



SAE 1.01, 1.02



# STACK'N'CRÊPE

Groupe 6

 IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

## Sommaire

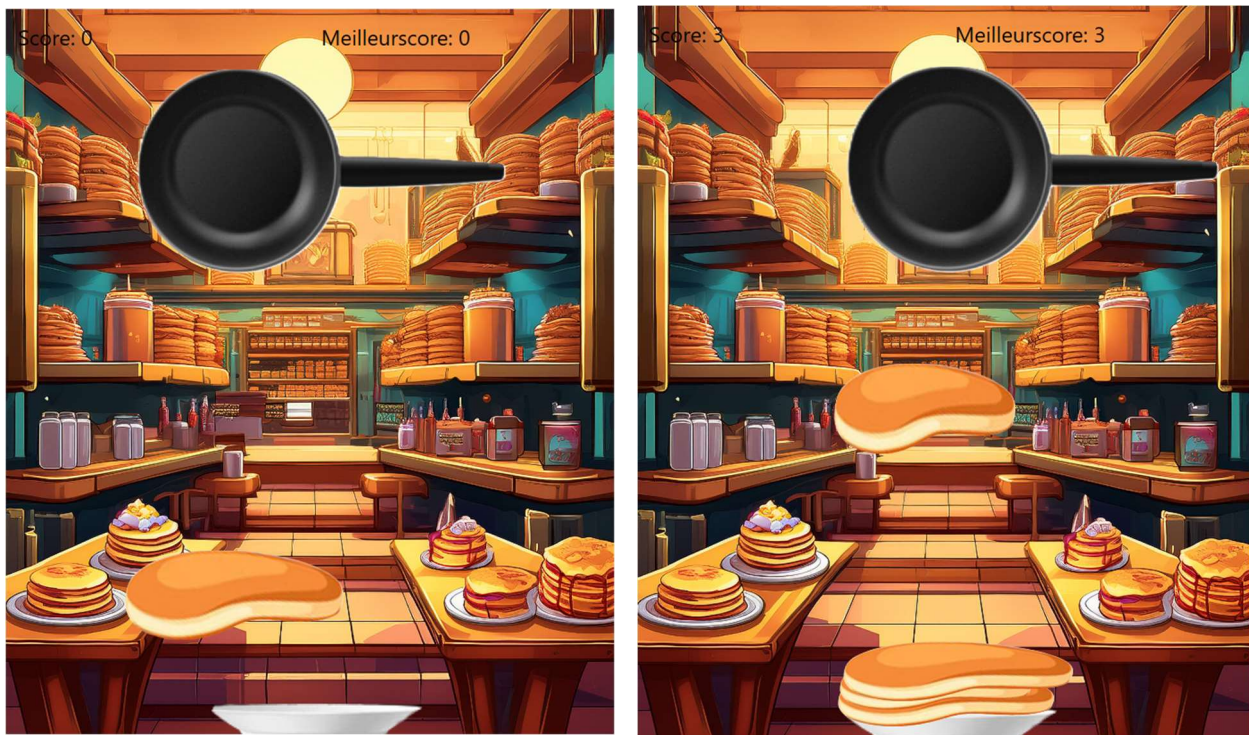
<b>1. PRESENTATION</b>	<b>3</b>
1.1. DESCRIPTION GENERALE	3
1.2. REGLES DU JEU	3
1.3. CINEMATIQUE DES ECRANS	4
<b>2. CONCEPTION – DIAGRAMME DE CLASSE</b>	<b>5</b>
2.1. PRESENTATION GENERALE	5
2.2. PRESENTATION DETAILLEE MAINWINDOW	5
2.3. PRESENTATION DETAILLEE AUTRE CLASSE	7
<b>3. PARTIE ALGORITHMIE</b>	<b>8</b>
3.1. DETECTION DES COLLISIONS TIRS JOUEUR SUR ENNEMI	ERREUR ! SIGNET NON DEFINI.
<b>4. CONCEPTION GRAPHIQUE</b>	<b>11</b>
<b>5. CAHIER DE RECETTES</b>	<b>12</b>
5.1 TESTS DE VALIDATION	12
5.2 TESTS DE PERFORMANCE	12

	<p>Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor</p>	<p>SAÉ S1_01_02 Développement en C# Jeu WPF</p>	
<p>NOM DU JEU</p>			

# 1.Présentation

## 1.1. Description générale

Stack'N'Crêpe est un jeu de précision sans prise de tête dont le but est d'empiler des crêpes et ainsi d'obtenir le meilleur score possible.



