GOLD RUSH

Mathieu Servonnet, Yanis Ranc

COMPTE RENDU DE LA SAE1.01 ET LA SAE 1.02

Bonjour, voici les étapes de conception et le rapport sur notre jeu vidéo d'aventure nommé « gold rush »

SAE1.01-1.02

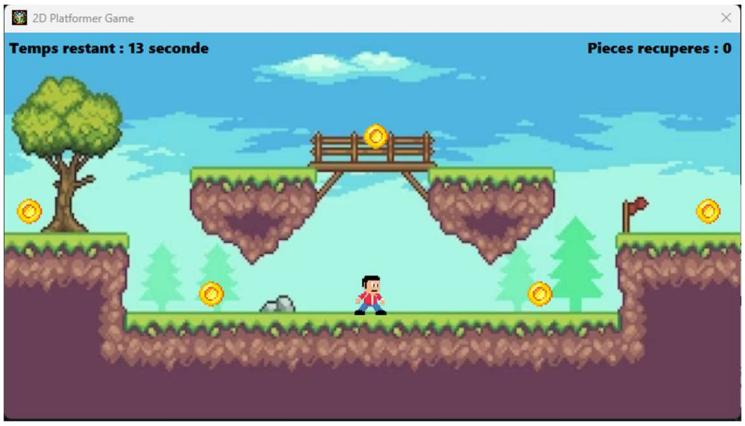
Sommaire

IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC	NOM DU GROUPE : TP 11 NOM DES ETUDIANTS : RANC Yanis, SERVONNET Mathieu	Saé S1_01_02 Développement en C# Jeu WPF	INFO
	NOM DU JEU : GOLD RUSH		

1.Présentation

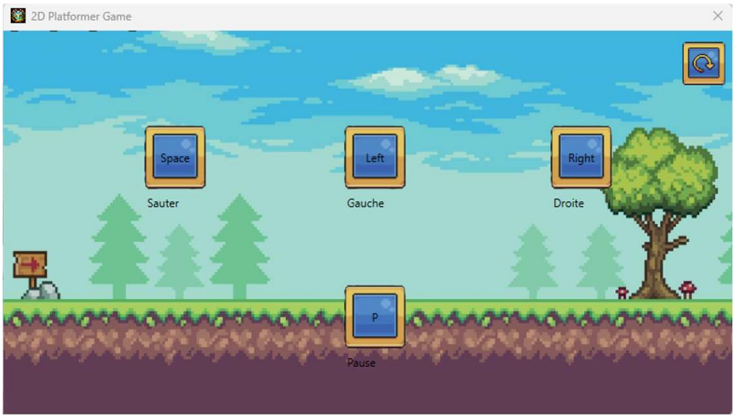
1.1. Description générale

Tout d'abord, notre jeu est de type aventure qui comporte 5 niveaux, avec un univers de jeu au style 8 bits. Au sein des différents niveaux présents, ayant chacun des caractéristiques tel que le changement de paysage (style plaine, désert) ou la présence d'ennemi, le personnage devra franchir tous les obstacles pour pouvoir récolter les pièces afin de finir le jeu. Dans le but de renforcer l'expérience du joueur, des effets sonores sur les pièces ont été rajouté ainsi que leur placement à des endroits stratégiques pour augmenter la complexité, difficulté de chaque niveau. Pour finir, le personnage a la capacité de sauté et courir afin de récupérer les pièces, juste un contact permet d'obtenir ces dernières.

