Sujet

Développer un projet en langage C# à l'aide d'une interface graphique et d'une classe Personnage. Cette classe Personnage représentera un personnage réutilisable dans n'importe quel jeu.

Cette classe possèdera les informations suivantes :

- Un nom

- Une image

- Ses positions sur la grille de jeu

- Sa taille (largeur et hauteur de l'image)

- Un nombre de points : 0 au départ, cette valeur augmente au fur et à mesure des actions du jeu

Ce personnage pourra opérer les actions suivantes :

- se déplacer : horizontalement à droite et à gauche, verticalement en haut et en bas, en diagonales (4 directions : haut droit-haut gauche-bas droit-bas gauche)

- augmenter son nombre de points en fonction d'une action

Version du jeu

Plusieurs personnages sont placés sur la grille à des positions aléatoires.

-Le premier personnage dispose d'un temps limité pour tous les attraper.

-A chaque personnage attrapé le nombre de points augmente.

-Le jeu se termine quand le nombre de points a atteint une valeur maximum (à définir) ou lorsque le temps de jeu est écoulé ou encore par fermeture de la fenêtre.

Algorithmes

```
if (Keyboard.IsKeyDown(Key.Up)) // Pour se diriger vers le haut
{
    Ghost.haut(5); ;
    Ghost.actualiserPersonnage();
}
if (Keyboard.IsKeyDown(Key.Down))// Pour se diriger vers le bas
{
    Ghost.bas(5); ;
    Ghost.actualiserPersonnage();
}
if (Keyboard.IsKeyDown(Key.Right))// Pour se diriger vers la droite
{
    Ghost.droit(5); ;
    Ghost.actualiserPersonnage();
}
if (Keyboard.IsKeyDown(Key.Left))// Pour se diriger vers la gauche
{
    Ghost.gauche(5); ;
    Ghost.actualiserPersonnage();
}
```

On détecte l'entrée du clavier, ainsi on appelle dans la classe Personnage en fonction de la direction saisie. Dans la classe personnage on met à jour la position x ou y de l'objet

```
1 référence
public void droit(int vitesse)
{
    this.x += vitesse;
    if (ispersonnage)this.AvatarPicture.Image = JeuPersonnage.Properties.Resources.RightGhost;
}

1 référence
public void gauche(int vitesse)
{
    this.x -= vitesse;
    if (ispersonnage)this.AvatarPicture.Image = JeuPersonnage.Properties.Resources.LeftGhost;
}

1 référence
public void haut(int vitesse)
{
    this.y -= vitesse;
    if (ispersonnage) this.AvatarPicture.Image = JeuPersonnage.Properties.Resources.TopGhost;
}

1 référence
public void bas(int vitesse)
{
    this.y += vitesse;
    if (ispersonnage) this.AvatarPicture.Image = JeuPersonnage.Properties.Resources.BottomGhost;
}
```

```
    ///Test de position de deux Objet, retourne Vrai si le 1er objet est sur le 2eme
3 références
public static bool testMiam(Personnage Mangeur, Personnage Manger)
{
    if ((Manger.getX() >= Mangeur.getX() + (Mangeur.getW())/2) || (Manger.getY() + (Manger.getH()/2) <= Mangeur.getY())
        || (Manger.getY() >= Mangeur.getY() + (Mangeur.getH())/2) || (Manger.getX() + (Manger.getW()/2) <= Mangeur.getX()))
        return false;
    else return true;

}
```

```
8 références
public int getX()
{
    return this.x;
}

8 références
public int getY()
{
    return this.y;
}

4 références
public int getW()
{
    return this.width;
}

4 références
public int getH()
{
    return this.height;
}
```

On test le position le l'objet au-dessus (Mangeur) par rapport à l'objet en dessous (Manger) grâce au coordonnées x,y et size h,w retourné grâce au get de la classe personnage. Ainsi si les objets sont les un au-dessus des autres alors on retourne vrai.

Ainsi on définit à nouveau les coordonnées de l'objet numéro i, l'actualise et on augmente de 1 les points de l'objet goths.

```
//  Si l'objet n°i se trouve en dessous de l'objet Ghost alors on replace l'objet i aléatoirement sur la form puis on augmente les points et redessine l'objet
if (i != 0)
{
    Objet[i].setX(aleatoireX.Next(62, 732));
    Objet[i].setY(aleatoireY.Next(62, 380));
    Objet[i].actualiserPersonnage();
    Ghost.setPoints();
    Objet[i].getPb().Region = new System.Drawing.Region(Transparent(Objet[i].getPb().Image));
    Objet[i].getPb().BringToFront();
}
```

Jeu d'essai

Démarrage de l'application Aucune erreur ne doit apparaitre, la fenêtre doit s'ouvrir	Fonctionne, la fenêtre s'ouvre comme elle le doit
Boutons menu Aucune erreur ne doit apparaitre, quitter doit fermer la fenêtre et Jouer en ouvrir une nouvelle	Bouton Jouer et Bouton Quitter sont fonctionnels
Bouton de minimisation et fermeture de la fenêtre Aucune erreur ne doit apparaitre, la fermeture ou la minimisation totale du programme doit se faire selon le bouton appuyé	Fonctionnel, ces boutons placés dans une barre customisée n'affichent aucune dis-fonctionnalité
Boutons Start Le bouton doit être fonctionnel. Aucune erreur ne doit apparaitre	Fonctionne, disparait après appuie.
Contrôles du joueur Il doit être contrôlable. Aucune erreur ne doit apparaitre. Les déplacements doivent être fluides. Une légère trainée doit apparaitre.	Le personnage est parfaitement contrôlable via les touches directionnelles du clavier. Pas de problème de fluidité. Trainée bien apparente.
Label Score et Timer Aucune erreur ne doit apparaitre. Le score et le timer doivent être fonctionnels. La fin du timer met fin à la partie.	Lorsque le timer arrive à 0, la partie se termine. Pas d'erreur
Comportement des personnages Les autres objets personnage doivent avoir un changement de localisation aléatoire une fois 'mangé' par le fantôme.	A chaque passage du premier personnage sur un objet personnage tiers, ces derniers changent de localisation. Pas d'erreur.

Conclusion

Malgré moins de temps que mes collègues de classe, je pense que mon travail est convenable, même si je me suis aperçue après coup ne pas avoir respecté toutes les consignes du sujet.

On pourrait ajouter à ce projet, la saisie du nom du personnage par l'utilisateur afin de sauvegarder le score de ce dernier.