2 touches s'offrent au joueur, le largage de la crêpe, que vous pouvez voir ci-dessus, ainsi que la pause dont il peut changer les touches parmi trois mappages prédéfinis.



## 1.2. Règles du jeu

Bienvenue dans le jeu Stack'N'Crêpe, si votre but est de vous amuser alors vous êtes au bon endroit. Le but est simple : empiler le plus de crêpes possibles les unes sur les autres.

Mais attention la tâche n'est pas si simple car il existe plusieurs façons de perdre :

- Faire tomber une crêpe à côté du haut de la pile
- Atteindre une pile de 42
- Avoir un côté trop déséquilibré qui ferait tomber l'assiette

Heureusement, si vous avez la chance d'attraper la pelote, cela attirera le chat qui fera alors malencontreusement tomber votre assiette tout en vous permettant de continuer le jeu.



	Groupe 2 Bouvier-Sobrin Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

Cependant ne vous reposez pas sur vos acquis car un courant d'air pourrait vous surprendre.

Vous avez trois difficultés : facile, normal et difficile. Si vous êtes débutant nous vous conseillons évidemment de prendre le niveau facile, mais si le challenge vous manque n'hésitez à tester le dernier mode.

### 1.3. Cinématique des écrans

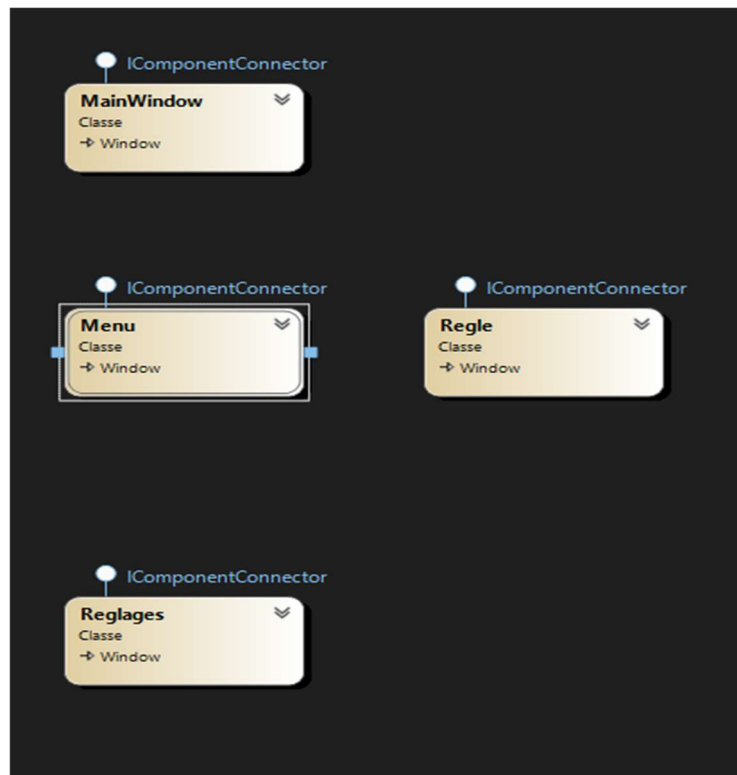


	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

## 2. Conception – Diagramme de classe



Attention : cette partie doit justifier et expliquer la note de l'autoévaluation

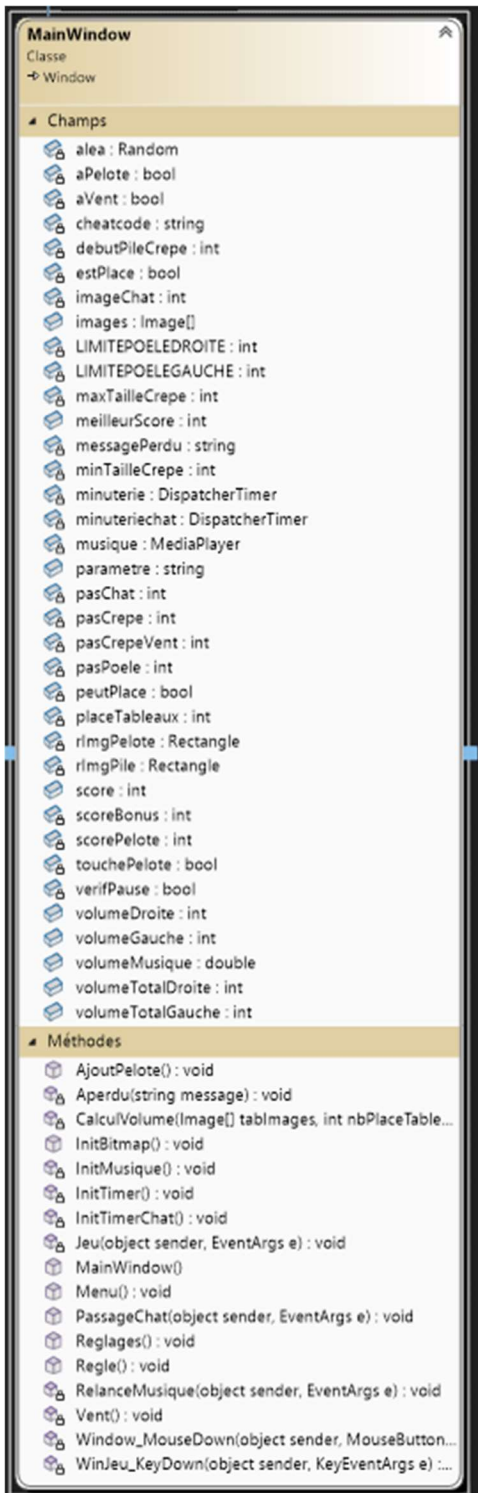
