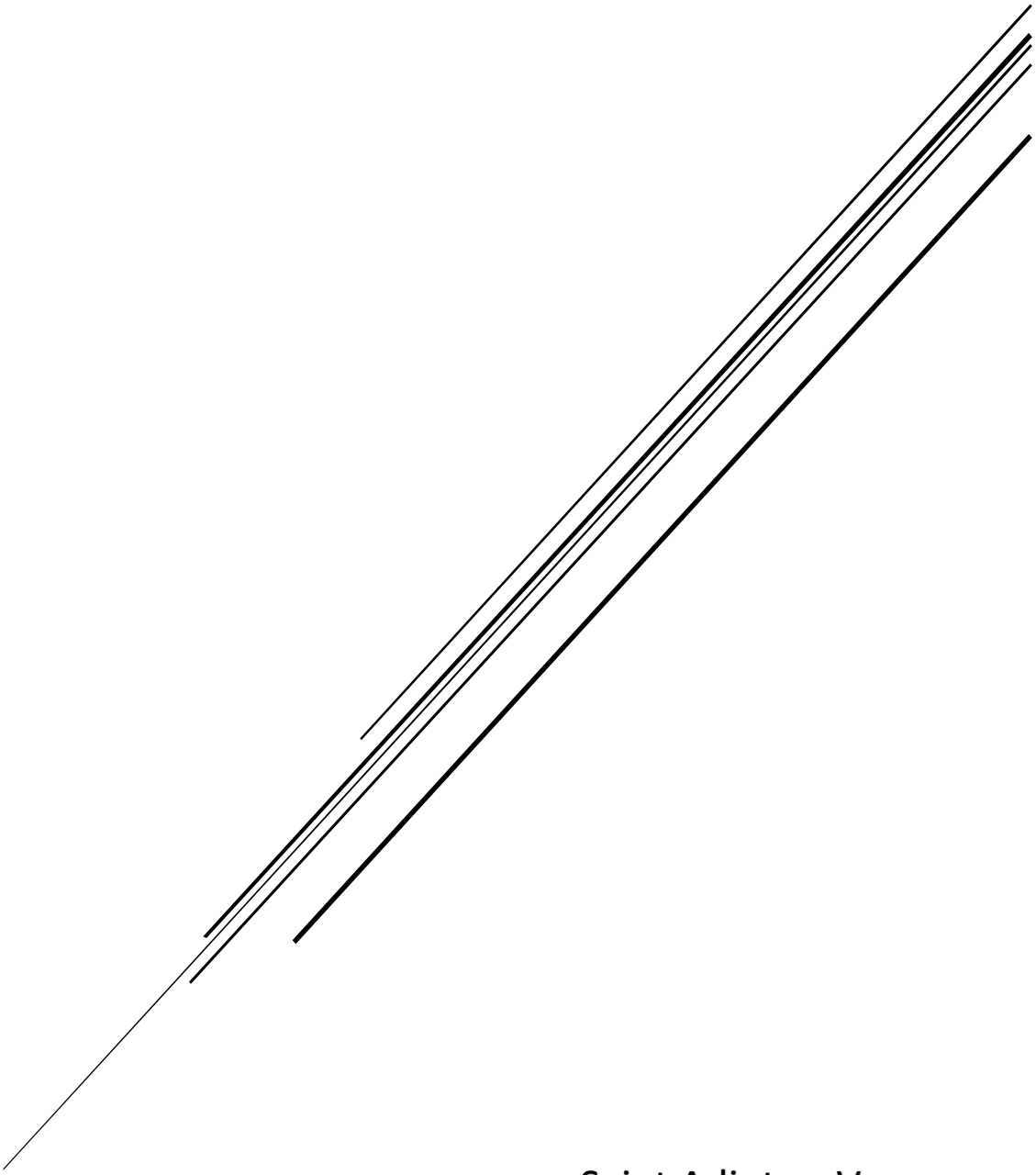
# Dossier de programmation

DP Numéro 1



Saint Adjutor, Vernon

Dossier de programmation N°1

# Sommaire :

# 1. Présentation du dossier

## 1.1. Présentation globale

Le dossier de programmation qui m'a été confié avait pour but de réaliser un jeu en langage C#. Ce jeu doit contenir une interface graphique afin de demander au joueur son pseudonyme pour pouvoir afficher son score en fin de partie. Le personnage doit être capable de se déplacer horizontalement et verticalement sur le plateau. Une fois arrivé à une extrémité, le personnage doit disparaître et réapparaître de l'autre côté du plateau. Ce jeu doit également contenir une classe celle-ci contenant les propriétés du personnage comme sa taille (longueur et hauteur), son nom, sa position et plus. On a ensuite le choix entre 3 propositions de projet supplémentaire :

- Un deuxième personnage se déplace aléatoirement et à une fréquence définie sur le carte. Le premier personnage doit attraper le second pour gagner des points.
- Un deuxième personnage apparait à des coordonnées aléatoires. Le premier doit attraper le second et une fois attrapé un second apparait et le joueur gagne des points.
- Plusieurs personnages apparaissent à des positions aléatoires et doivent être attrapés dans un temps limité. Les points augmentent dès qu'un personnage est attrapé.

