**Journal de travail TPI**

**Journal réalisé par : Sacha Leone**

Classe : 3IND-3TPMa

Commencé le : 08.04.2025

Projet : Jeu de société « Qui est-ce » digital

Table des matières

[1. 08.04.2025 3](#_Toc196806991)

[Étapes réalisées 3](#_Toc196806992)

[Problèmes rencontrés 3](#_Toc196806993)

[Étapes suivantes 3](#_Toc196806994)

[2. 09.04.2025 4](#_Toc196806995)

[Étapes réalisées 4](#_Toc196806996)

[Problèmes rencontrés 4](#_Toc196806997)

[Étapes suivantes 4](#_Toc196806998)

[3. 11.04.2025 5](#_Toc196806999)

[Étapes réalisées 5](#_Toc196807000)

[Problèmes rencontrés 6](#_Toc196807001)

[Étapes suivantes 6](#_Toc196807002)

[4. 28.04.2025 7](#_Toc196807003)

[Étapes réalisées 7](#_Toc196807004)

[Problèmes rencontrés 7](#_Toc196807005)

[Étapes suivantes 7](#_Toc196807006)

# 08.04.2025

## Étapes réalisées

Lors de la réception et après la prise de connaissance du cahier des charges, la première étape à été d’organiser mon projet comme suit :

* Préparation d’un repos GitHub pour le projet à venir.
* Préparation des différents répertoires après « git clone ».
* Préparation des différents documents (Websummary, JT, Rapport, etc…).
* Préparation de la planification.

J’étais déjà en possession de canevas de documentation (Ex. : Rapport, JT, Planification) prêt à l’emploi.

Ensuite, j’ai résumé les objectifs principaux demandé, qui sont le minimum requis pour finir officiellement le produit.

Puis, j’ai listé les différents chemins que j’aurai pu emprunter pour réaliser le projet. Incluant donc les idées et leur faisabilité, les contraintes, le temps à disposition, les capacités réels.

## Problèmes rencontrés

Aucun problème rencontré.

## Étapes suivantes

Faire des recherches sur le sujet. (Randomisation de portrait, éviter les sosies, algorithme, etc…)

# 09.04.2025

## Étapes réalisées

Ayant dû réaliser ma soutenance TIP (Travail de maturité interdisciplinaire) je n’ai pas avancé sur le projet entre 8h10-9h00.

Mon planning étant encore incomplet, j’ai pris le temps de le terminer.

En faisant des recherches plus approfondies, j’ai trouvé un projet GitHub open-source appelé « Avataaaar » qui crée des portraits aléatoirement grâce à des images svg superposées.

Voici le site [getavataaars.com](https://getavataaars.com/) qui présente le fonctionnement.

Ce repos GitHub sera d’une grande aide puisqu’il contient déjà une bonne partie du fonctionnement que je recherche, je pourrai comprendre le fonctionnement et réaliser mon application de l’algorithme de création de portrait en C#.

## Problèmes rencontrés

Aucun problème rencontré.

