

# Manuel Technique

## Travail pratique individuel

### Éditeur de Jeu 2D

Programmation orientée objet en C#

GOUVEIA DE OLIVEIRA André



# Table des matières

---

## Table des matières

### Introduction

#### Résumé du cahier des charges

1. But de l'application
2. Fonctionnalités à réaliser
3. Matériel et logiciels nécessaires
4. Livrable

### Méthodologie

#### Analyse fonctionnelle

Liste des fonctionnalités disponibles

Présentation de l'interface

L'icone

La fiche frmMain

La fiche frmAjoutSprite

La fiche frmPlateauJeu

La fiche frmCreationProjet

Les messages d'information

Enregistrement impossible sans Sprite

Mesures de sécurité mise en place

#### Analyse Organique

Classe Sprite

AjoutControlPanel - Sprite

SuprControlPanel - Sprite

AjoutControlPlateauJeu - Sprite

UpdateValue - Sprite

DemareAnimation - Sprite

CreationDossier - Sprite

Deplacement - Sprite

ChangePositionFinal - Sprite

ActiverDesactiverEvenement - Sprite

Rotation- Sprite

pbx\_MouseUp - Sprite

pbx\_MouseDown - Sprite

pbx\_MouseMove - Sprite

pbx\_Paint - Sprite

pbx\_Click - Sprite

tmp\_Tick - Sprite

## Classe Jeu

AddSprite - Jeu

RefreshControl - Jeu

ModifZOrder - Jeu

UpdateValueSpriteSelected - Jeu

GetValueSpriteSelected - Jeu

CheckNomExist - Jeu

CreationDossierProjet - Jeu

XMLSerialize - Jeu

CreateSpriteAfterDeserialize - Jeu

XMLDeserialize - Jeu

## Classe SpriteSerialisable

SetValue - SpriteSerialisable

## Forme frmMain

## Forme frmCreationProjet

## Forme frmAjoutSprite

frmPlateauJeu - Sprite

## Réalisation

Arborescence du projet

Diagramme de Classe

## Test

Plan de test

Rapport de test

## Comparaison planning prescrits et planning réel

Planning prescrit

Planning réel

## Conclusion

Bilan personnel

## Liens et références

Images

Autres

## Annexes

# Table des matières

---

## Table des matières

### Introduction

#### Résumé du cahier des charges

1. But de l'application
2. Fonctionnalités à réaliser
3. Matériel et logiciels nécessaires
4. Livrable

### Méthodologie

#### Analyse fonctionnelle

Liste des fonctionnalités disponibles

Présentation de l'interface

L'icone

La fiche frmMain

La fiche frmAjoutSprite

La fiche frmPlateauJeu

La fiche frmCreationProjet

Les messages d'information

Enregistrement impossible sans Sprite

Mesures de sécurité mise en place

#### Analyse Organique

Classe Sprite

AjoutControlPanel - Sprite

SuprControlPanel - Sprite

AjoutControlPlateauJeu - Sprite

UpdateValue - Sprite

DemareAnimation - Sprite

CreationDossier - Sprite

Deplacement - Sprite

ChangePositionFinal - Sprite

ActiverDesactiverEvenement - Sprite

Rotation- Sprite

pbx\_MouseUp - Sprite

pbx\_MouseDown - Sprite

pbx\_MouseMove - Sprite

pbx\_Paint - Sprite

pbx\_Click - Sprite

tmp\_Tick - Sprite

## Classe Jeu

AddSprite - Jeu

RefreshControl - Jeu

ModifZOrder - Jeu

UpdateValueSpriteSelected - Jeu

GetValueSpriteSelected - Jeu

CheckNomExist - Jeu

CreationDossierProjet - Jeu

XMLSerialize - Jeu

CreateSpriteAfterDeserialize - Jeu

XMLDeserialize - Jeu

## Classe SpriteSerialisable

SetValue - SpriteSerialisable

## Forme frmMain

## Forme frmCreationProjet

## Forme frmAjoutSprite

frmPlateauJeu - Sprite

## Réalisation

Arborescence du projet

Diagramme de Classe

## Test

Plan de test

Rapport de test

## Comparaison planning prescrits et planning réel

Planning prescrit

Planning réel

## Conclusion

Bilan personnel

## Liens et références

Images

Autres

## Annexes

# Introduction

---

Cette documentation est un rapport destiné au collègue d'expert en charge d'évaluer le projet. Il permet de présenter les différents aspects de la conception du projet FlappySharp. Le projet a été réalisé dans le cadre du Travail Pratique Individuel (TPI) pour valider mon CFC d'informaticien.

FlappySharp est un éditeur de jeux 2Ds qui permet aux utilisateurs de faire des jeux 2Ds qu'il pourront ensuite essayer. Pour ce faire, l'utilisateur peut créer des sprites à partir d'images qu'il peut placer où il veut.

## Résumé du cahier des charges

---

### 1. But de l'application

Offrir à l'utilisateur de faire un jeu 2D facilement sans avoir besoin de coder. Il permet de faire un jeu en 2D du type flappy bird en posant les Sprites où on veut.

### 2. Fonctionnalités à réaliser

L'application permet de faire des jeux 2Ds du type flappy bird. Dès le lancement, l'utilisateur peut créer un jeu 2D soit en utilisant les Sprites mis à disposition, soit en créant des Sprites avec des images importées. À la fin de la réalisation de son jeu 2D l'utilisateur pourra jouer au jeu qu'il vient de créer. Il pourra aussi le sauvegarder pour y rejouer ultérieurement ou y faire des modifications.

## 3. Matériel et logiciels nécessaires

- Ordinateur (Pc)
- Visual studio 2019
- Typora
- Suite Office

## 4. Livrable

- Planning
- Rapport de projet
- Journal de travail

# Méthodologie

---

Pour planifier mon projet, je me suis basé sur la méthode en 6 étapes. Cette méthodologie consiste à diviser le projet en plusieurs étapes afin de faciliter la planification et le développement.

1. S'informer
2. Planifier
3. Décider
4. Réaliser
5. Contrôler
6. Évaluer

# Analyse fonctionnelle

---

L'analyse fonctionnelle traite de la partie visible de l'application vue par l'utilisateur. Dans cette partie je parlerai en premier des fonctionnalités qui seront disponibles dans l'application. Par la suite, je parlerai des interfaces qui seront disponibles aux utilisateurs. Enfin, des mesures de

## Liste des fonctionnalités disponibles

Voici la liste des actions possibles pour l'utilisateur du FlappySharp :

- Ajout de Sprite

L'utilisateur a deux possibilités pour ajouter un Sprite dans la scène :

- utiliser un des Sprites déjà existant : l'application propose déjà des Sprites qui sont présents par défaut. L'utilisateur peut donc les ajouter et les modifier.
- créer un Sprite : l'utilisateur peut aussi créer son propre Sprite. Lors de l'ajout une fenêtre s'ouvre et l'utilisateur doit indiquer les valeurs pour le Sprite tel que la position, la taille, les images et le calque dans lequel il sera.

