Table des matières

[1 Présentation 2](#_Toc124168214)

[1.1 Description générale 2](#_Toc124168215)

[1.2 Règles du jeu 2](#_Toc124168216)

[1.3 Cinématique des écrans 2](#_Toc124168217)

[2 Conception – Diagramme de classe 3](#_Toc124168218)

[2.1 Présentation générale 3](#_Toc124168219)

[2.2 Présentation détaillée Game1 4](#_Toc124168220)

[*2.3* Présentation … 4](#_Toc124168221)

[3 Partie Algorithmie 5](#_Toc124168222)

[3.1 Explications 5](#_Toc124168223)

[3.2 Extrait de code 5](#_Toc124168224)

[4 Conception graphique 5](#_Toc124168225)

[5 Cahier de recettes 6](#_Toc124168226)

[5.1 Tests de validation 6](#_Toc124168227)

[5.2 Tests de performance 6](#_Toc124168228)

# Présentation

## Description générale

## 

Genre et principe du jeu, but du joueur.

Impressions écran, avec des phrases introductives et explicatives.

Tantopie est un RPG fantastique médiéval. Le but du joueur est d’arriver dans la salle du trône pour pouvoir assister au couronnement de son ami. Pour cela, il doit se déplacer au travers de plusieurs maps pour y arriver. Sur son chemin, le joueur devra se confronter à plusieurs ennemis et devra les battre pour pouvoir continuer sa découverte des lieux

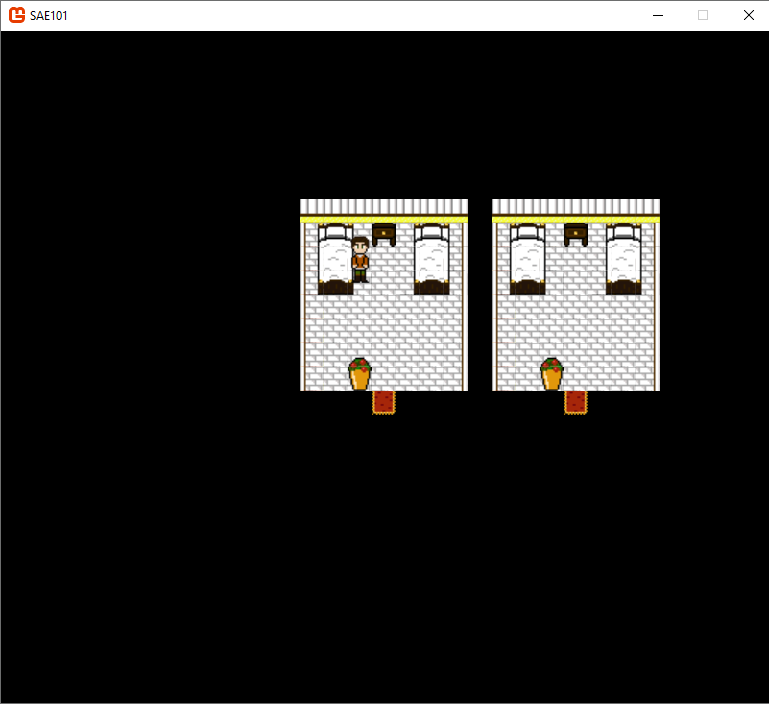
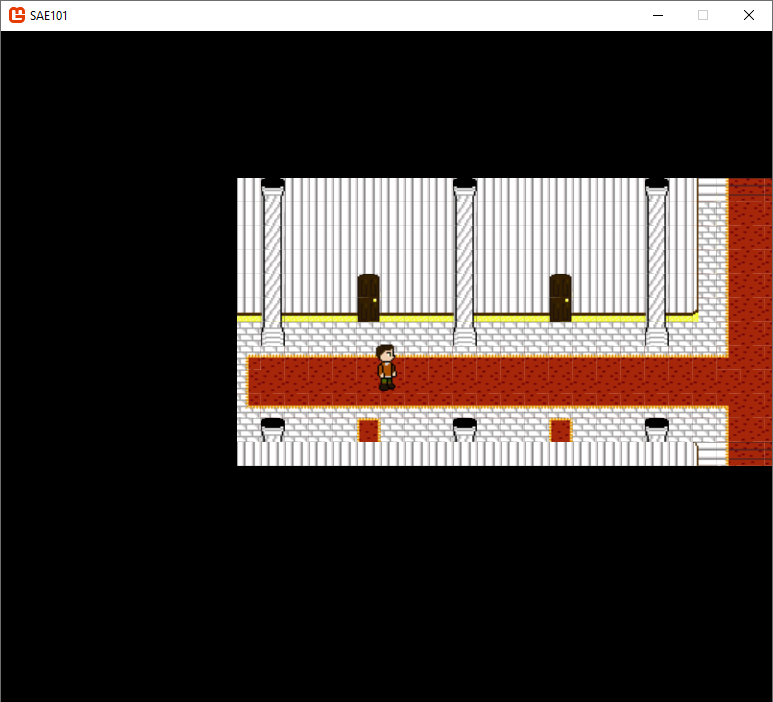
## Règles du jeu