## 1.2. Mon projet

J'ai choisi de réaliser la troisième option. J'ai réalisé ma une form de connexion demandant le pseudonyme du joueur et activant le bouton « JOUER ». Une seconde form se lance ensuite lors de l'appui du bouton. Mon projet est inspiré du jeu PAC MAN, le personnage principal (PAC MAN) apparait au milieu du plateau et son but est d'attraper les 4 fantômes. Le joueur à une minute pour attraper 25 fantômes et gagner la partie.

## 1.3. Liste des compétences

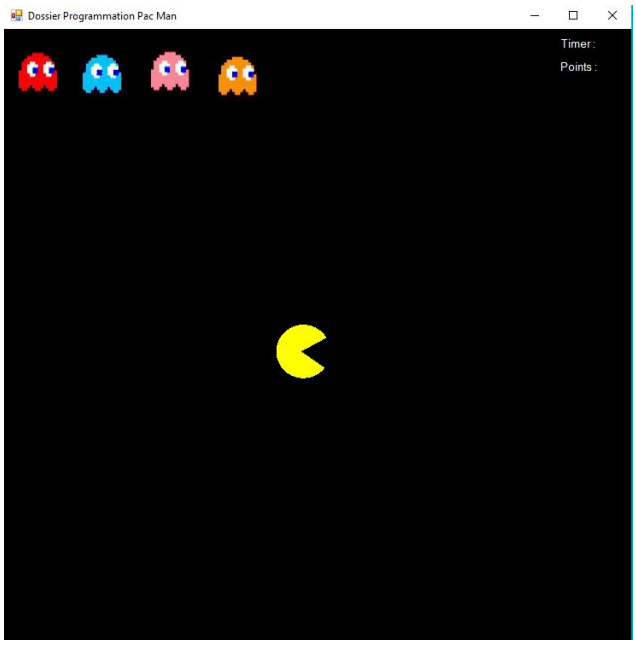
Les compétences associées sont les suivantes :

- A4.1.6 Gestion d'environnements de développement et de test
- A4.1.7 Développement, utilisation et adaptation de composants logiciels
- A4.1.8 Réalisation des tests nécessaires à la validation d'éléments adaptés ou développés
- A4.1.9 Rédaction d'une documentation technique

# 2. Structures de données

## 2.1. Structure globale

Le dossier de programmation est composé de 2 forms, une de menu et une composée du jeu. Il y a aussi 2 classes, une pour le personnage principal (PAC MAN) et une pour les ennemis (les fantômes). On retrouve également dans le dossier « debug », les images nécessaires à mon projet. On y retrouve donc un gif de PAC MAN dans les 4 directions (haut, bas, droite et gauche) ainsi qu'un gif pour chacun des 4 fantômes qui constituent le jeu original. On retrouve peu d'éléments graphiques Windows Form, seulement un bouton pour jouer, un TextBox pour le nom du joueur et quelques labels pour les différents affichages (« Objectif », « Timer » et « Points »). On retrouve également 5 PictureBox pour les 5 personnages et une ImageList pour les images de PAC MAN et un Timer.

