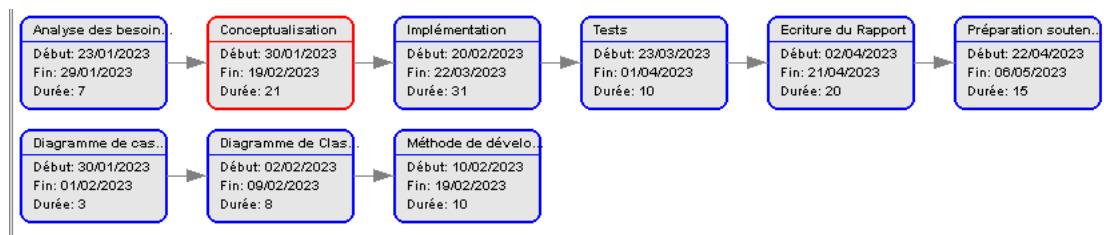


Figure 2.1 - Diagramme de Pert

Nous avons divisé les deux parties : la première partie est la rédaction d'un cahier de charge et la deuxième partie est la conception du jeu.

La première partie est nous l'avons divisée en deux parties : l'UML et la rédaction. Rajibane s'occupait des diagrammes de cas d'utilisation et de classe. Anojh s'occupait du diagramme d'activité. Après avoir terminé le cahier des charges, nous avons commencé à conceptualiser le jeu.

## B - Github

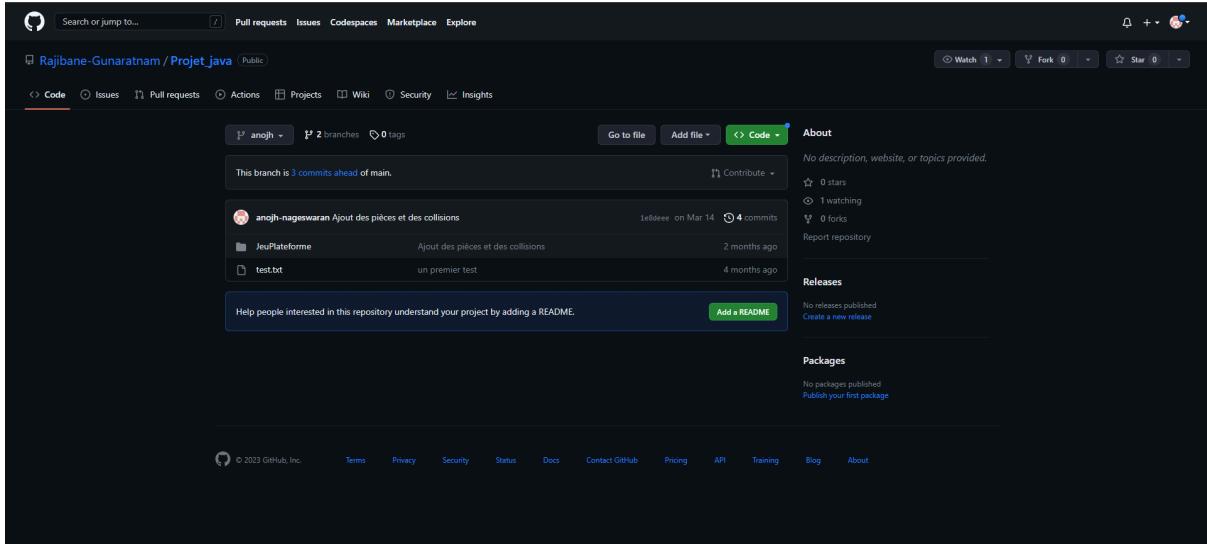


Figure 2.2 - Image de notre Github

Github permet que nous puissions mettre nos projets dans un répertoire en ligne qui peut être en public ou en privé. Nous avons créé le répertoire “Projet\_Java” avec deux branches “main” et “anojh” (Figure 2.2) car Rajibane mettait ses travaux dans main.

## VI - Les solutions

Nous allons donc vous expliquer, les fonctions conçues pour répondre aux différentes problématiques qu'on a soulevées lors des points précédents.