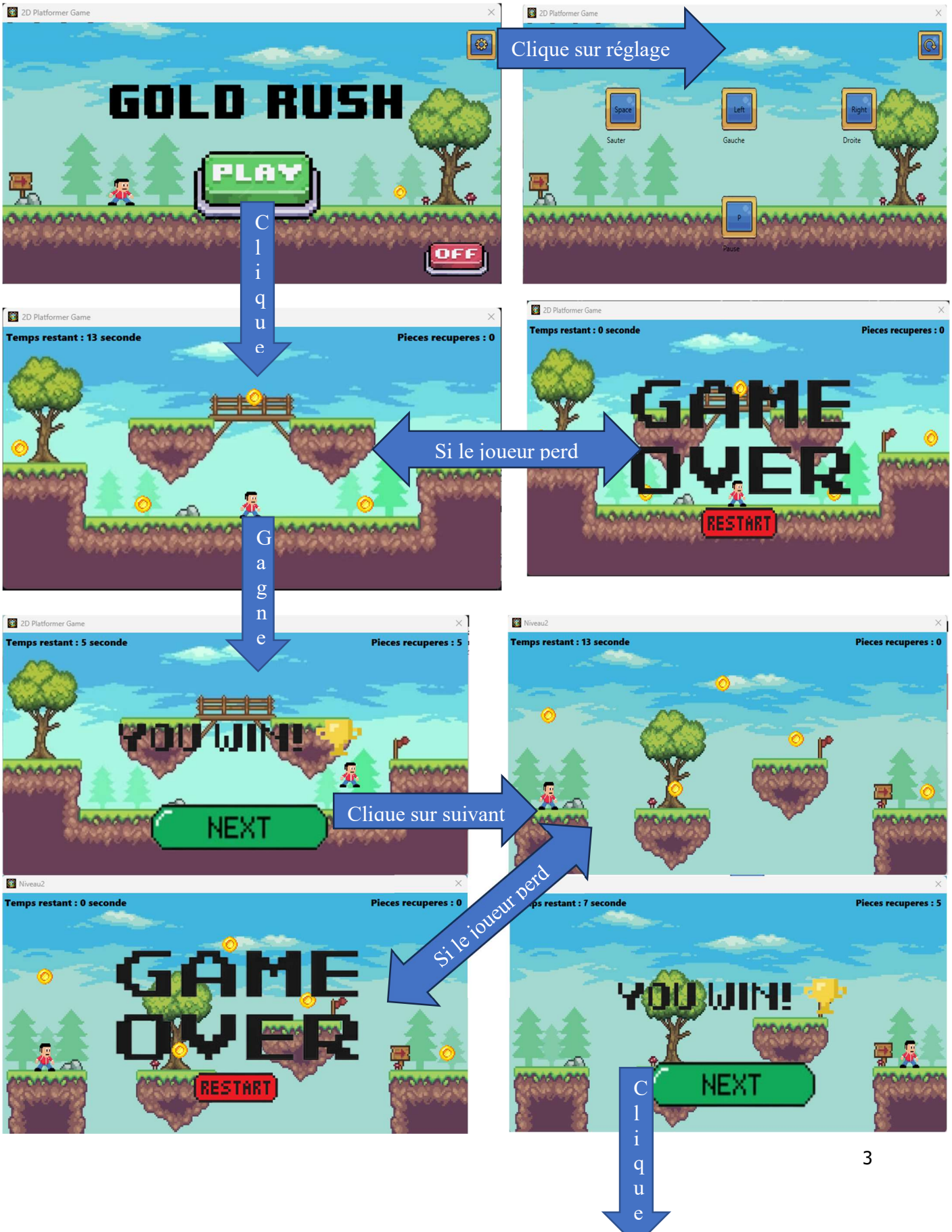


1.2. Règles du jeu

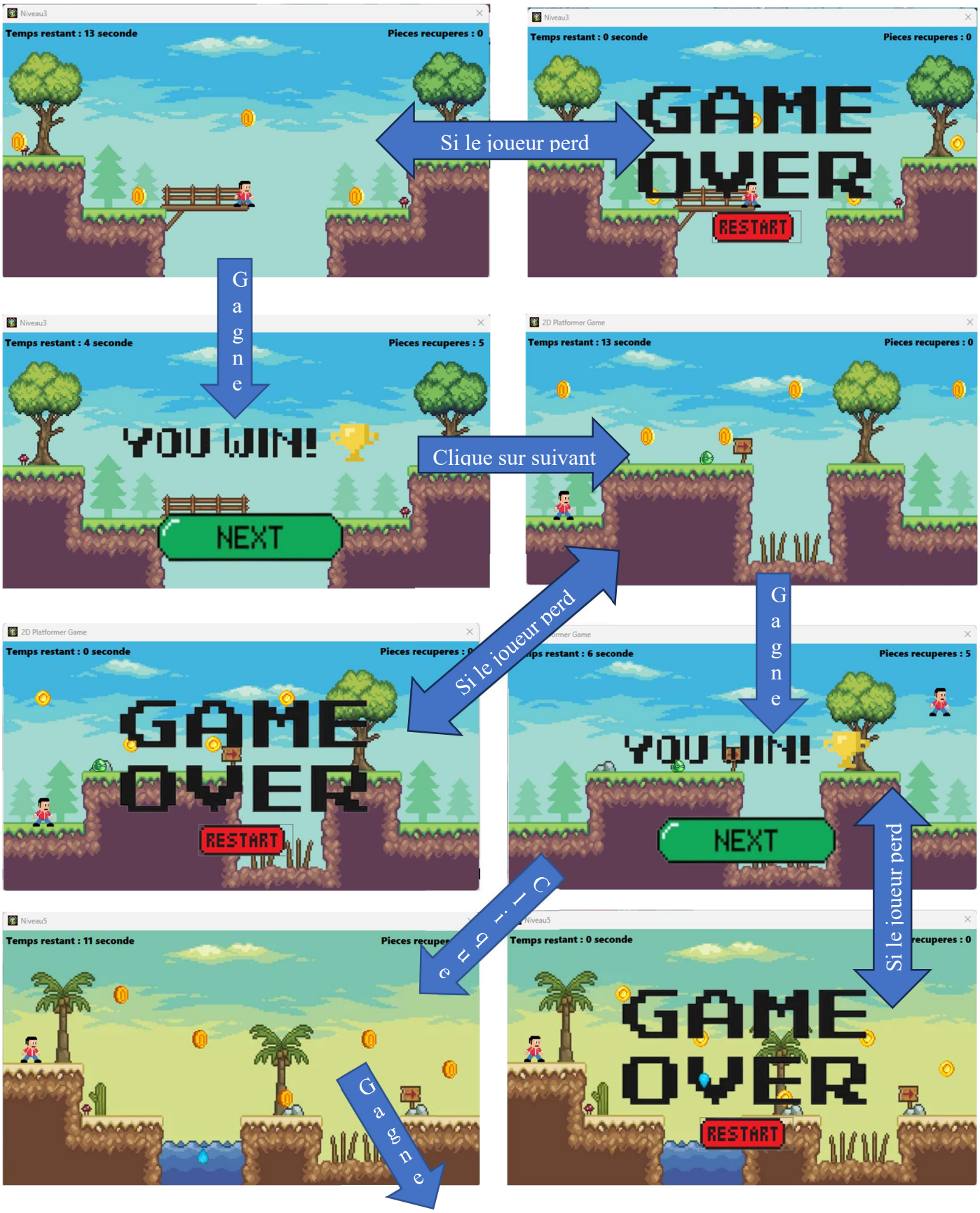
En premier lieu, le jeu comporte 5 niveaux avec pour but de récolter toutes les pièces en un temps déterminé, si le joueur ne parvient pas à récolter toutes les pièces dans les temps, c'est perdu (GAME OVER). Pour faire déplacer le personnage, le joueur pourra paramétrer ses propres touches à l'aide du menu dans la page d'accueil, sinon, les touches de bases du jeu, sont, les flèches droite et gauche pour se déplacer de droite à gauche et la barre espace pour sauter. Le jeu