- Modification des paramètres des Sprites

- modifier le nom du Sprite
- modifier la taille du Sprite
- modifier la liste d'images du Sprite
- modifier l'intervalle de temps pour l'animation du Sprite
- modifier le calque et le ZOrder du Sprite
- modifier la position du Sprite
- modifier le tag du Sprite
- modifier la rotation du Sprite

- Lancement du jeu :

Une fenêtre s'ouvre et le jeu se lance donnant ainsi un aperçu du projet réalisé.

- Sauvegarder/Ouvrir le projet :

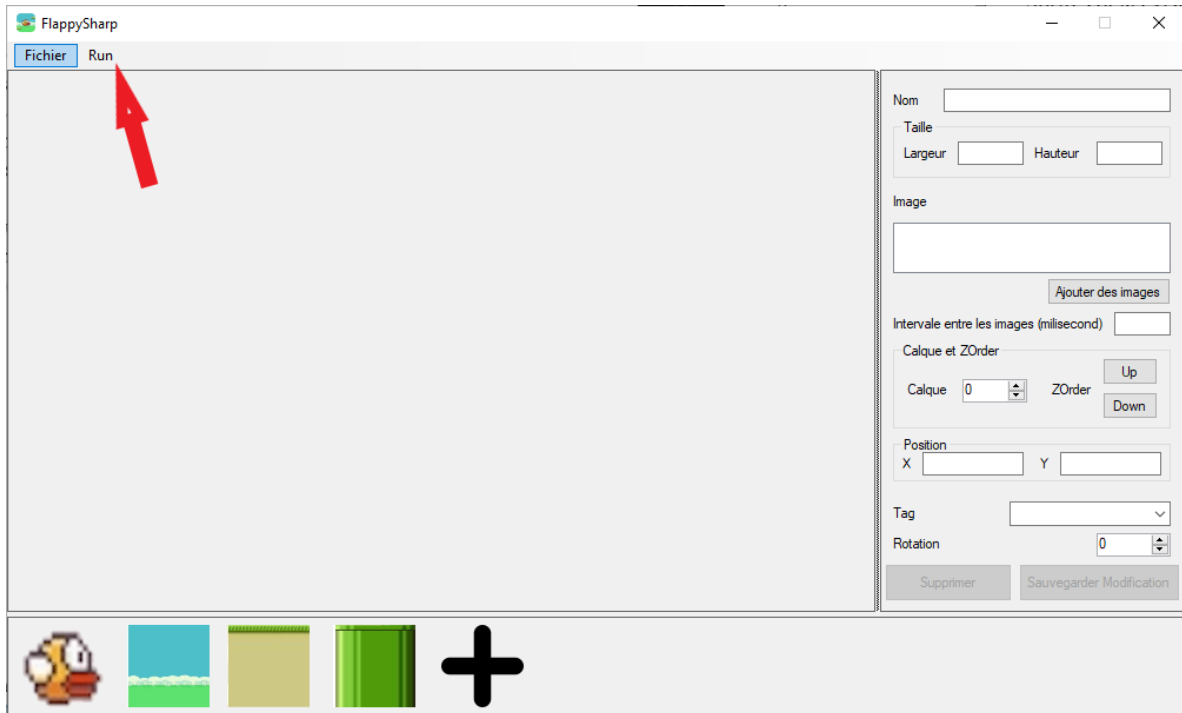
L'utilisateur peut sauvegarder son projet au format XML qu'il pourra par la suite ouvrir avec l'application.

## Présentation de l'interface



## L'interface utilisateur comprend un total de 4 formulaires

WindowsForms. La première interface est la forme principale, celle où l'utilisateur fait son jeu 2D avec tous les paramètres des Sprites et il peut créer des Sprites, soit en utilisant des Sprites déjà existant, soit en en créant de nouveau. A la suite de ce choix, une fenêtre s'ouvre et l'utilisateur indique dans les champs les valeurs pour le Sprite. Une fois le jeu terminé par l'utilisateur, il peut y jouer et il lui suffit donc d'appuyer sur le Button "Run" pour ouvrir une fenêtre où il pourra jouer à son jeu.



Toutes les images utilisées au sein de l'application sont des images totalement libres de droits et utilisables dans le cadre du TPI.

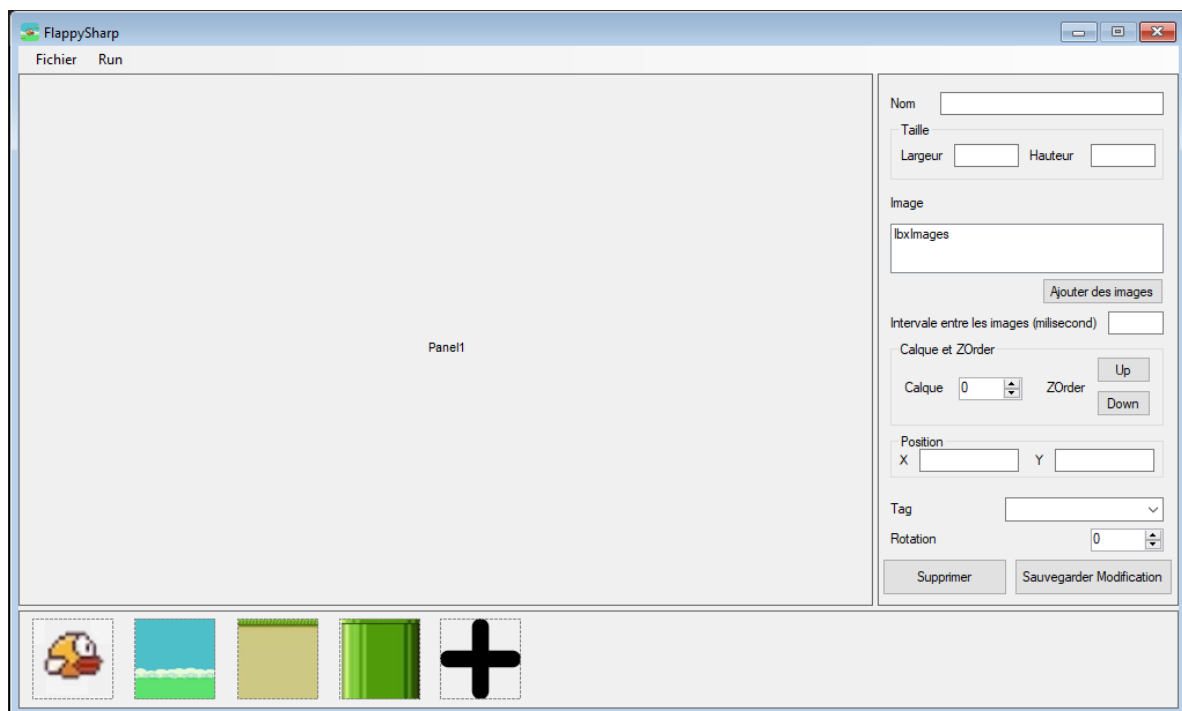
## L'icone



L'icone de l'application a été récupéré sur Google Image à l'adresse : <http://dlpng.com/png/6781141>