La première étape était de générer un fond d'un niveau, puis gérer les déplacements de Mario, pour cela nous avons été amenés à réaliser plusieurs fonctions. En effet, la génération de fond du niveau et les déplacements de mario sont étroitement lié, car le fond est une image en 700 \* 360 (la taille de la fenêtre), qu'on dessine grâce à la fonction paintComponent de la bibliothèque swing, et mario est également une image qu'on dessine sur le panel. Cependant, pour donner l'impression que Mario se déplace, nous allons faire déplacer le fond derrière lui. Pour cela, nous avons réalisé un flux indépendant, un programme qui va s'exécuter en tâche de fond, donc un thread qui va repeindre avec la fonction repaint() à une certaine fréquence l'écran. Cela va donner l'impression que Mario se déplace

## A - Gestion du menu

Pour la réalisation du menu qui n'est pas totalement fini, on a utilisé les JPanel(), pour cela on a un JPanel qui est le main et deux autres JPanel() représentant le menuPrincipal et menuNiveaux. Le but est d'afficher les deux panels dans le mainPanel par un bouton.

```
// Création des panneaux et des boutons

mainPanel = new JPanel();

MenuPrincipal = new JPanel();

MenuNiveau = new JPanel();

BoutonRetour = new JButton("retour");

BoutonJouer = new JButton("Jouer");

BoutonNiveau1 = new JButton("1");

// Ajout des boutons aux panneaux et configuration des listeners

//MenuPrincipal.setLayout(new BorderLayout());

BoutonJouer.setFont(new Font("Century", Font.BOLD, 26));

MenuPrincipal.add(BoutonJouer);

BoutonJouer.addActionListener(this);

MenuNiveau.add(BoutonRetour);

BoutonRetour.addActionListener(this);

MenuNiveau.add(BoutonNiveau1);

BoutonNiveau1.addActionListener(this);

// Configuration du panneau principal

mainPanel.setLayout(new CardLayout());

mainPanel.add(MenuPrincipal, "MenuPrincipal");

mainPanel.add(MenuNiveau, "MenuNiveau");

// Configuration de la fenêtre principale
```

```
        setContentPane(mainPanel);

        setTitle("Mario");

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setSize(700, 360);

        setLocationRelativeTo(null);

        setVisible(true);

    }

//*****METHODES*****
*****



    public void actionPerformed(ActionEvent e) {

        // Si le bouton 1 est cliqué, activez le panneau 1

        if (e.getSource() == BoutonRetour) {

            CardLayout Card = (CardLayout) (mainPanel.getLayout());

            Card.show(mainPanel, "MenuPrincipal");

        }

        // Si le bouton 2 est cliqué, activez le panneau 2

        else if (e.getSource() == BoutonJouer) {

            CardLayout Card = (CardLayout) (mainPanel.getLayout());

            Card.show(mainPanel, "MenuNiveau");

        }

        else if (e.getSource() == BoutonNiveaul) {

            fenetrejeu = new FenetreJeu();

            fenetrejeu.setVisible(true);

            dispose();

        }

    }

}
```

## B - Le déplacement

Le déplacement de mario que cela soit vers la gauche ou la droite est une succession de plusieurs images à la suite à une fréquence donnée, qui permet de créer l'animation. Pour cela nous utiliserons la méthode marche qui est une méthode de super classe Personnage, donc c'est une fonctionnalité qui est utilisée pour mario mais également les autres personnages de la plateforme.



```
public Image marche(String nom, int fréquence) {  
  
    String str;  
    ImageIcon ico;  
    Image img;  
  
    if (this.marche == false) {  
  
        if(this.versDroite == true) {str = "/images/" + nom + "ArretDroite.png";}  
        else {str = "/images/" + nom + "ArretGauche.png";}  
    } else {  
  
        this.compteur++;  
  
        if(this.compteur / fréquence == 0) {  
  
            if(this.versDroite == true) {str = "/images/" + nom + "ArretDroite.png";}  
            else {str = "/images/" + nom + "ArretGauche.png";}  
        } else {  
  
            if(this.versDroite == true) {str = "/images/" + nom + "MarcheDroite.png";}  
            else {str = "/images/" + nom + "MarcheGauche.png";}  
        }  
  
        if (this.compteur == 2 * fréquence) {this.compteur = 0;}  
    }  
  
    //Affichage du personnage  
  
    ico = new ImageIcon(getClass().getResource(str));  
    img = ico.getImage();  
  
    return img;  
}
```