Règle du jeu détaillée ainsi que la description des touches ou autre nécessaire pour jouer (code triche par exemple …)

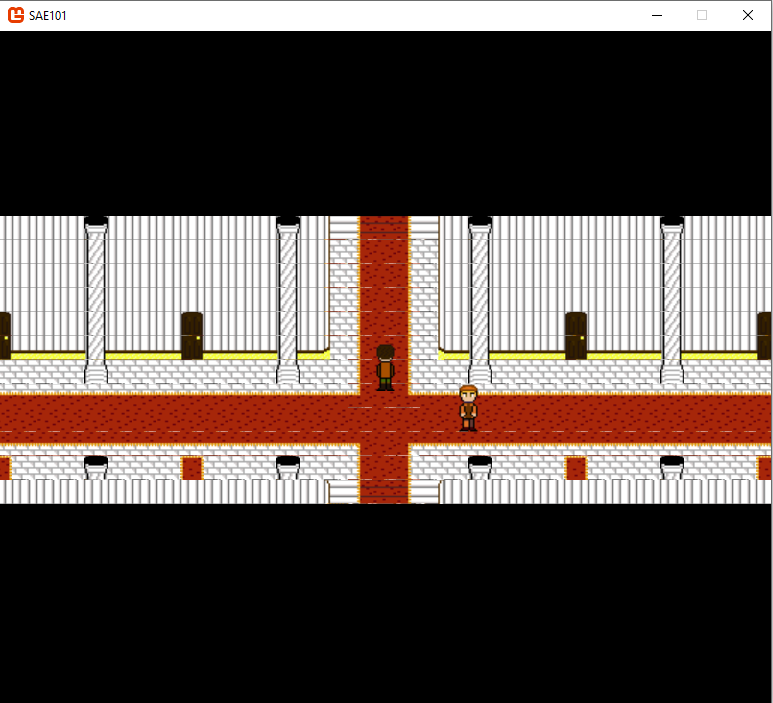
La touche W sert à valider, permet de passer les différentes boites de dialogue et d’activer les différents objets présents dans le jeu. La touche X permet de revenir en arrière lorsque le joueur se situe dans le menu, les options et lors d’un combat.

Les déplacements sont gérés par le pavé directionnel.

## Cinématique des écrans



Arrivé au seuil rouge



Arrivé aux escaliers du haut

# Conception – Diagramme de classe

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

## Présentation générale

Faites une impression écran générale (sans le détail des classes) accompagnée d’explications synthétiques pour chaque classe. N’hésitez pas à regrouper et découper si schéma trop grand.

Aide : sur votre projet, « Ajoutez un nouvel élément / Diagramme de classes » puis faites glisser vos classes depuis l’explorateur de solution dans la fenêtre.

Justifiez vos choix de conception : expliquez et mettez en évidence si vous avez fait des classes pour améliorer, factoriser votre code.(Ex : classe Sprite ou Personnage, ….)