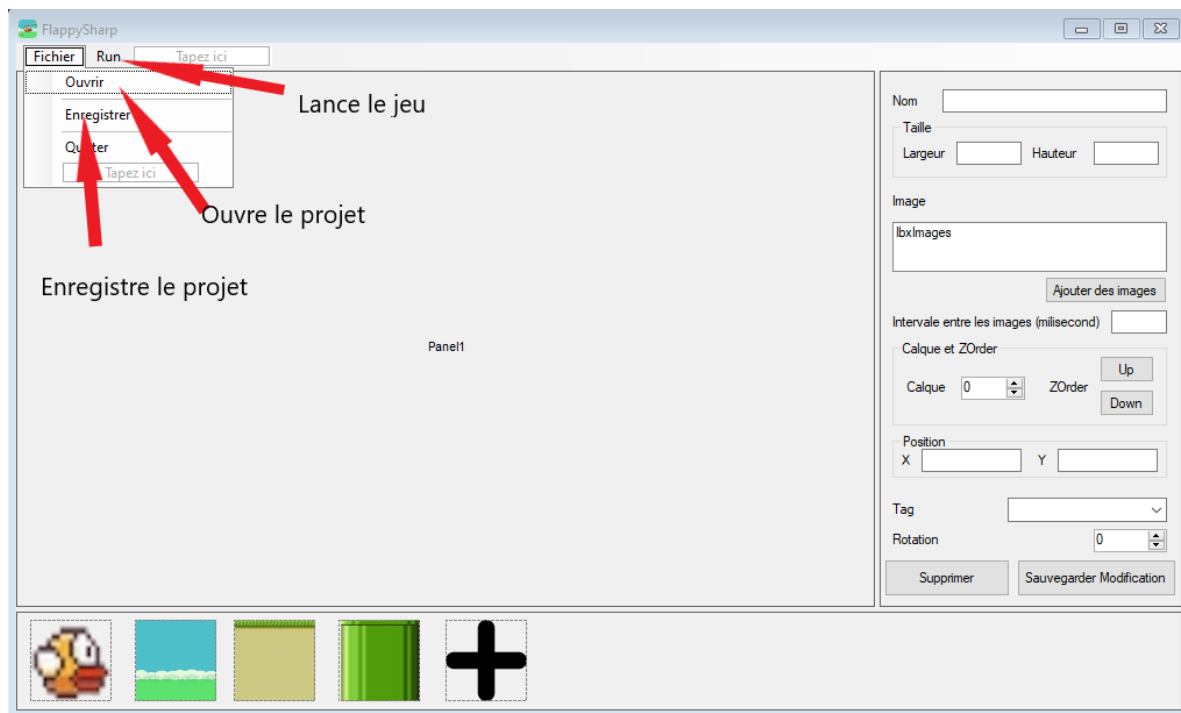
Je l'ai par la suite ajouter à l'application par le biais e propriétés du projet.  
Licone est un flappy bird qui passe entre deux tuyau.

## La fiche frmMain



Le formulaire principale est la forme de création de jeu. Le formulaire contient plusieurs parties :

- La première partie qui est la scène de jeu où l'on place les Sprites du jeu que l'on peut déplacer.
- Après, il y a le menuStrip qui permet de lancer le jeu en appuyant sur "Run" ce qui ouvrira une nouvelle fenêtre avec le jeu que l'utilisateur à créer. Il permet aussi de sauvegarder son jeu au format XML en utilisant la sérialisation (le format XML a été choisie ici pour que l'utilisateur puisse directement changer les valeurs de ces Sprites sans avoir besoin de lancer l'application) et d'ouvrir un projet au format XML en utilisant la désérialisation.



- Ensuite il y a la partie de paramétrage des Sprites qui se met à jour quand on clique sur un Sprite. Cette partie contient six TextBox qui permettent de modifier des paramètres tel que le nom du Sprite, la taille du Sprite (que ce soit en largeur ou en hauteur), le délai entre les images que l'utilisateur a choisi pour le Sprite et enfin la position du Sprite. Cette partie contient aussi des numeriqueUpDown qui sont utilisés pour la rotation du Sprite ainsi que pour définir le calque du Sprite ainsi que d'une listBox pour afficher les images que contiennent les Sprites. Il y a aussi un Button qui permet d'ajouter des images aux Sprites, deux Button pour changer le ZOrder de chaque Sprite, ce qui permet de changer dans un même calque la profondeur des

Sprites, et un Button qui permettent de sauvegarder les modifications faite sur le Sprite et un Button de suppression du Sprite qui supprime totalement le Sprite.

Nom

Taille  
Largeur  Hauteur

Image

Intervale entre les images (milisecond)