## C - Collisions

Après la création d'objet dans la carte, il a fallu gérer les contact et les collision avec les objets, pour cela on a créé une méthode contactAvant(), qui va vérifier à chaque instant dès que la position X de Mario et la hauteur en Y de Mario rentre en contact a une position X du bord du tuyau rouge il renverra True. Pour cela dans la méthode on vérifie tous les cas ou il n'y a pas contact. Cela va nous permettre de gérer facilement le cas lorsqu'il y a contact.

```
protected boolean contactAvant(Objet objet){  
  
    if(this.isVersDroite() == true){  
  
        if(this.x + this.largeur < objet.getX() || this.x + this.largeur > objet.getX() + 5  
        || this.y + this.hauteur <= objet.getY() ||  
  
            this.y >= objet.getY() + objet.getHauteur()){return false;}  
  
        else{return true;}  
  
    }else{return false;}  
}
```

Donc si les vérification des cas lorsqu'il y a contact sont false, alors il y a contact avant. On réalise 2 autres méthodes pour les contactDessous() et contactArriere(). A partir on réalise une méthode générale contact() qui englobe tous ses contact au niveau de la classe Mario, et de là nous allons sur la classe Niveau\_1 et on traite ça dans le paintComponent():

```
for(int i = 0; i < this.tabObjets.size(); i++) {  
  
    if(this.mario.proche(this.tabObjets.get(i)))  
    {this.mario.contact(this.tabObjets.get(i));}  
}
```

La méthode proche va permettre de savoir la proximité à 10 pixels près en position X et en position Y entre chaque personnage et objets présents sur la plateforme afin de gérer au mieux les contacts.

```
public void contact(Objet objet) {  
  
    if((super.contactAvant(objet) == true && this.isVersDroite() == true)  
    || (super.contactArriere(objet) == true && this.isVersDroite() == false)){  
  
        FenetreJeu.niveau_1.setDx(0);  
  
        this.setMarche(false);  
    }  
}
```

Et lorsqu'il y a contact avec un objet, mario arrête de marcher. De plus, nous avons également gérer les collisions et les contacts entre png et objet, et png et Mario, dans ce cas de figure Mario meurt.

## D - Le saut

Après l'insertion des collisions avec les objets pour Mario, nous avons commencé à implémenter son saut pour qu'il puisse gravir des objets et récupérer les pièces. Pour cela, on a initialisé une variable qui permet de régler la limite de l'hauteur et de la durée du saut que l'on a nommé "compteurSaut". Si la tête de Mario ne touche pas le haut de la carte alors on soustrait la position en Y de Mario de 4 pour le faire monter sinon il ne sautera pas car on bloque le compteur saut. Si Mario saute vers la droite, on affiche son image qui saute vers la droite sinon on affiche son image qui saute vers la gauche. Pour la retombée de Mario, si le bas de l'image de Mario n'est pas au-dessus du sol de la carte alors son image se rapprochera très lentement du bas de la carte. Quand Mario a terminé son saut, on affiche l'image de Mario à l'arrêt du côté droit ou gauche et on met la variable saut en false car Mario ne saute pas et le "compteurSaut" à 0. A la fin de la méthode, on initialise l'affichage des images de Mario.

```
public Image saute() {  
    ImageIcon ico;  
    Image img;  
    String str;  
  
    this.compteurSaut++;  
  
    // Montée du Saut  
  
    if(this.compteurSaut <= 40) {  
  
        if(this.getY() > FenetreJeu.niveau_1.getHauteurPlafond()) {this.setY(this.getY() - 4);}  
        else {this.compteurSaut = 41;}  
  
        if(this.isVersDroite() == true) {str = "/images/marioSautDroite.png";}  
        else {str = "/images/marioSautGauche.png";}  
  
        //Retombée du saut  
  
    } else if (this.getY() + this.getHauteur() < FenetreJeu.niveau_1.getySol()) {this.setY(this.getY() + 1);  
  
        if (this.isVersDroite() == true) {str = "/images/marioSautDroite.png";}  
        else {str = "/images/marioSautGauche.png";}  
  
        //Saut terminé  
  
    } else {  
  
        if(this.isVersDroite() == true) {str = "/images/marioArretDroite.png";}  
        else {str = "/images/marioArretGauche.png";}  
  
        this.saut= false;  
        this.compteurSaut = 0;  
    }  
  
    ico = new ImageIcon(getClass().getResource(str));  
    img = ico.getImage();  
  
    return img;  
}
```