Ex :

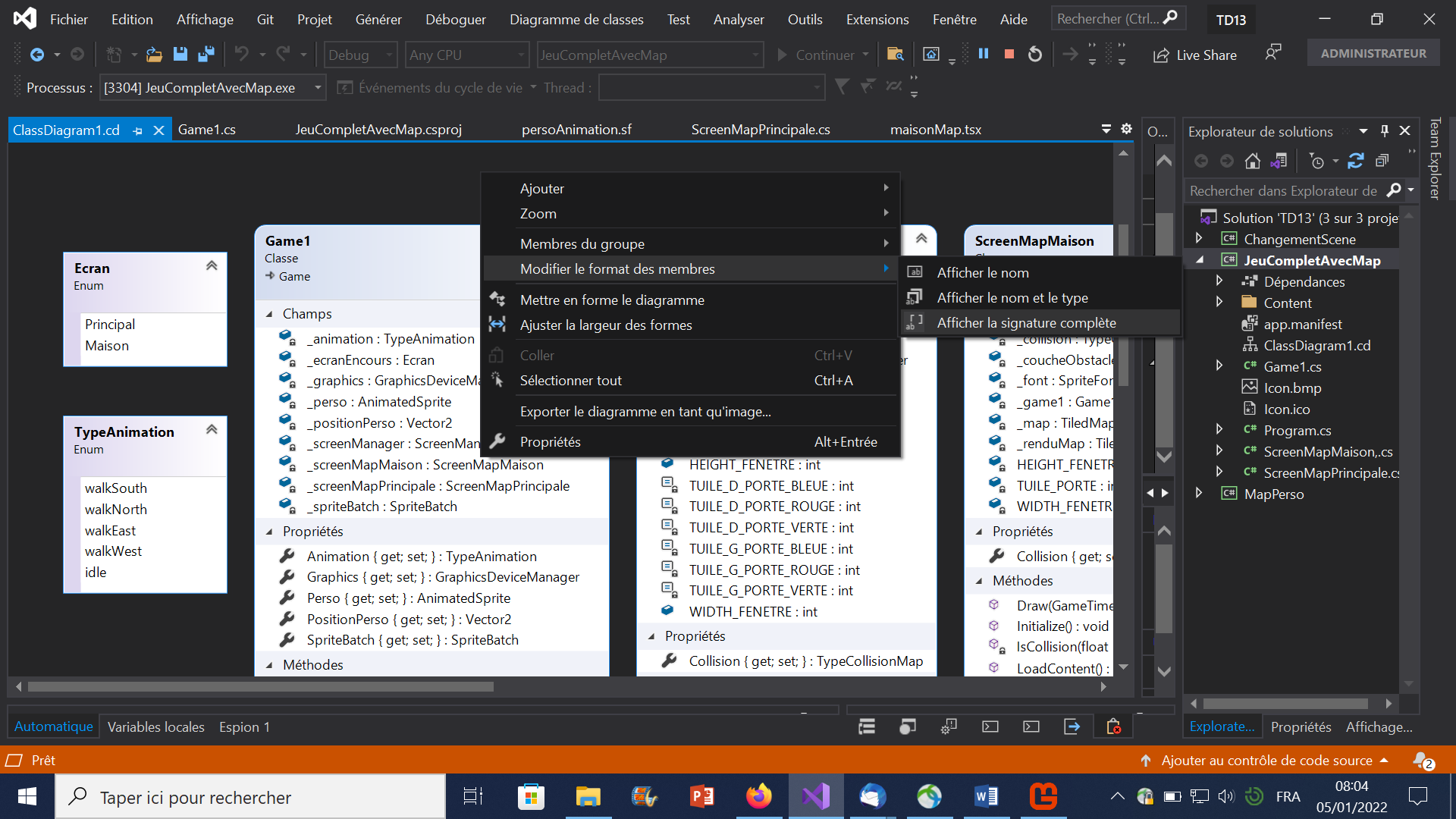
Game1 est le point d’entrée : elle contrôle tous les enchainements des différents écrans ainsi que les pauses ou sortie du jeu.