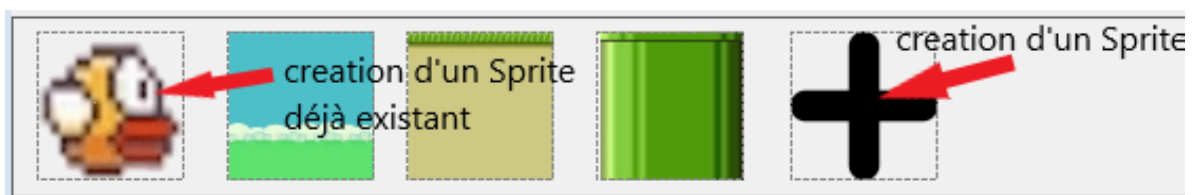
Calque et ZOrder  
Calque  ZOrder

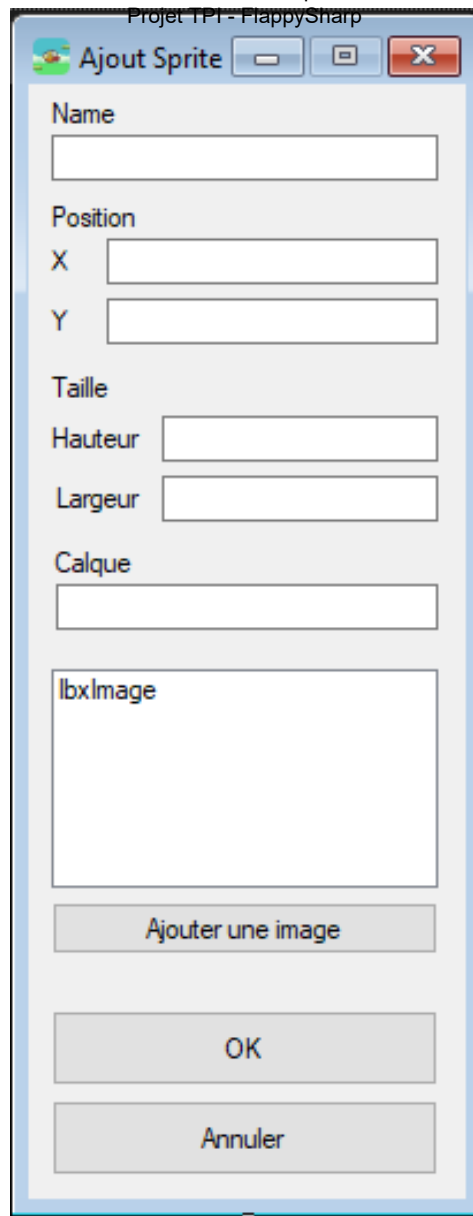
Position  
X  Y

Tag

Rotation

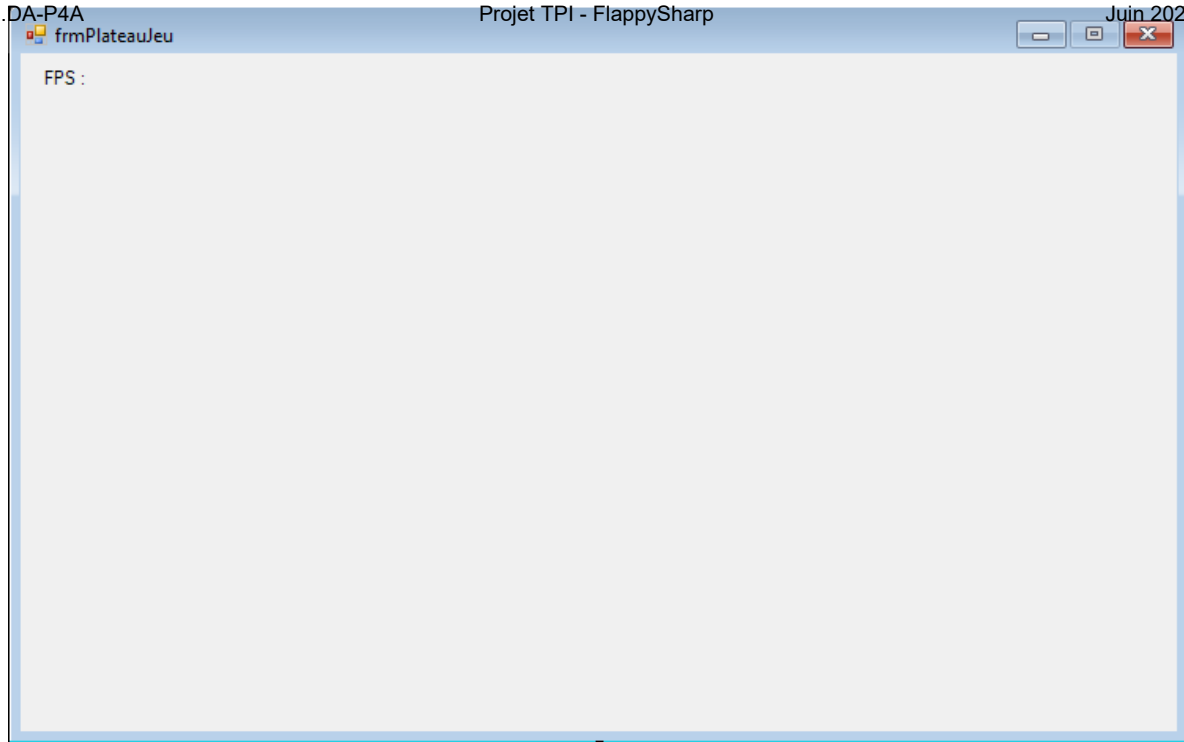
- Pour finir, il y a la partie "ajout de Sprite" qui contient des Sprites déjà prédéfinies qui se mettent automatiquement sur la scène de jeu avec des paramètres par défaut que l'utilisateur peut ensuite modifier. Il y a aussi le PictureBox qui contient un "plus" qui ouvre une nouvelle fenêtre qui permet de créer un Sprite selon les volontés de l'utilisateur en spécifiant des paramètres.





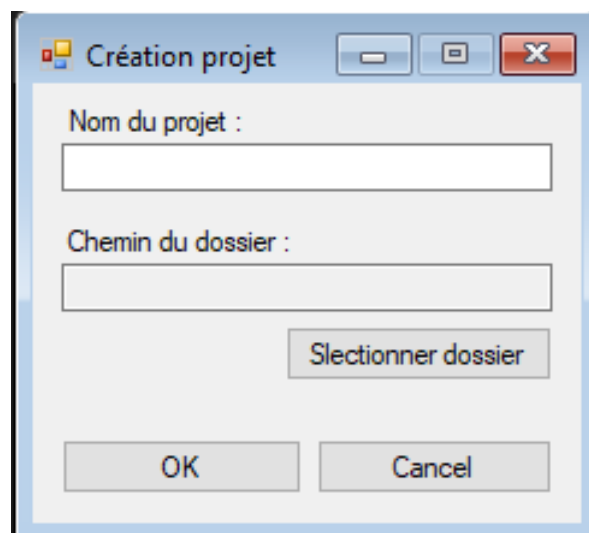
La forme frmAjoutSprite s'ouvre après avoir cliquer sur le pictureBox avec une image de "plus" sur la forme frmMain. La forme permet de créer un Sprite en indiquant dans les champs les valeurs qu'aura le Sprite. Les valeurs que l'utilisateur peut définir pour le Sprite sont : le nom, qui ne peut pas être le même si le nom choisi est le même qu'un déjà existant (un chiffre s'ajoute à la fin du nom le rendre unique), la position (X et Y), la taille (Hauteur et Largeur), le calque où il sera et le ZOrder qui se place automatiquement afin qu'il n'y ait pas deux Sprite sur le même ZOrder. Et enfin il y a les images qui permettent de faire une animation du Sprite. Il y a ensuite deux Button qui permettent à l'utilisateur d'ajouter le Sprite ou de stopper l'ajout du Sprite.

## La fiche frmPlateauJeu



La forme PlateauJeu s'ouvre après avoir appuyé sur le bouton "Run" du menuStrip de la forme Main. La forme permet à l'utilisateur de jouer au jeu qu'il vient de faire. La forme affiche le nombre de fps qu'il y a sur la forme.