## 2.1. Classes

Mon projet dispose de 2 classes reprenant la même structure de base :

```
3 références
class Personnage_Gentil
{
    private string nom;             //Nom du personnage
    private int widht;              //Largeur du personnage
    private int height;             //Hauteur du personnage
    private int X;                  //Position sur X
    private int Y;                  //Postition sur Y
    private int points;             //Nombres de points
    private PictureBox nomPicture;  //Nom de la pb contenant le personnage
```

La classe Personnage gentil contient un attribut privé vitesse qui permet de définir la vitesse à laquelle le personnage va se déplacer sur la carte. Les ennemis ne bougeant pas ils n'ont pas besoin de l'attribut vitesse. On a ensuite getter afin de localiser le personnage sur la grille.

```
//Accesseurs
0 références
public int getPositionGentilX() { return X; }
0 références
public int getPositionGentilY() { return Y; }
```

On a ensuite dans la classe des setters pour faire bouger le personnage et se téléporter :

```
//Mutateurs
//Mouvements
1 référence
public void setAllerDroite() { this.X = X + vitesse; }
1 référence
public void setAllerGauche() { this.X = X - vitesse; }
1 référence
public void setAllerBas() { this.Y = Y + vitesse; }
1 référence
public void setAllerHaut() { this.Y = Y - vitesse; }

//Teleportation
1 référence
public void setTpHaut() { this.Y = -49; this.X = nomPicture.Location.X; }
1 référence
public void setTpBas() { this.Y = 689; this.X = nomPicture.Location.X; }
1 référence
public void setTpDroite() { this.X = 739; this.Y = nomPicture.Location.Y; }
1 référence
public void setTpGauche() { this.X = -49; this.Y = nomPicture.Location.Y; }
```

# 3. Algorithmes utilisés

## 3.1. Déplacements

L'algorithme de déplacement est assez simple. Se déplacer signifie simplement ajouter à la position du personnage en X ou Y le nombre de pixel définit par l'attribut privé vitesse. Pour le déclenchement du setter, j'utilise la méthode KeyDown. Lorsqu'une touche est enfoncée la boucle « if » va s'enclencher. L'origine de l'image et de la form se trouve en haut à gauche donc le 0, 0. Pour aller à droite on ajoute donc la position de la PictureBox du personnage à la vitesse, à gauche on retire la vitesse, en haut on retire la vitesse et enfin en bas on ajoute la vitesse. On actualise la position du personnage après chaque déplacement pour pouvoir le suivre en temps réel et mettre l'image correspondant à sa direction.

```
//Déplacement du personnage
//Aller en haut
if (e.KeyCode == Keys.Up)
{
    PacMan.setAllerHaut();
    pb_gentil.Image = ilPacMan.Images[0];
    PacMan.ActualiserGentil();
}
//Aller à droite
if (e.KeyCode == Keys.Right)
{
    PacMan.setAllerDroite();
    pb_gentil.Image = ilPacMan.Images[1];
    PacMan.ActualiserGentil();
}
//Aller à gauche
if (e.KeyCode == Keys.Down)
{
    PacMan.setAllerBas();
    pb_gentil.Image = ilPacMan.Images[2];
    PacMan.ActualiserGentil();
}
//Aller en bas
if (e.KeyCode == Keys.Left)
{
    PacMan.setAllerGauche();
    pb_gentil.Image = ilPacMan.Images[3];
    PacMan.ActualiserGentil();
}
```