## E - Les fonctionnalités à faire

Dans les fonctionnalités à implémenter, nous avons la sauvegarde, les niveaux et le scénario du jeu. La sauvegarde permet de mettre en sécurité l'avancée de l'utilisateur dans le jeu. Nous avons d'abord décidé d'implémenter une sauvegarde manuelle dans le jeu mais nous nous sommes dit qu'on devait la remplacer par une sauvegarde automatique. Par rapport aux niveaux, nous avons imaginé de mettre à chaque fin et à chaque début d'un niveau un panel qui affiche un bout du scénario.

Malheureusement, nous n'avons pas pu aboutir à 100% la totalité du projet au niveau de l'implémentation, dans un délai de 4 mois environ. En effet, plusieurs fonctionnalités n'ont pas été implémentées pour aboutir au projet et la cause est le manque de temps, car on a eu les yeux plus gros que le ventre. Les fonctionnalités manquantes sont les suivantes :

- La gestion de la sauvegarde, une partie intéressante à implémenter et également très importante pour un jeu vidéo. L'idée était de réaliser des sauvegardes automatiques après la fin de chaque niveau, par exemple, lors du lancement d'un nouveau jeu, un fichier ini est créé sur la machine avec des paramètres comme représenté ici. Puis, à chaque fois que le joueur termine un niveau on va modifier ce fichier, et si le joueur décide de continuer sa partie plus tard et quitte la fenêtre ce n'est pas un souci. Car dès lors qu'il relance le jeu et clique sur "charger partie", le programme va lire ce fichier et il pourra reprendre la partie là où il s'était arrêté.
- La gestion des niveaux, effectivement nous n'avons pu réaliser qu'un seul niveau dans le temps imparti, mais le but était de rendre le jeu de plus en plus dur, et également de changer de décors, de monstres, d'amener différents univers, atmosphère pour chaque niveau.
- La gestion des cinématiques, c'est une fonctionnalité bonus, qui aurait pu être très plaisante pour le joueur. En effet, le but était d'ouvrir une fenêtre pop-up après chaque niveau, qui à travers des images successive et du texte raconte un peu les aventures de Mario après chaque niveau voici quelques exemples :

```
[Paramètre]
Niveau1 = 1
Niveau2 = 1
Niveau3 = 0
Niveau4 = 0
JeuTerminé = 0
JeuRecommencé = 0
Etat_Avancement = 50%
```



*“La princesse Arielle est en train de se promener dans les jardins du château d'Euphoria, entourée de fleurs et d'oiseaux colorés. Elle est vêtue d'une magnifique robe de bal et porte une couronne de diamants sur la tête”*



*“Soudain, le ciel devient sombre et un éclair frappe le sol, laissant apparaître le sorcier maléfique Anojh et ses sbires.”*



*“Mario est représenté en train de sortir de son tuyau de plomberie dans le château d'Euphoria. Il est habillé de sa célèbre salopette rouge et porte sa casquette verte. Il est prêt pour l'action et déterminé à sauver la princesse Arielle.”*

## VII - Conclusion

Globalement, le module “Programmation JAVA” nous a permis d’améliorer nos compétences de programmation et notre créativité. La réalisation de ce jeu nous a permis de faire sa conception de A à Z. Malgré les difficultés rencontrées, nous avons pu faire l’essentiel pour gérer le temps et développer les bases fonctionnelles de notre projet.

Techniquement, on a pu utiliser tout ce qu’on a acquis durant cette année de licence professionnelle au CNAM comme la modélisation UML. L’utilisation de JAVA nous a permis de savoir qu’avec ce langage et son IDE Eclipse nous pouvons l’utiliser pour programmer nos envies qui sont les jeux vidéo. Nous avons mis en œuvre nos compétences de la méthode d’Agile Scrum que nous avons acquises durant ces deux dernières années.

Pour notre futur parcours, nous avons décidé de rejoindre un Master MIAGE pour l’année prochaine