## La fiche frmCreationProjet

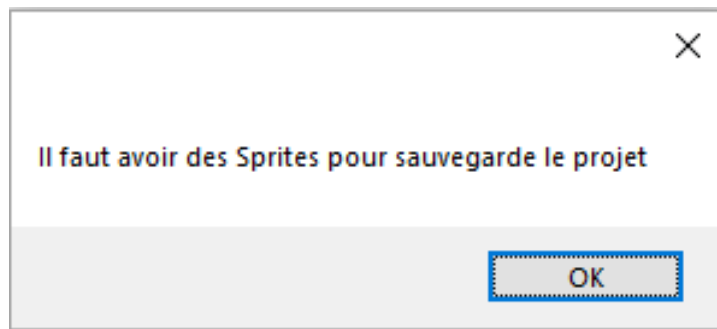


La forme CreationProjet s'ouvre après avoir appuyé sur le bouton "Enregistrer" du menuStrip de la forme Main. La forme crée un dossier ou elle enregistre un fichier xml contenant toutes les données des Sprites dans la forme et elle crée un dossier pour chaque Sprite contenant les images du Sprite.

# Les messages d'information

Des messages d'informations à destination de l'utilisateur sont prévus pour informer les utilisateurs.

## Enregistrement impossible sans Sprite



Ce message indique à l'utilisateur qu'il lui est impossible d'enregistrer son projet sans avoir de Sprite dans la scène.

## Mesures de sécurité mise en place

Voici une liste des contrôles mis en place pour éviter tout éventuelles mauvaises manipulations des utilisateurs et d'empêcher les erreurs dans le code.

- La taille des fenêtres est fixe pour ne pas avoir de problème d'affichage
- Un traitement des inputs a été fait sur les TextBox qui ne doivent pas contenir de texte
- Les button "Supprimer" et "Sauvegarder les modifications" sont inaccessibles si aucun Sprite n'est sélectionné
- On ne peut pas ajouter de Sprite tant que tous les champs ne sont pas remplis
- On ne peut pas enregistrer un projet sans Sprite

# Analyse Organique

---

## Classe Sprite

Cette classe qui est héritée de PictureBox est utilisée pour créer les Sprites. Pour la création d'un Sprite le constructeur reçoit des paramètres.

- string Nom
- Size Taille
- Dictionary<string,Bitmap> Images
- int Calque
- int ZOrder
- Point Position
- Panel ZoneScene (ou les sprites sont ajouter)

Cette classe est utilisée dans la classe Jeu.

## AjoutControlPanel - Sprite

Cette méthode a pour but de mettre le Sprite dans les controls du panel. Elle est appelée dans le constructeur du Sprite et dans la méthode RefreshControl() de la classe Jeu.

## SuprControlPanel - Sprite

Cette méthode a pour but de supprimer le Sprite. Elle est appelée par la méthode RefreshControl() et par btnSupr\_Click(). Elle reçoit un bool en paramètre. Si le bool est false la méthode supprime le Sprite seulement dans les controls du panel, mais si le bool est true la méthode supprime totalement le Sprite.

## AjoutControlPlateauJeu - Sprite



Cette méthode a pour but d'ajouter le Sprite dans les controls de la forme PlateauJeu. Elle est appelé par l'événement frmPlateauJeu\_Load().

## UpdateValue - Sprite

Cette méthode a pour but de mettre à jour les valeurs du Sprite. Elle est appelée par la méthode UpdateValueSpriteSelected() dans la classe Jeu. Elle reçoit en paramètre un nom, une taille, des images, un interval entre les images, un calque, un ZOrder, un tag, une rotation que les paramètres vont ensuite mettre dans les bonnes variables.

## DemareAnimation - Sprite

Cette méthode a pour but de démarrer le timer du Sprite pour que l'animation puisse se lancer. Elle est appelé par deux événements frmPlateauJeu\_Load() et frmPlateauJeu\_FormClosing(). Cette méthode inverse le Enable du timer.

## CreationDossier - Sprite

Cette méthode a pour but de créer le dossier et sauvegarder les images dans le dossier. Elle est appelée par la méthode CreationDossierProjet(). Cette méthode reçoit en paramètre un cheminDossier qui permet de mettre à jour le paramètre du chemin du dossier où l'on va enregistrer les images. On crée le dossier où l'on va stocker les images en indiquant le chemin où il doit être créé et en indiquant le nom qu'il aura. Ensuite on parcourt le Dictionary des images pour sauvegarder toutes les images en indiquant le chemin du dossier.

## Deplacement - Sprite

Cette méthode a pour but de gérer les déplacements du Sprite. Elle est appelée par l'événement tmp\_Tick(). Cette méthode gère les déplacements du Sprite en fonction de son tag :

- Si le tag vaut "Player" alors on vérifie si la variable positionFinal est plus petite que zéro, si c'est le cas on la met à zéro sinon on ne fait rien. Ensuite on vérifie si la position en Y du Sprite est plus grande que la variable positionFinal, si oui on met spriteVYDescend à 1, on met à jour la position du Sprite en diminuant la position en Y du Sprite de spriteVYMonte et on augmente de 0,08 spriteVYMonte. Sinon on met spriteVYMonte égal à 1, on modifie la variable positionFinal qui prend comme valeur la Position en Y du Sprite plus 200, on met à jour la position du Sprite en augmentant la Position en Y de spriteVYDescend et on augmente spriteVYDescend de 0.08. Enfin on met à jour la variable collision.
- Si le tag vaut "Ennemie" alors on vérifie si la position du Sprite est égale à la taille du Sprite en négatif, si oui on met à jour la position du Sprite en la mettant l'image en bordure à droite de la forme. Sinon on met à jour la position du Sprite en diminuant la position Y du Sprite de vitesseLateral.

## ChangePositionFinal - Sprite

Cette méthode a pour but de mettre à jour la variable positionFinal. Elle est appelé par l'évènement frmPlateauJe\_KeyDown(). Cette méthode modifie la variable positionFinal qui prend comme valeur la position en Y du Sprite moins la valeur en paramètre.

## ActiverDesactiverEvenement - Sprite

Cette méthode a pour but de désactiver ou activer les événements du Sprite. Elle est appelé deux fois par la méthode runToolStripMenuItem\_Click(). Cette méthode active ou désactive les événements de Sprite en fonction du paramètre.

## Rotation- Sprite

Cette méthode a pour but de tourner l'image en fonction d'un angle. Elle est appelée par la méthode RotateImage(). Cette méthode crée un bitmap qui contient l'image mise en paramètre et crée un Graphics du bitmap. Elle fait ensuite un RotateTronform() de l'angle mis en paramètre et enfin elle dessine le bitmap dans l'image mise en paramètre et la return.