comporte également la fonctionnalité de le mettre en pause, touche de base P, qui pourra elle aussi être paramétré selon les préférences du joueur. Ensuite, pour les codes de triches, nous avons, F5 qui permet de passer au niveau suivant et F6 qui permet d'arriver directement au dernier niveau du jeu.

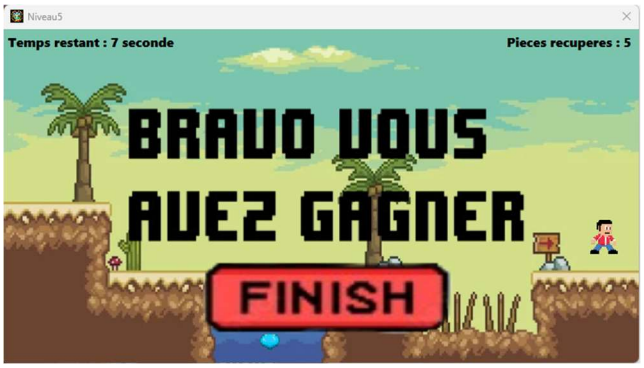


1.3. Cinématique des écrans



IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC	NOM DU GROUPE : TP 11 NOM DES ETUDIANTS : RANC Yanis, SERVONNET Mathieu	Saé S1_01_02 Développement en C# Jeu WPF	INFO
	NOM DU JEU : GOLD RUSH		