## 3.2. Téléportations

Pour pourvoir téléporter le personnage, il faut savoir à quelles coordonnées en X et Y la PictureBox contenant le personnage sors du domaine visible. Pour cela, j'ai utilisé un « Console.WriteLine(X) » dans le getter « Actualiser Personnage » pour savoir les coordonnées ou le personnage disparaît totalement et de même pour les coordonnées en Y. J'ai ensuite fait des boucles « if » de manière à dire que « lorsque les coordonnées en X ou Y dépassent les coordonnées relevées, on récupère la coordonnées qui ne change pas donc en X ou Y et on remet à la coordonnées de l'autre côté ». J'ai téléporté le personnage à un pixel avant les coordonnées de téléportation pour éviter de faire une boucle infinie.

```
//Teleportation du personnage
//Teleportation en haut
if (pb_gentil.Location.Y > 690)
{
    PacMan.setTpHaut();
}
//Teleportation en bas
if (pb_gentil.Location.Y < -50)
{
    PacMan.setTpBas();
}
//Teleportation à droite
if (pb_gentil.Location.X < -50)
{
    PacMan.setTpDroite();
}
//Teleportation à gauche
if (pb_gentil.Location.X > 740)
{
    PacMan.setTpGauche();
}
```

```
//Teleportation
1 référence
public void setTpHaut() { this.Y = -49; this.X = nomPicture.Location.X; }
1 référence
public void setTpBas() { this.Y = 689; this.X = nomPicture.Location.X; }
1 référence
public void setTpDroite() { this.X = 739; this.Y = nomPicture.Location.Y; }
1 référence
public void setTpGauche() { this.X = -49; this.Y = nomPicture.Location.Y; }
```

## 3.3. Génération des ennemis

Afin de générer un personnage, il faut déclarer un objet pour chaque ennemi voulu, ici j'en voulais 4.

```
Personnage_Mechant Blinky;
Personnage_Mechant Inky;
Personnage_Mechant Pinky;
Personnage_Mechant Pokey;
```

Ensuite, On doit instancier chaque objet.

```
Blinky = new Personnage_Mechant("Blinky", 64, 64, pbBlinky);
Inky = new Personnage_Mechant("Inky", 64, 64, pbInky);
Pinky = new Personnage_Mechant("Pinky", 64, 64, pbPinky);
Pokey = new Personnage_Mechant("Pokey", 64, 64, pbPokey);
```

Puis enfin on lui attribue des coordonnées grâce à une génération d'aléatoire puis à une génération aléatoire d'un nombre pour la coordonnée X et un autre pour la coordonnée Y.

```
//Position fantome
Blinky.setPositionMechantX();   //Positionnement aléatoire en X
Blinky.setPositionMechantY();   //Positionnement aléatoire en Y
Blinky.ActualiserMechant();     //Actualisation du personnage à sa nouvelle position

Inky.setPositionMechantX();
Inky.setPositionMechantY();
Inky.ActualiserMechant();

Pinky.setPositionMechantX();
Pinky.setPositionMechantY();
Pinky.ActualiserMechant();

Pokey.setPositionMechantX();
Pokey.setPositionMechantY();
Pokey.ActualiserMechant();
```

```
public void setPositionMechantX()   //Générer une coordonnée aléatoire sur X
{
    Random posX = new Random(Guid.NewGuid().GetHashCode()); //Génère un nouveau domaine d'aléatoire
    X = posX.Next(0, 680);  //Génère un nombre entre 0 et 680
}
8 références
public void setPositionMechantY()   //Générer une coordonnéé aléatoire sur Y
{
    Random posY = new Random(Guid.NewGuid().GetHashCode()); //Génère un nouveau domaine d'aléatoire
    Y = posY.Next(0, 620);  //Génère un nombre entre 0 et 620
}
```