## pbx\_MouseUp - Sprite

Cet événement se produit sur le relâchement du click de la souris sur Sprite. Le paramètre sender permet de savoir sur quel Sprite le relâchement du click a été fait. Par la suite on vérifie si le Sprite existe, sinon on arrête la méthode ou alors on change la variable dragging a false et on crée un nouveau rectangle qu'on implémente à la variable collision.

## pbx\_MouseDown - Sprite

Cet événement se produit sur l'appui constant de la souris sur le Sprite. Le paramètre sender permet de savoir sur quel Sprite le click est appuyé. Ensuite on vérifie si le Sprite existe, sinon on arrête la méthode, après on met la variable dragging à true, on met à jour les variables xPos et yPos avec la position X et Y du Sprite et on crée un nouveau rectangle pour la collision.

## pbx\_MouseMove - Sprite

Cet événement se produit au moment du déplacement de la souris. La méthode permet de vérifier si le sender n'est pas nul. S'il ne l'est pas il fait bouger le Sprite en fonction de la position de la souris.

## pbx\_Paint - Sprite

Cet événement se produit au moment où le rafraîchissement est demandé. Le PaintEventArgs dessine un rectangle autour du Sprite ce qui permettra par la suite de faire la collision entre deux Sprites.

## pbx\_Click - Sprite

Cet événement se produit au moment où l'on click sur le Sprite. On met la variable selcted à true, qui est utilisé pour savoir quelle Sprite est sélectionnée.

## tmp\_Tick - Sprite

Cet événement a pour but de faire l'animation du Sprite. Cet événement se produit après la valeur contenue dans la variable intervalEntreImage. L'événement met l'image du dictionary indiqué avec la variable imageAnime, et si l'identifiant est égal au nombre d'images dans le dictionary alors on met la variable imageAnime à zéro. Sinon on l'augmente de 1.

## Classe Jeu

Cette classe est utilisée pour gérer les Sprites et les valeurs qui passent par les vues.

## AddSprite - Jeu

Cette méthode a pour but de créer un Sprite à partir des valeurs reçue en paramètre. La classe reçoit les paramètres, elle calcule automatiquement le zorder du Sprite pour que ce soit le dernier Sprite du Calque afin qu'il n'y ait pas deux Sprite sur le même. Au moment de l'ajout du Sprite, en appelant le constructeur de la classe Sprite, on appelle la méthode CheckNomExist() pour vérifier si le nom existe déjà. Si c'est le cas la méthode ajoute un nombre à la fin du nom pour le rendre unique. Pour finir on appelle la méthode RefreshControl() pour mettre à jour tous les controls.

## RefreshControl - Jeu

Cette méthode a pour but de mettre à jour les controls. Elle est appelée cinq fois par `runToolStripMenuItem_Click()`, `AddSprite()`, `ModifZOrder()`, `UpdateValueSpriteSelected()` et `CreateSpriteAfterDeserialize()`. Cette méthode parcourt toute la liste de Sprite et les supprime en appelant la méthode `SuprControlPanel()` pour chaque Sprite, puis elle tri la liste de Sprite et elle finit par ajouter toute la liste de Sprite en appelant la méthode `AjoutControlPanel()` pour chaque Sprite.

## ModifZOrder - Jeu

Cette méthode a pour but de modifier le zorder. Elle est appelée deux fois par la méthode `UpdateValueSpriteSelected()`. Cette méthode vérifie que la liste de Sprite n'est pas vide. Si elle ne l'est pas, elle la parcourt et pour chaque Sprite elle vérifie si le nom n'est pas le même que le Sprite passer en paramètre. Le zorder et le calque sont les mêmes que le Sprite passés en paramètre. Ensuite on vérifie

- si la variable du `ModificationZOrder` du Sprite passé en paramètre est égale à "+". On modifie le zorder du Sprite sélectionné dans la liste et on lui enlève un et enfin on met à null la variable `ModificationZOrder` du Sprite passé en paramètre.
- Si la variable du `ModificationZOrder` du Sprite passé en paramètre est égale à "-", on modifie le zorder du Sprite sélectionné dans la liste et on lui ajoute un et enfin on met à null la variable `ModificationZOrder` du Sprite passé en paramètre.

Pour finir on appelle la méthode `RefreshControl()`.

## UpdateValueSpriteSelected - Jeu

Cette méthode a pour but de mettre à jour les données du Sprite sélectionné.

Elle est appelée par l'événement `btnSauveModif_Click()`. Cette méthode vérifie

- si la variable, calque du Sprite sélectionné, est la même que la variable calque passée en paramètre. Alors on vérifie

- si le zorder du Sprite sélectionné est plus petit que le zorder du Sprite sélectionné, plus le zorder passé en paramètre. On appelle la méthode UpdateValue(), on met les variable en paramètre dans la méthode, on modifie la variable ModificationZOrder à "+" et enfin on appelle la méthode ModifZOrder en mettant en paramètre le Sprite sélectionné.
- Si le zorder du Sprite sélectionné est plus grand que le zorder du Sprite sélectionné, plus le zorder passer en paramètre, alors on appelle la méthode UpdateValue(). On met les variable en paramètre dans la méthode, on modifie la variable ModificationZOrder à "-" et enfin on appelle la méthode ModifZOrder en mettant en paramètre le Sprite sélectionné.
- Si le zorder du Sprite sélectionné est égale au zorder du Sprite sélectionné plus le zorder passé en paramètre, alors on appelle la méthode UpdateValue(). On met les variables en paramètre dans la méthode et on appelle la méthode RefreshContol().
- Sinon on met la variable zorder à zéro, ensuite on vérifie si la liste de Sprite n'est pas vide et qu'il n'y a pas de Sprite sur le même calque
  - La variable zorder prend le zorder du dernier Sprite dans le même calque et ensuite on ajoute un à la variable zorder.
- On appelle la méthode UpdateValue(), on met les variable en paramètre dans la méthode et pour finir on appelle la méthode RefreshContol().

## GetValueSpriteSelected - Jeu

Cette méthode a pour but de return un Sprite. Elle est appelée deux fois par l'événement tmp\_Tick(). Cette méthode parcourt toute la liste de Sprite et vérifie si la variable Selected du Sprite sélectionné dans la liste est true et que le sprite est différent de la variable spriteSelected. On met

le Sprite sélectionné dans la liste de la variable `spriteSelected` et ensuite on met la variable `Selected` de `spriteSelected` à `false`.

## CheckNomExist - Jeu

Cette méthode a pour but de retourner un nom unique. Elle est appelée par la méthode `AddSprite()`. Cette méthode vérifie si la liste de `Sprite` n'est pas vide, ensuite tant que la variable `nomModifier` est égale à un nom d'un des `Sprite` dans la liste de `Sprite`, alors la variable `nomModifier` prend la variable en paramètre `checkNomSprite` plus la variable `compteur`.