2. Conception – Diagramme de classe

2.1. Présentation générale des fenêtres

Nous avons donc la fenêtre principale (La MainWindow) qui est le niveau 1 au lancement du jeu elle appel la fenêtre Menu puis une fois le niveau 1 terminer on appelle le niveau 2 qui appel le niveau 3 quand le 2 est terminé puis le niveau 3 appel le niveau 4 et ainsi de suite jusqu'au niveau 5. Normalement nous aurions dû avoir uniquement la fenêtre niveau 1 et menu avec la génération des niveaux qui change uniquement dans le niveau 1 ce qui crée des niveaux différents et non crée des nouvelles fenêtres, ou si on le fait avec plusieurs fenêtres il aurait fallu faire en sorte que ce soit la fenêtre principale qui appelle tous les niveaux c'est à dire le niveau 1 qui appelle le menu et tous les autres niveaux.

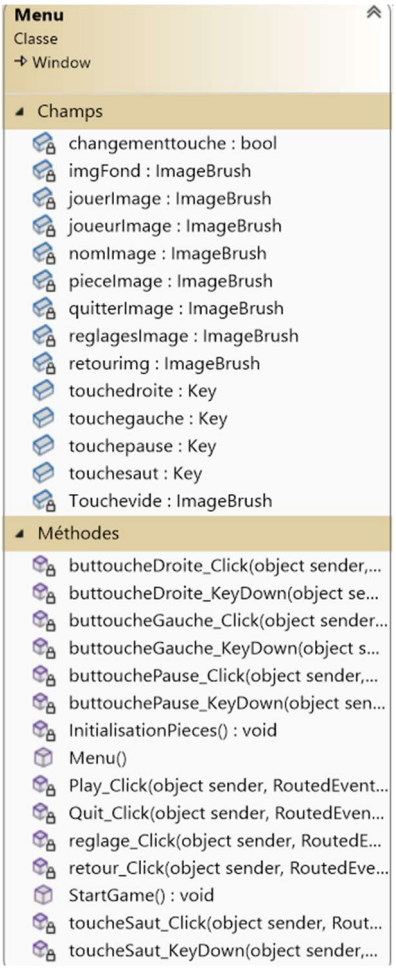
2.2. Présentation détaillée de la fenêtre Menu

Nous allons maintenant voir les méthodes présente au sein du menu. Il y a d'abord toutes les méthodes buttoucheDroite_Click, buttoucheGauche_Click, buttouchePause_Click, toucheSaut_Click et aussi les méthodes buttoucheDroite_KeyDown, buttoucheGauche_KeyDown, buttouchePause_KeyDown, toucheSaut_KeyDown ces 8 méthodes nous permettent de rendre possible le changement de touche pour le joueur.

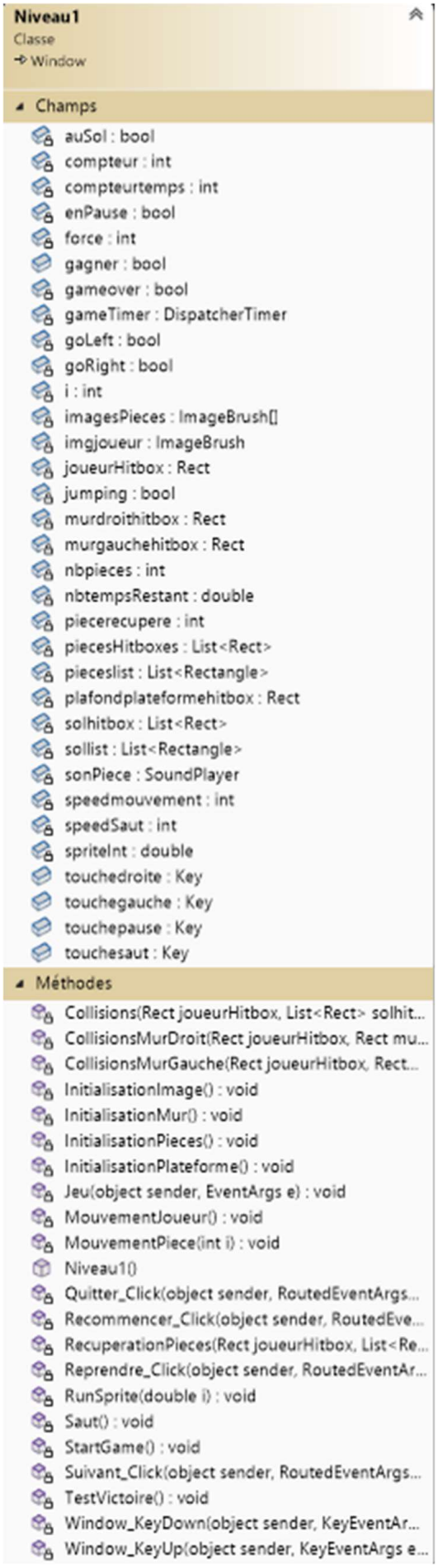
Pour la méthode InitialisationPieces elle nous permet d'afficher la pièce présente sur la page d'accueil. Pour la méthode Play_Click elle correspond au clique sur le bouton jouer ce qui va lancer le jeu elle est donc indispensable car elle va permettre de fermer le menu et de commencer à jouer.