## 3.4. Augmenter les points

Afin d'augmenter les points, il faut qu'un fantôme soit « attrapé », c'est-à-dire que PAC MAN doit être au-dessus du fantôme. Pour cela on détecte si l'entité qu'on souhaite attraper est présente sur la grille. Ensuite si le point au milieu de la PictureBox englobe la PictureBox du fantôme qu'on désire attraper alors on définit le fantôme à null et on cache la PictureBox. Ensuite on incrémente le score car un fantôme a été attrapé.

```
//Attraper un mechant
if (Blinky != null) //Si le méchant est sur la grille
{
    //Si le point au milieu de l'axe horizontal la pb_gentil contient celui de la pbBlinky
    if (pb_gentil.Location.X + pb_gentil.Width/2 > pbBlinky.Location.X && pb_gentil.Location.X < pbBlinky.Location.X + pbBlinky.Width)
    {
        //Et si le point au milieu de l'axe vertical de la pb-gentil contient celui de la pbBlinky
        if (pb_gentil.Location.Y + pb_gentil.Height/2 > pbBlinky.Location.Y && pb_gentil.Location.Y < pbBlinky.Location.Y + pbBlinky.Height)
        {
            Blinky = null;                      //Le méchant est null car "attrapé"
            pbBlinky.Visible = false;           //On cache ce méchant
            score++;                            //On incrémente le score
            lbPoint.Text = score.ToString();    //On affiche le score
        }
    }
}
```

Ensuite, on veut faire réapparaitre un fantôme sur le plateau. Pour cela, on doit instancier une nouvelle fois le fantôme qui a été attrapé. Comme on l'a vu si un fantôme est attrapé sa valeur est à null. Donc si un fantôme est à null alors on le ré-instancie, on appelle une nouvelle fois les setters pour lui définir des coordonnées aléatoires et on actualise le fantôme de façon à ce qu'il apparaisse sur le plateau. Enfin on doit rendre sa PictureBox visible afin de pouvoir enfin le voir apparaitre sur le plateau.

```
if (Blinky == null) //Si le personnage n'est plus sur la grille
{
    Blinky = new Personnage_Mechant("Blinky", 64, 64, pbBlinky); //On en instancie un noveau

    pbBlinky.Visible = true;    //On le rend visible

    Blinky.setPositionMechantX();   //On lui attribue des coordonnées aléatoires sur X et Y
    Blinky.setPositionMechantY();
    Blinky.ActualiserMechant();     //On actualise sa position pour le faire apparaître
}
```

## 3.5. Gestion du temps

Dans les règles que j'ai créées, une partie correspond à 60 secondes soit une minute. Afin de gérer la minute de jeu, j'ai inclus un Timer. Celui-ci se déclenche lors de l'apparition de la form à l'écran. Ce Timer dure une seconde soit 1000 millisecondes et à chacun des tick de ce Timer, la variable seconde est incrémentée. Ensuite cette même variable va servir à l'affichage du temps restant. Le temps restant est affiché en haut à droite de la form au-dessus des points. On actualise donc l'affichage à chaque seconde ecoulée.

```
secondes++; //A chaque tick on incrémente seconde de 1 (1000 ticks = 1 seconde)
lbTimer.Text = (60 - secondes).ToString();  //On affiche le temps restant
```