## CreationDossierProjet - Jeu

Cette méthode a pour but de créer l'arborescence du projet. Elle est appelée par l'événement `enregistrerToolStripMenuItem_Click()`. Cette méthode change la valeur de la variable `NomProjet` par la variable en paramètre `nomProjet` et change la valeur de la variable `CheminDossierProjet` par la variable en paramètre `cheminDossierProjet`. Il crée ensuite le dossier du projet avec, en paramètre, le chemin où l'on veut mettre le dossier et le nom du dossier. Il parcourt ensuite toute la liste de `Sprite` et pour chaque `Sprite` il appelle la méthode `CratationProjet()` où il faut mettre en paramètre le chemin dossier créé juste avant.

## XMLSerialize - Jeu

Cette méthode a pour but de créer un fichier xml qui contient toutes les données du projet. Elle est appelée par l'évènement `enregistrerToolStripMenuItem_Click()`. Cette méthode parcourt la liste de `Sprite` et crée un `spriteSerialisable` pour chaque `Sprite` de la liste. Ensuite elle enregistre dans une liste de string le nom de toutes les images du `Dictionary` et elle appelle la méthode `SetValue()` qui met à jour les valeurs du `spriteSerialisable` pour l'ajouté à la liste de `spriteSerialisable`. Ensuite on serialise la liste de `spriteSerialisable` en indiquant l'emplacement où on veut mettre le fichier xml.

## CreateSpriteAfterDeserialize - Jeu

Cette méthode a pour but de créer d'ajouter des Sprites à la liste de Sprite en fonction de la liste de spriteSerialisable. Elle est appelée par l'évènement ouvrirToolStripMenuItem\_Click(). Cette méthode supprime tous les Sprites du panel et après elle modifie la valeur de la liste de spriteSerialisable en appelant la méthode XMLDeserialize(). Elle parcourt ensuite la liste de spriteSerialisable et elle met à jour la variable cheminDossierProjet et parcourt la liste de nom de chaque spriteSerialisable pour les ajouter dans le Dictionary des images. Elle crée ensuite un Sprite en utilisant les données du spriteSerialisable sélectionné quel met en paramètre. Enfin elle appelle la méthode RefreshControl().

## XMLDeserialize - Jeu

Cette méthode a pour but de retourner une liste de spriteSerialisable. Elle est appelée par la méthode CreateSpriteAfterDeserialize(). Elle ouvre une connexion sur le dossier contenu dans la variable passer en paramètre. Ensuite elle deserialise le fichier sélectionné quel met dans la variable obj. Pour finir elle return la variable obj.

## Classe SpriteSerialisable

### SetValue - SpriteSerialisable

Cette méthode a pour but de mettre à jour les valeurs du SpriteSerialisable. Elle est appelée par la méthode XMLSerialize(). Elle met les valeurs passer en paramètres dans les bonnes variables.

## Forme frmMain

Cette forme est la forme principale. Elle permet de créer, modifier et Supprimer les Sprites mais aussi de sauvegarde et d'ouvrir des projets. Enfin elle permet de lancer le jeu pour y jouer.



# Forme frmCreationProjet

Cette forme est la forme pour la création du projet. Elle permet de récupérer l'emplacement du projet ainsi que son nom grâce à l'appel des méthodes qui les return.

## Forme frmAjoutSprite

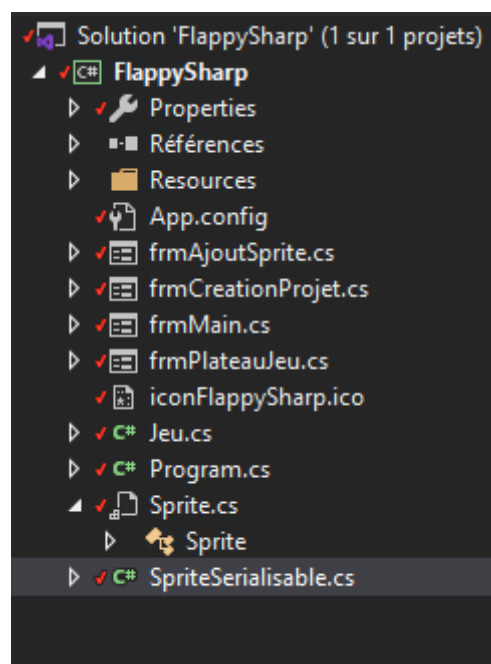
Cette forme est la forme pour la création d'un Sprite. Elle permet de récupérer le nom, la position, la taille, le calque et les images du Sprite grâce à l'appel des méthodes qui les return.

## frmPlateauJeu - Sprite

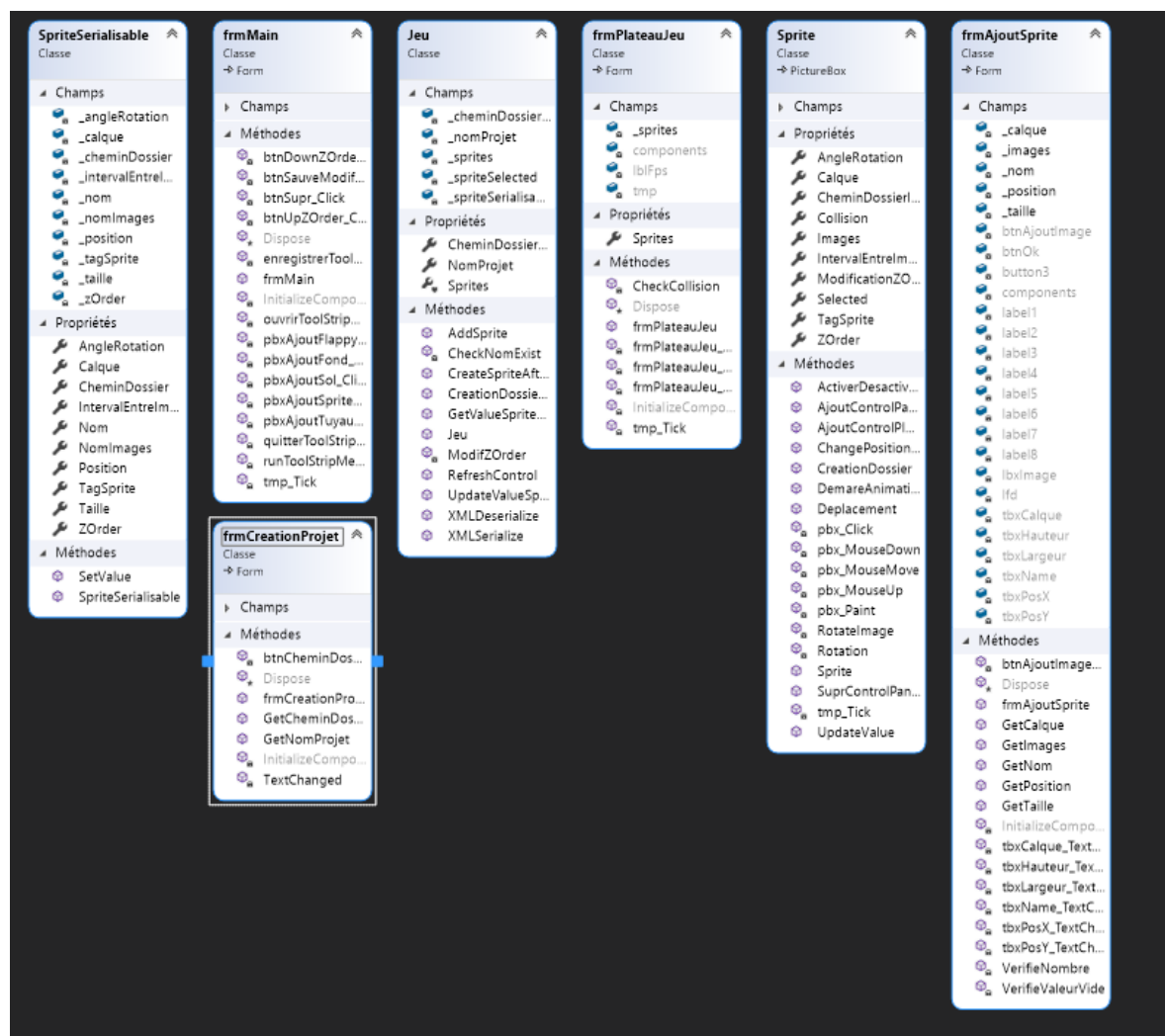
Cette forme est la forme où l'on joue au jeu. Elle permet de faire l'animation et déplacement des Sprites grâce à un timer. Elle permet aussi de gérer les collisions.