Pour la méthode Quit_Click quant à elle, elle correspond au bouton off en bas à droite, il permettra d'éteindre le jeu vidéo si le joueur veut arrêter.

Pour les méthodes reglage_Click et retour_Click elle permette d'ouvrir les réglages pour reglage_Click ce qui est essentiel pour que le joueur puisse changer ces touches et retour_Click permet de fermer le menu réglage et de revenir sur l'affichage avec le bouton qui permet de jouer.



2.3. Présentation détaillée de la fenêtre Niveau1



Tout d'abord, pour les méthodes InitialisationImage, InitialisationMur, InitialisationPieces et InistialisationPlateforme elles servent à créer et à instancier toutes les listes de hitboxes et d'éléments tel que les sols, les murs et les pièces et aussi à remplir certains éléments avec les bonnes images.

Puis pour les méthodes Collisions, CollisionMurDroit, CollisionMurGauche elle vont tester toutes les collisions entre le joueur et les murs, les sols ce qui va pouvoir permettre de l'empêcher de traverser les murs ou les sols. Elles permettent donc de remettre le joueur à la bonne position.

Pour la méthode Jeu est le moteur de jeu c'est elle qui va appeler en boucle d'autre méthodes comme les collisions, la récupération des pièces ou encore le test de la victoire.

Quant à la méthode MouvementJoueur va s'occuper des mouvements de gauche a droite du joueur en fonction des touches appuyé.

Ensuite pour MouvementPiece quant à elle va s'occuper de changer les images de la pièce pour que l'on est l'impression qu'elle tourne sur elle-même.

Pour les 4 méthodes Quitter_Click, Recommencer_Click, Reprendre_Click et Suivant_Click correspondent chacune a un bouton respectivement sortir du jeu, recommencer le niveau, le reprendre si on a mis pause et passer au niveau suivant si on a fini le niveau actuel.

Puis pour la méthode RecuperationPieces elle gère la récupération des pièces ainsi que le score et supprimer les pièces une fois qu'elles sont récupérées.

Par la suite la méthodes RunSprite nous permet de changer les images du personnage en fonction de ce qu'il fait s'il est immobile s'il va à gauche ou à droite.

Puis pour Saut elle gère le saut avec la hauteur du saut ainsi que sa vitesse.

Ensuite pour la méthode StartGame comme son nom l'indique elle va lancer le jeu en appelant notamment toutes les méthodes d'initialisation, en commençant le timer, en plaçant le joueur etc.

Quant à la méthode TestVictoire, comme son nom l'indique elle va tester si l'on gagne on si l'on perd et dans les deux cas stopper le timer et afficher les Canvas adéquats a la situation.

Pour finir les 2 méthodes Window_KeyDown et Window_KeyUp elle vont permettre de gérer l'appui sur les touches en renvoyant des booléens suivant les touches sur lesquelles on appuie.

Pour finir, les autres niveaux sont semblables à celui-ci avec juste pour le niveau 4 et 5 la création et la gestion de l'ennemi.

3. Partie Algorithmie

3.1. Détection des collisions pièces avec joueur

On a décidé de tester toutes les pièces avec le joueur :

On crée une liste d'index
Pour chaque pièce dans la liste d'hitboxes de pièces
    On teste la collision entre l'hitbox du joueur et celle des pièces
    S'il rentre en collision avec la pièce
        On ajoute l'index dans la liste d'index
        On enlève un au nombre de pièce
        On joue le son de récupération de pièce
        Et on ajoute un au nombre de pièce récupéré par le joueur
    Fin
Fin
Pour chaque élément dans la liste d'index
    On supprime les pièces du Canvas
    On supprime les pièces de la liste
    Et on supprime les hitboxes des pièces de la liste
Fin

Cet algorithme est présent dans Niveau1.xaml.cs au sein de la méthode RecuperationPieces (ligne 326). Pour tous les niveaux 1,2,3,4, et 5 on peut compter 5 sols par niveau donc 5 tests au maximum toutes les 20ms, soit 250 tests par seconde.