### 2.1. Présentation générale



La MainWindow contient la plupart des algorithmes avec la grande partie du jeu.  
Le Menu sert à faire le lien entre le trois classes MainWindow, Regle ainsi que Reglages.  
Les Regles expliquent simplement les règles du jeu.  
Les Reglages servent à mapper les touches.



### 2.2. Présentation détaillée MainWindow

	Groupe 2 Bouvier-Sobrin Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

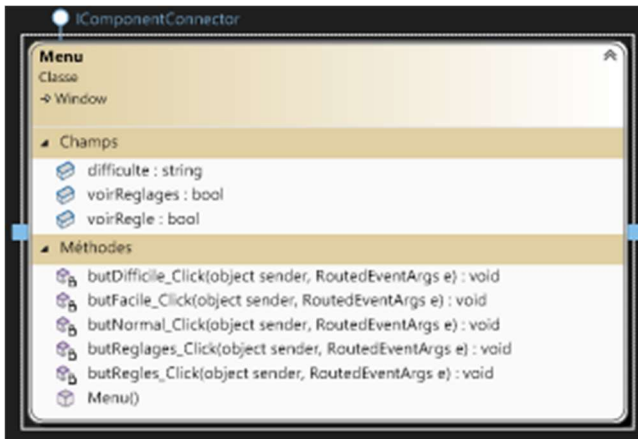


La MainWindow est composée de la plupart des variables composant le jeu, dont toutes les images ainsi que le tableau d'image de la pile de crêpe. On retrouve également l'ensemble des rectangles qui permettent la detection des colisions. On observe également la présence des timers qui permettent au jeu de s'actualiser au rythme de 60 FPS (Frames par secondes) et qui permettent donc de donner vie à notre jeu. Finalement on retrouve quelques booléens permettant pour la plupart de lier les touches au jeu.



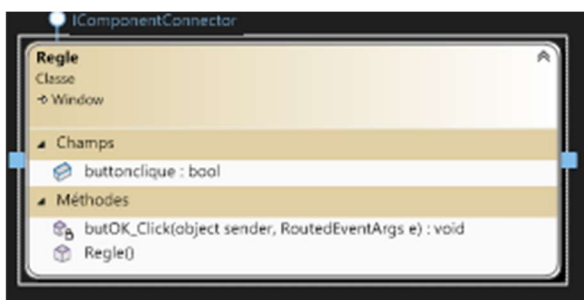
 IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

### 2.3. Présentation détaillée Menu





Cette classe permet d'ouvrir toutes les autres classe c'est donc la première fenêtre que l'on affiche afin de pouvoir naviguer parmi les différents menus. Elle permet également de choisir la difficulté du jeu parmi trois possibles : Facile, Normale et difficile. Cela change quelques paramètres tel que la vitesse de la poêle.