## Réalisation

### Arborescence du projet



# Diagramme de Classe



## Test

### Plan de test

Un plan de test des principales fonctionnalités de l'application a été prévu. Il prévoit de contrôler les fonctionnalité du programme.

N°	DESCRIPTION DU TEST	RÉSULTAT ATTENDU
1	Lancement de l'application	Impossible de supprimer et de sauvegarder

N°	DESCRIPTION DU TEST	RÉSULTAT ATTENDU
2	Création de Sprite avec image existante	Chaque PictureBox avec une image peut crée un Sprite
3	Création de Sprite en choisissant les images	On peut crée un Sprite en choisissant l'image
4	Suppression d'un Sprite	Le Sprite se supprime correctement
5	Modification d'un Sprite	Les modifications s'appliquent au Sprite
6	Enregistrement du projet	L'arborescence se crée correctement ainsi que la création du fichier xml contenant les valeurs des Sprites
7	Ouvrir le projet	La création des Sprites à partir du fichier xml se fait correctement
8	Lancement du jeu	Les animation se font correctement et les déplacement fonctionnent

## Rapport de test

N°	04.06.2020	05.06.2020	08.06.2020
1	OK	OK	OK
2	OK	OK	OK
3	OK	OK	OK
4	OK	OK	OK
5	OK	OK	OK
6	NOK	OK	OK
7	OK	OK	OK
8	NOK	OK	OK

# Comparaison planning prescrit et planning réel

Pour ce travail j'ai eu 11 jours de 8 de travail par jour. J'ai donc dû planifier mon travail sur ce temps sans savoir s'il était juste. J'ai donc sur la base de mon planning prescrit fait un planning réel qui démontre le temps que j'ai pris pour faire les tâches prévues. Je vais donc maintenant faire la comparaison de ces deux tableaux pour voir si mon planning était optimal pour ce projet.

On peut donc voir que dans la globalité le planning prescrit était juste. Il y a juste eu le point "conceptions des plans, Sprites, run" qui a duré plus longtemps que prévu ainsi que le test et la résolution des problèmes trouver durant les tests mais mis à part ça les plannings sont assez semblables.

## Planning prescrit

Jour	1	2	3	4	5	6	7	8	9	10	11
Demi-Journée	1	2	3	4	5	6	7	8	9	10	11
Etude du sujet. Planification											
Installation, mise en place											
Conception interface											
Classes de base, OA											
Conception des plans, Sprites, run											
Gestion fichiers											
Finalisation / Corrections											
Tests											
Documentation											
Résumé											
Finalisation / Impressions											
Journal											

## Planning réel

Jour	1	2	3	4	5	6	7	8	9	10	11
Demi-Journée	1	2	3	4	5	6	7	8	9	10	11
Etude du sujet. Planification											
Installation, mise en place											
Conception interface											
Classes de base, OA											
Conception des plans, Sprites, run											
Gestion fichiers											
Finalisation / Corrections											
Tests											
Documentation											
Résumé											
Finalisation / Impressions											
Journal											

# Conclusion

---

Ce projet a été réalisable grâce aux connaissances acquises durant toute ma formation d'informaticien au CFPT-I.

## Bilan personnel

Le TPI m'a donné l'occasion d'avoir fait un projet avec des contraintes imposées tel que dans le monde du travail avec un temps imparti. Il m'a permis de savoir gérer mon temps pour avancer efficacement dans mon projet tout en avançant dans la documentation. Même avec ce délai restreint j'ai réussi à finaliser mon projet même s'il n'est pas aussi bien que j'aurais voulu qu'il soit.

## Liens et références

---

### Images

- Images utilisées dans frmMain dans les pictureBox : [https://www.pngfind.com/mpng/iRmmbbJ\\_flappy-bird-atlas-png-atlas-png-flappy-bird/](https://www.pngfind.com/mpng/iRmmbbJ_flappy-bird-atlas-png-atlas-png-flappy-bird/)
- Icon de l'appli : <https://dlnpng.com/png/6781141>

### Autres

- Permet de créer des dossiers : <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/file-system/how-to-create-a-file-or-folder>
- Déplacement de Sprite : [https://www.raspberrypi.org/magpi-issues/Essentials\\_Games\\_v1.pdf](https://www.raspberrypi.org/magpi-issues/Essentials_Games_v1.pdf)
- Permet de récupérer le nom du fichier et le chemin du fichier : <https://www.aspsnippets.com/Articles/Windows-Forms-WinForms>

- OpenFileDialog-Box-Tutorial-with-example-in-C-and-VBNet.aspx#:~:text=In%20order%20to%20select%20a%20Multiple%20Files%2C%20the%20Multiselect%20property,in%20the%20Windows%20Forms%20MessageBox.

## Annexes

---

Vous trouverez le code source dans le fichier Code\_Source\_Projet\_FlappySharp.pdf il a été généré automatiquement depuis overleaf.

Vous trouverez aussi en fichier zip des pages html qui ont été généré par doxygen pour voir les classe avec leurs méthodes sur index.html.