3.2. Détection des collisions sol avec joueur

On veut tester les collisions du joueur avec chaque sol :
Pour chaque sol dans la liste d'hitbox de sol
    On teste la collision entre l'hitbox du joueur et celle des sols
    Si elles sont en collision
        On modifie la hauteur du joueur
        On dit que la vitesse de saut est nul
        Et que le joueur ne saute pas
        Et qu'il est au sol
        On ajoute 1 à l'index pour les images du personnage

Si cet index est supérieur a 4
On le remet a 1
Fin
On appelle la méthode qui modifie les images
Fin
Fin

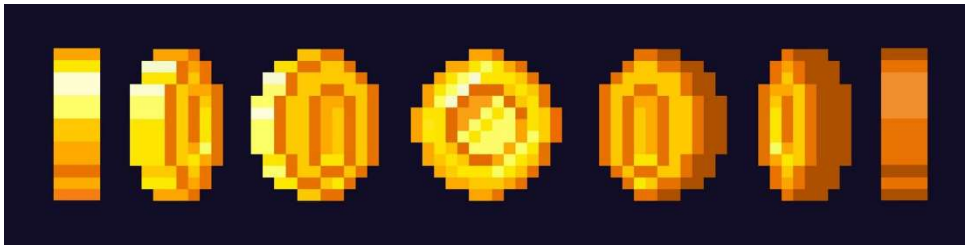
Cet algorithme est présent ans Niveau1.xaml.cs au sein de la méthode Collision (ligne 362)

Pour tous les niveaux 1,2,3,4, et 5 on peut compter 4 sols par niveau donc 4 tests au maximum toutes les 20ms, soit 200 tests par seconde.

4. Conception graphique

4.1. Conception des Pieces

Premièrement, toutes les images choisit ont été prisent à partir d'internet, ici pour les pièces nous avons choisi l'image ci-dessous et nous les avons redécoupés afin d'obtenir le rendu souhaité. Nous avons, ensuite, supprimé le fond violet pour obtenir les pièces seules puis nous les avons redimensionnés à la taille souhaiter.
https://www.freepik.com/premium-vector/pixel-game-coins-animation-golden-pixelated-coin-16-bit-pixels-gold-video-games_31082627.htm



4.2. Conception du personnage

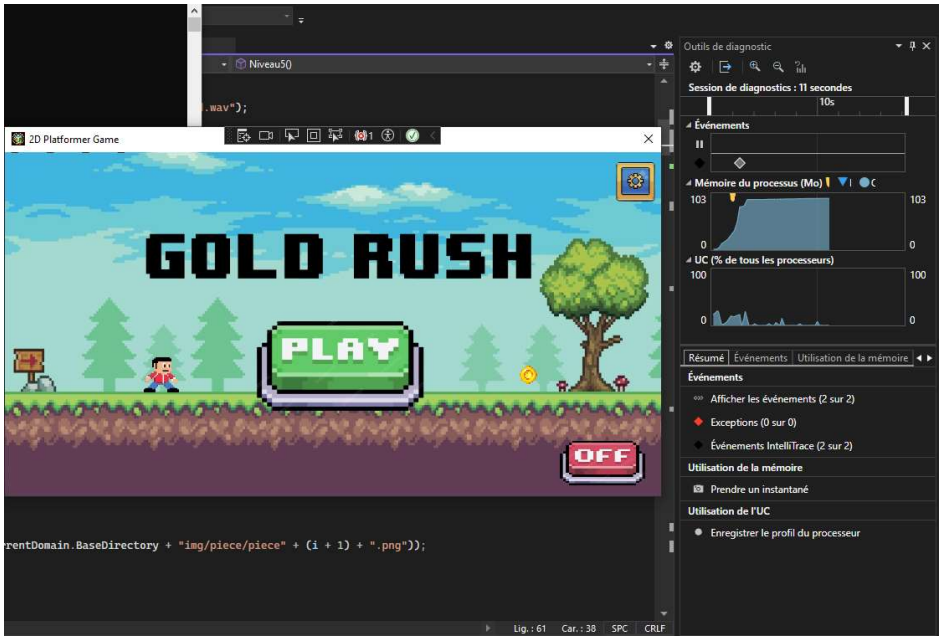
Tout d'abord, le personnage choisit était à la base un fichier GIF (une courte vidéo d'animation), nous avons, alors, redécouper en plusieurs images différentes afin d'obtenir les actions voulues. Ensuite, le fond violet a été supprimé et nous avons aussi redimensionné toutes les images aux bonnes tailles.
https://dribbble.com/shots/5663255-Avatar-Run-Jump-Sequence