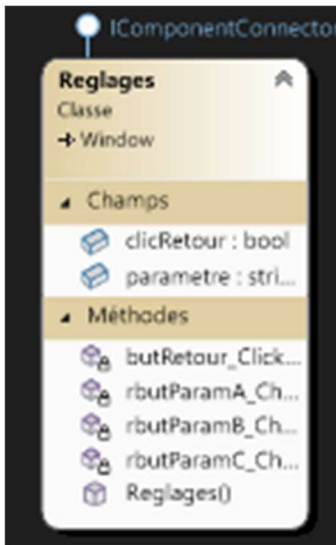
### 2.4. Présentation détaillée Regle



Ce menu est assez explicite puisqu'il contient tout simplement les règles du jeu avec un bouton « OK » permettant le retour en arrière vers le menu.

### 2.5. Présentation détaillée Reglages

 IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		



Nous Pouvons enfin voir le menu réglages qui permet de mapper les touches parmi trois mappages prédéfinis représentés par les variables Parmètre A, B ou C et avec de nouveau un bouton « OK » permettant le retour en arrière vers le menu.

## 3.Partie Algorithmie

### 3.1. Mouvement de la poêle

**On souhaite ici faire un mouvement de va et viens avec la poêle.**

**Emplacement :** Méthode `Jeu` (ligne 473 et suivantes)

- Déplace la poêle, en vérifiant les limites latérales.
- Gère les collisions avec les bords de l'écran.

**Pseudo-code :**

```



Si crêpe n'est pas placée
    Mettre à jour la position horizontale de la poêle
Fin si
Sinon, Si
    Continuer d'avancer
Fin Sinon, Si
  
```

**Débogage :**

```
Console.WriteLine($"Position crêpe : ({Canvas.GetLeft(imgPoele)})");
```

*On fait ainsi 2 tests par 16 millisecondes, soit 120 tests par secondes.*



	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

### 3.2. Détection de la crêpe sur l'assiette

**Emplacement :** Méthode `Jeu` (ligne 540 et suivantes)

- Simule la chute lorsqu'une crêpe est lâchée.
- Gère les collisions avec la pile de crêpes.

**Pseudo-code :**

```

Si crêpe est pas en collision avec la pile
    Mettre à jour la position verticale de la crêpe
Fin Sinon Si
Si collision avec la pile
    Ajouter la crêpe à la pile
    Calculer les volumes gauche/droite pour l'équilibre
    Mettre à jour le score (si il est supérieur au meilleur score on le remplace)
Fin Si
Sinon, si la crêpe tombe hors de la pile ou d'autres limites
    Déclencher la fin de partie
Fin Sinon si

```

**Débogage :**

```

Console.WriteLine($"Position crêpe : ({Canvas.GetLeft(imgCrepe)},
{Canvas.GetTop(imgCrepe)})");

```

```

Console.WriteLine($"Volume Gauche : {volumeTotalGauche}, Volume Droite :
{volumeTotalDroite}");

```

```

Console.WriteLine($"Collision détectée avec la pile : {rImgCrepe.Intersects(rImgPile)}");



```

*On effectue ainsi 4 tests (avec le score) ce qui nous fait 250 tests par secondes*

### 3.3. Détection de la pelote

Nous avons décidé de rajouter un élément afin de vider l'assiette de crêpes, ainsi lorsqu'une pelote apparait, il faut la toucher avec l'aide d'une crêpe afin de vider l'assiette sinon la pile atteindrait 42 crêpes ce qui causerait la perte du jeu.

**Emplacement :** Méthode `Jeu` (ligne 597 et suivantes)

	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

Pour chaque pelote

    Si la poêle touche la pelote

        Le jeu se stoppe

        L'animation du chat se lance

        L'assiette se vide

    Fin Si

Fin Pour

### Débogage :

```
Console.WriteLine($"Collision détectée avec la pelote : {rImgCrepe.IntersectsWith(rImgPelote)}");
```

*Le test n'effectue que 1 tests toutes les 16 millisecondes, soit environ 62 tests par secondes*

## 3.4. Calcul des volumes gauche/droite de la pile de crêpes

Nous avons également décidé de rajouter un calcul des volumes afin de simuler la déséquilibre de la crêpe.