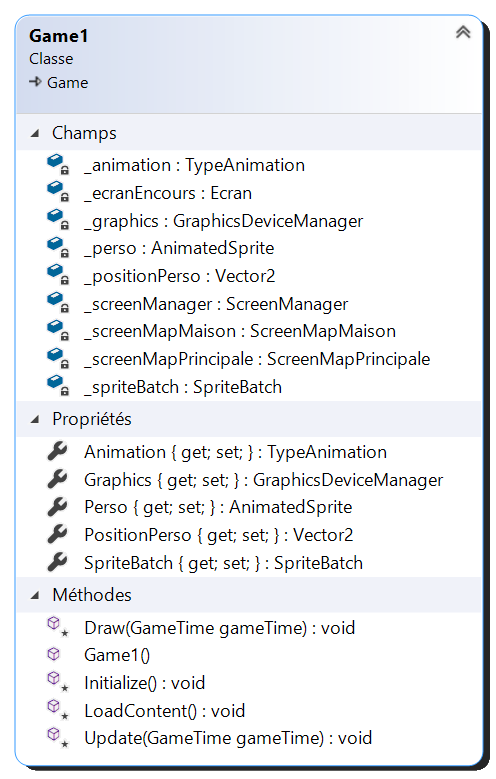
****

## Présentation détaillée Game1

Puis faites une impression écran détaillée de chaque classe avec leur signatures accompagnée d’explications un peu plus détaillées.

Aide : clic droit sur le fond de fenêtre vide puis Menu ci-dessous :

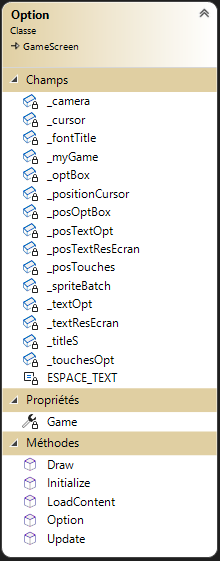




Game1 : instancie les 2 GameScreen : ScreenMapPrincipale et ScreenMapMaison .

Game1 charge en tout 1er ScreenMapPrincipale….. Elle contient 9 champs :

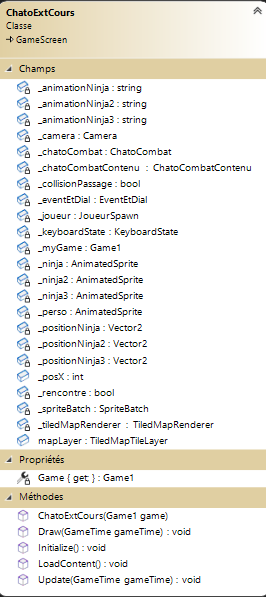
* \_screenManager : c’est un objet de classe ScreenManager qui sert à gérer le chargement des GameScreen ….

**

Option : permet d’afficher l’écran des options. Cette classe est chargé après l’écran de titre.

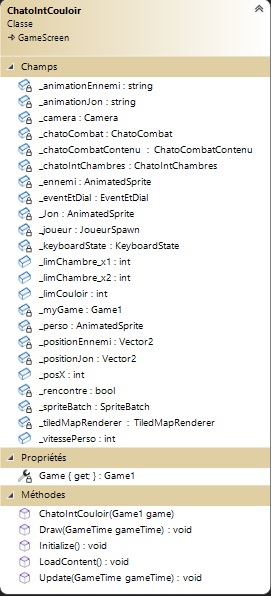
Elle contient 16 champs :

* \_screenManager : c’est un objet de classe ScreenManager qui sert à gérer le chargement des GameScreen ….

ChatoExtCours : permet d’afficher la map de la cour. Cette classe est chargé au moment quand l’utilisateur sort du château.