5. Cahier de recettes

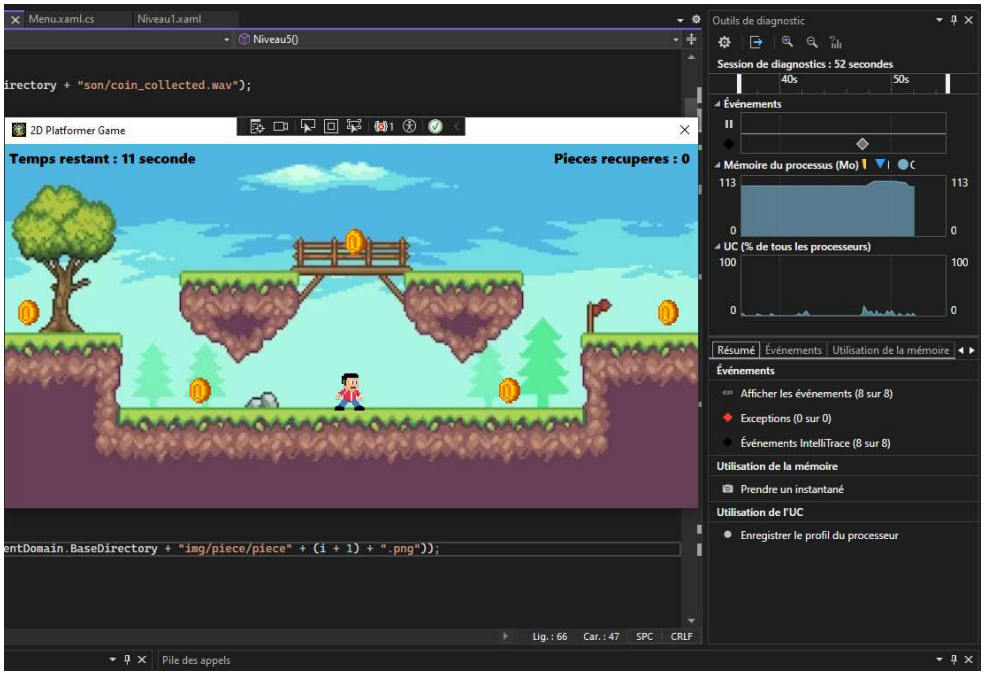
5.1 Tests de validation

Nom	Fonctionnalités/ Dialogues/Classes	Etat
Ranc	Changement de touche	Bug marche que pour niv1
Servonnet	Récupération des pièces et score	OK
Servonnet	Collision joueur avec sol	OK
Servonnet	Collision joueur avec mur	OK
Ranc	Collision joueur avec ennemi	OK
Ranc	Fenêtre menu / niveau4/ niveau5	OK
Servonnet	Niveau1 / Niveau2 / Niveau3	OK
Servonnet	Test victoire ou défaite	OK
Ranc	Recommencer/ Suivant / Quitter / Pause	OK