**Emplacement :** Méthode `CalculVolume` (ligne 434 et suivantes)

Création d'une ligne invisible pour simuler le milieu de l'assiette

Calculs des points gauches et droite de la crêpe

Calculs des longueurs et largeurs de chaque partie de la crêpe (à droite et à gauche de la ligne)

Si crêpe toute à gauche

    Calcul du volume entier de la crêpe à gauche

Fin Si

Sinon, si toute à droite



    Calcul du volume entier de la crêpe à droite

Fin Sinon, Si

### Débogage :

```
Console.WriteLine($"numéro crêpe {nbPlaceTableau}: Volume gauche = {volumeCrepeGauche},  
Volume droite = {volumeCrepeDroite}");
```

*Le test effectue 2 tests toutes les 16 millisecondes, soit 125 tests par secondes*

	Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor	SAÉ S1_01_02 Développement en C# Jeu WPF	
	NOM DU JEU		

### 3.5. Gestion de la perte du joueur

Nous avons également décidé de rajouter un calcul des volumes afin de simuler le déséquilibre de la crêpe.

**Emplacement :** Méthode `Jeu` (ligne 567 et suivantes)

Si (pile déséquilibrée OU crêpe hors de la pile OU trop de crêpes (42))  
 Déclencher la fin de partie avec un message spécifique  
 Réinitialiser les variables nécessaires

**Débogage :**

Console.WriteLine(\$"Fin de partie: {messagePerdu}");

*Le test effectue 15 tests toutes les 16 millisecondes, soit 937 tests par secondes. ON ajoute à cela la réinitialisation du tableau de 42 crêpe soit  $937 + 42 = 979$  tests par secondes*

## 4. Conception graphique

---

Indiquez si vos images, animations, sons sont des sources existantes, dans ce cas donnez leur provenance. Expliquez les retouches ou création que vous avez peut-être réalisé.

La plupart des images viennent de freepik ( <https://fr.freepik.com/> ), un site d'images libres de droit, mais certaines images telles que celles des arrières plans ont été générée avec des IA génératives, ici : ChatGPT ( <https://chatgpt.com/> ). La musique, elle, de Pixabay ( <https://pixabay.com/fr/> ), et est donc libre de droit également.

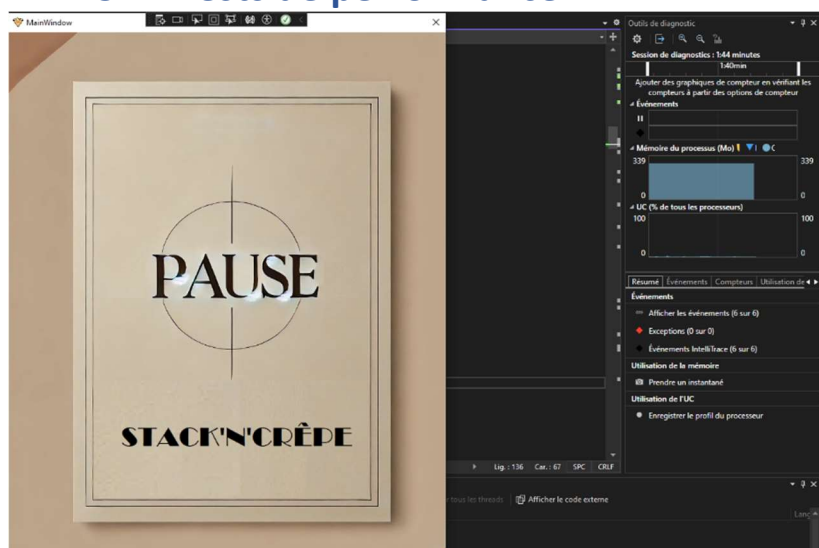
## 5. Cahier de recettes



### 5.1 Tests de validation

Nom	Fonctionnalités/ Dialogues/Classes	Etat
Bouvier	Largage de la crêpe	OK
Bouvier	Mouvement de la poêle	OK
Bouvier	Stockage de la pile	OK
Bouvier	Gestion des timers	OK
Bouvier	Gestion du son	Non achevé
Duchesnes	Gestion de perte	OK
Duchesnes	Gestion de la pelote	OK
Duchesnes	Gestion du vent	OK
Duchesnes	Gestion du score	OK
Duchesnes	Gestions des animations	Non achevé
Duchesnes	Gestion du code de triche	Ok

La première colonne n'est pas vraiment indicative puisque chacun de nous a contribué de la plus ou moins même manière à chacune de ses fonctionnalités.

### 5.2 Tests de performance



	<p>Groupe 2 Bouvier-Sobrino Maël Duchesne-Delapierre Victor</p>	<p>SAÉ S1_01_02 Développement en C# Jeu WPF</p>	
<p>NOM DU JEU</p>			