## 3.6. Gestion de la partie

Selon mes règles, la partie s'arrête dès que le joueur a réussi à marquer 25 points ou dès qu'une minute s'est écoulée. Pour les points, la méthode a déjà été vue à la rubrique **3.4 Augmenter les points**. Pour le temps, j'ai utilisé une identification avec pour base un booléen. Lorsque la variable seconde dépasse 60 le booléen passe alors à vrai. Dans ce cas un message expliquant que le joueur à perdu apparaît à l'écran et résume son nombre de points puis la form se ferme ramenant sur le menu. A l'inverse, si le joueur à cumulé 25 points et que le booléen est faux alors on informe le joueur qu'il a gagné et on affiche son temps.

```
if (secondes >= 60) //Si on dépasse une minute
{
    perdu = true;   //Le booléen perdu passe à vrai on entre dans la boucle au dessus
}
```

```
//Gestion du jeu
if (perdu) //Si le booléen perdu est vrai
{
    //On dit au joueur qu'il à perdu et son nombre de point
    MessageBox.Show("Perdu. Votre score est de : " + score + " points", "Fin de partie", MessageBoxButtons.OK, MessageBoxIcon.Information);
    this.Close();   //On ferme la fenêtre
}

//Si l'objectif est atteint
if (lbPoint.Text == "25" && !perdu)
{
    //On dit au joueur qu'il à gagné et en combien de temps
    MessageBox.Show("Gagné Votre score est de 25 et vous avez gagné en : " + secondes + " secondes", "Fin de partie", MessageBoxButtons.OK, MessageBoxIcon.Information);
    this.Close();   //On ferme la fenêtre
}
```

# 4. Jeu d'essai

## 4.1. Les différents essais effectués

| *Numéro de test* | *Scénario* | *Description* | *Résultat* |
|---|---|---|---|
| *1* | Ouverture | Ouverture de la form de menu | Succès |
| *2* | Fermeture | Fermeture de la form de menu | Succès |
| *3* | Affichage | Vérification de la disposition des éléments sur la form | Succès |
| *4* | Ouverture | Ouverture de la form de jeu avec début du Timer et score à 0 | Succès |
| *5* | Déplacement | Déplacement du personnage sur le plateau | Succès |
| *6* | Téléportation | Téléportation du personnage à la coordonnée opposée | Echec |
| *7* | Téléportation | Téléportation du personnage à la coordonnée opposée | Succès |
| *8* | Affichage | Affichage des ennemis sur le plateau | Succès |
| *9* | Disposition | Disposition aléatoire des ennemis sur le plateau | Echec |
| *10* | Disposition | Disposition aléatoire des ennemis sur le plateau | Echec |
| *11* | Disposition | Disposition aléatoire des ennemis sur le plateau | Succès |
| *12* | Attraper | Attraper un ennemi sur le plateau | Echec |
| *13* | Attraper | Attraper un ennemi sur le plateau | Echec |
| *14* | Attraper | Attraper un ennemi sur le plateau | Echec |
| *15* | Attraper | Attraper un ennemi sur le plateau | Succès |
| *16* | Réapparition | Réapparition de l'ennemi attrapé à un endroit aléatoire | Succès |
| *17* | Score | Augmentation du score lorsqu'un ennemi est attrapé | Succès |
| *18* | Victoire/Défaite | Affichage du message de victoire ou défaite et fermeture de la fenêtre pour revenir sur le menu | Succès |

# 5. Conclusion

## 5.1. Conclusion globale

Pour conclure sur ce premier dossier de programmation je dirais que celui-ci avait un niveau assez dur mais abordable dans les temps proposé (3 semaines). Certaines des fonctionnalités auraient pu être approfondies et d'autres ajoutées et ou modifiés comme par exemple les téléportations qui auraient pu être des bordures fixes et infranchissables mais cela aurait-été à l'inverse du sujet. Ce projet a été finit dans les temps avec de longs moments de recherches et de nombreuses, très nombreuses modifications. Mais le résultat atteint me satisfait grandement, je le trouve assez épuré et fonctionnel et je ne diagnostique pas de bugs apparents.

## 5.2. Améliorations possibles

Ce projet pourrait avoir quelques améliorations ou fonctionnalités supplémentaires comme :

- Une animation pour PAC MAN
- Une vitesse qui se conserve lorsque l'on change de direction pour un mouvement plus fluide et une jouabilité plus simple
- Un enregistrement des scores
- Une fonctionnalité de « leaderboard »
- Un mode 1 contre 1 ou un joueur commandera le fantôme et l'autre PAC MAN
- …