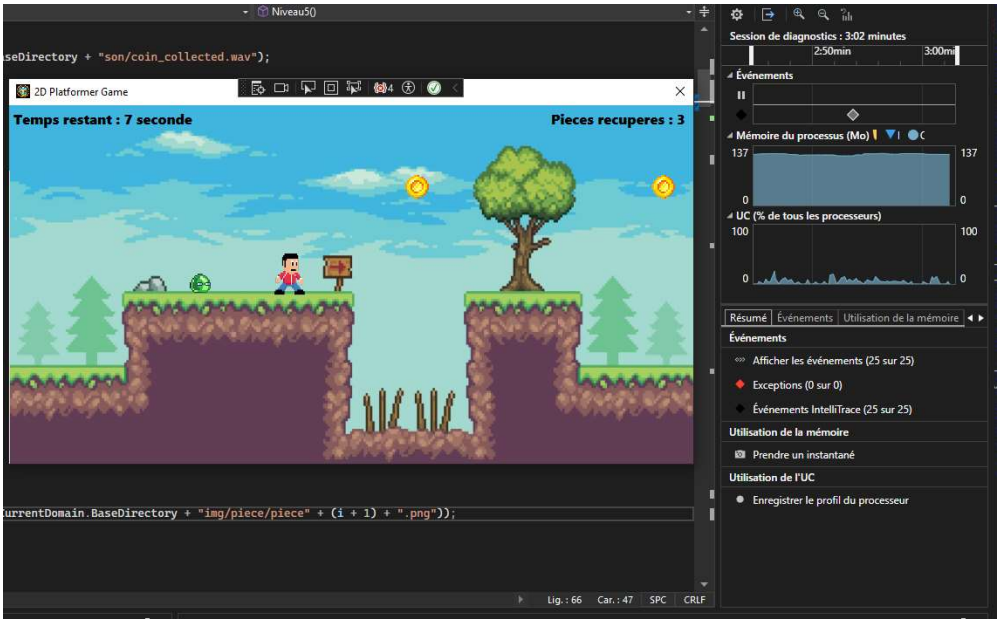
5.2 Tests de performance



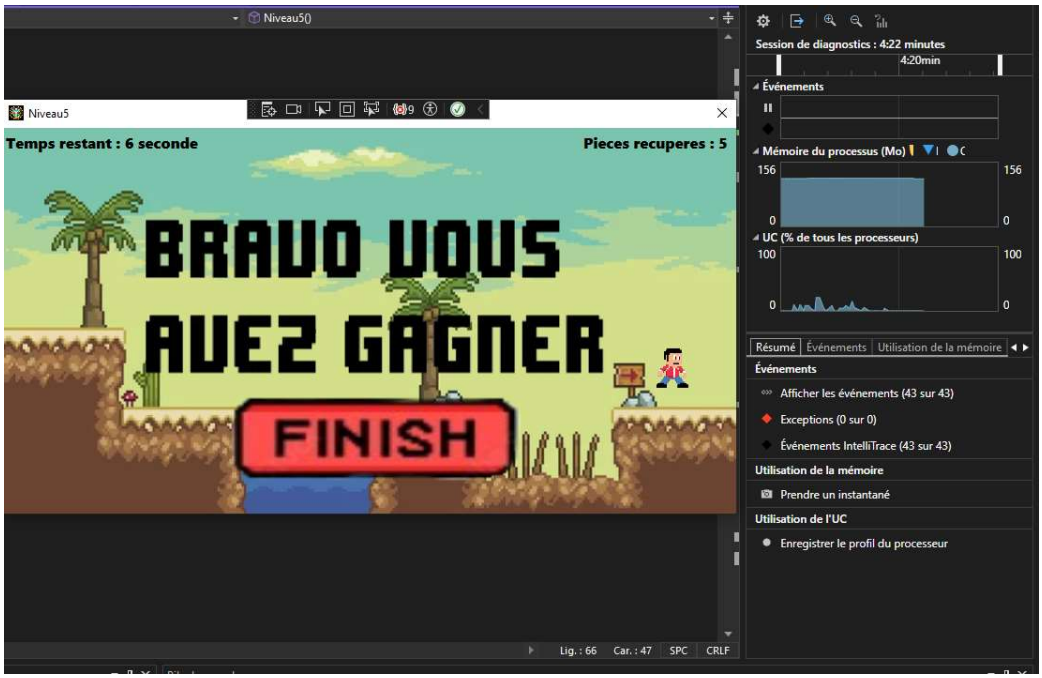
Lorsque l'on est au sein du menu on voit que le processus utilise 103Mo.



Lorsque l'on joue cette fois-ci on varie très peu avec 113Mo utilisé par le processus.



Lorsque l'on joue cette fois-ci avec un ennemi au sein du niveau on varie encore très peu avec 137Mo utilisé par le processus.



Lorsque l'on finit le jeu au niveau 5 on monte jusqu'à 156Mo par le processus.

Dans l'ensemble on voit que le processus consomme entre 100 et 160Mo ce qui ne reste pas très élevé.