Elle contient 23 champs :

* \_**animationNinja(2-3)** : c’est un objet de classe String qui permet de gérer les animations de trois ennemis ninjas présents sur la map.
* **\_camera**, **\_chatoCombat**, \_**chatoCombatContenu**, \_**joueur**, \_**eventEtDial**, \_ **myGame** : permettent la liasion avec les autres classes.
* **\_collisionPassage** et **\_rencontre** : ce sont deux Booléens qui permettent d’avoir les messages (et combat) qu’une seule fois.
* \_**keyboardState** : objet de classe KeyboardState permettant de récupérer l’état des touches.
* **\_ninja(2-3)** et \_perso : objet de classe AnimatedSprite qui permet d’avoir les ennemis et le héros sur la map.
* \_**positionNinja(2-3)** : ce sont des objets Vector2 avec lesquels on positionne les ennemis à ses places
* \_**posX** : objet Int avec lequel se passe le changement de maps.
* \_**SpriteBatch** : c’est un objet SpriteBatch permettant de dessiner les personnages, boîtes de dialogues et maps.
* \_**tiledMapRender** : objet TiledMapRenderer qui permet d’avoir la map
* **mapLayer** : objet TiledMapTileLayer avec lequel on gère les collisions.

ChatoIntCouloir : permet d’afficher la map du couloir. Cette classe est chargé au moment quand l’utilisateur sort de la chambre ou de la cour.

Elle contient 23 champs :

* \_**animationEnnemi, \_animationJon** : c’est un objet de classe String qui permet de gérer les animations de l’ennemi et de l’alié nomm é Jon.
* **\_camera**, **\_chatoCombat**, \_**chatoCombatContenu**, \_**joueur**, \_**eventEtDial**, \_ **myGame** : permettent la liason avec les autres classes.
* **\_collisionPassage** et **\_rencontre** : ce sont deux Booléens qui permettent d’avoir les messages (et combat) qu’une seule fois.
* \_**keyboardState** : objet de classe KeyboardState permettant de récupérer l’état des touches.
* **\_ninja(2-3)** et \_perso : objet de classe AnimatedSprite qui permet d’avoir les ennemis et le héros sur la map.
* \_**positionNinja(2-3)** : ce sont des objets Vector2 avec lesquels on positionne les ennemis à ses places
* \_**posX** : objet Int avec lequel se passe le changement de maps.
* \_**SpriteBatch** : c’est un objet SpriteBatch permettant de dessiner les personnages, boîtes de dialogues et maps.
* \_**tiledMapRender** : objet TiledMapRenderer qui permet d’avoir la map
* **mapLayer** : objet TiledMapTileLayer avec lequel on gère les collisions.

## Présentation …

# Partie Algorithmie

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

## Explications

Expliquez ici la logique et les algos choisis : mettez en évidence leur complexité.

Listez les points d’algo intéressants à mettre en valeur pour démontrer la qualité de votre travail ( avec emplacement dans votre projet : Fichier et numéro de ligne )

Il est possible d’utiliser des algorithmes existants (ex : A\*) dans ce cas la expliquer la façon dont vous les avez utilisés/codés.

## Extrait de code

Mettez ici des extraits de code commenté de la logique précédemment décrite.

# Conception graphique

Indiquez si vos maps, décors,sons sont des sources existantes, dans ce cas donnez leur provenance Expliquez les retouches ou création que vous avez peut être réalisés.

Les décors du Menu et des combats sont des images générées à l’aide d’une IA sur internet.

# Cahier de recettes

## 5.1 Tests de validation

Faites un listing des fonctionnalités, le nom de l’étudiant qui a travaillé dessus et son état :

|  |  |  |
| --- | --- | --- |
| Nom | Fonctionnalité | Etat |
| GIMENEZ | Menu principal | Non achevé/ Bug/ OK  Ok |
| Diard | Plateforme niveau 1 |  |
| Martin | Sprite Ennemi |  |

## 5.2 Tests de performance

A l’aide des outils de diagnostiques : vous prendrez quelques mesures (captures écrans) à des moments clefs de votre jeu de l’utilisation de la mémoire et du processeur. Vous commenterez bien évidemment les moments choisis et les mesures.