## Étapes suivantes

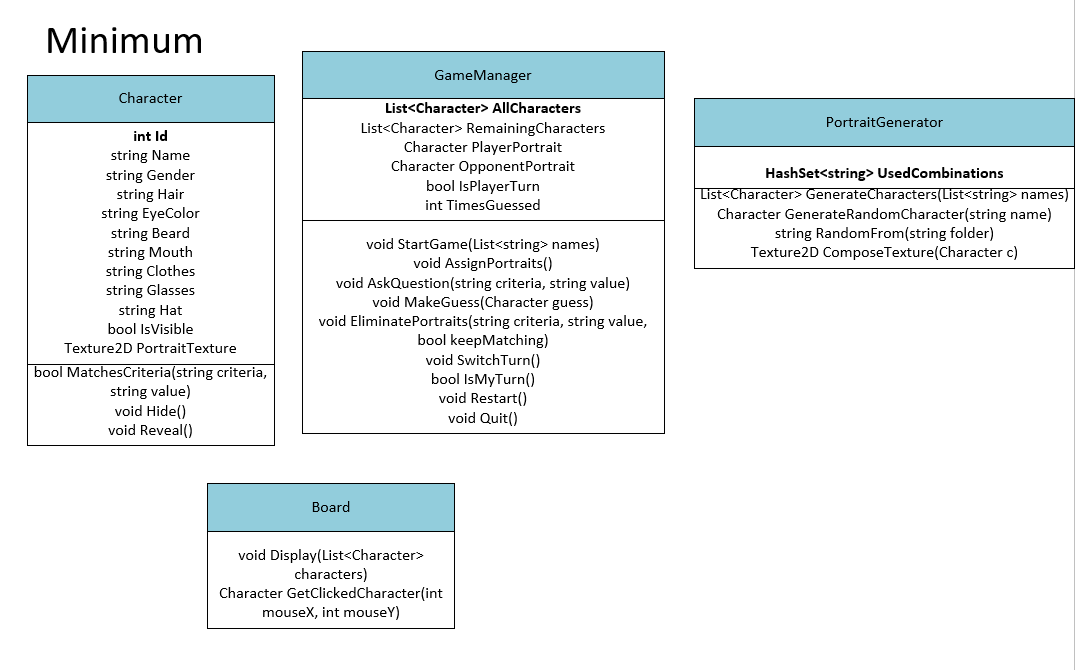
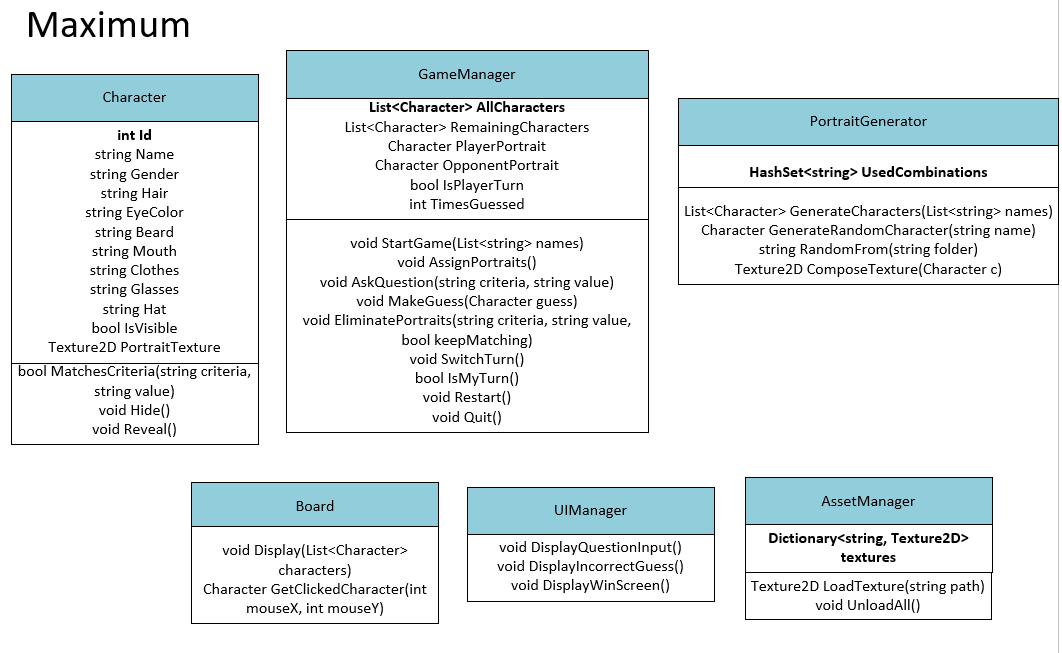
* Réflexions sur les ajouts faisables
* Choix de l'environnement de développement C#

# 11.04.2025

## Étapes réalisées

J’ai choisi de réaliser un projet avec Raylib en C# :

Raylib en C# (via Raylib-cs) permet de créer des jeux/applications 2D/3D simples avec gestion graphique, audio et entrées (clavier/souris/manette).

J’ai également créé deux diagrammes de classes pour définir les classes que je dois créer au minimum pour que mon « Qui est-ce ? » fonctionne. Les voici :

J’ai donc créé les quatre classes minimums, dans ma solution Raylib C# en application console, pour que le projet fonctionne. J’ai également importé les paquets :

* SkiaSharp
* Svg.Skia

Ces paquets me serviront pour convertir les images SVG en PNG et ensuite les faire afficher en texture par Raylib.

Mise à jour du planning et du JT.

## Problèmes rencontrés

Il est très compliqué de trouver un moyen de récupérer chaque partie de l’image SVG séparément. Par exemple uniquement : les yeux, la bouche, les cheveux dans leurs fichiers svg respectifs.

## Étapes suivantes

* Mise en place simple de l'algorithme de création de portraits sans sosies.

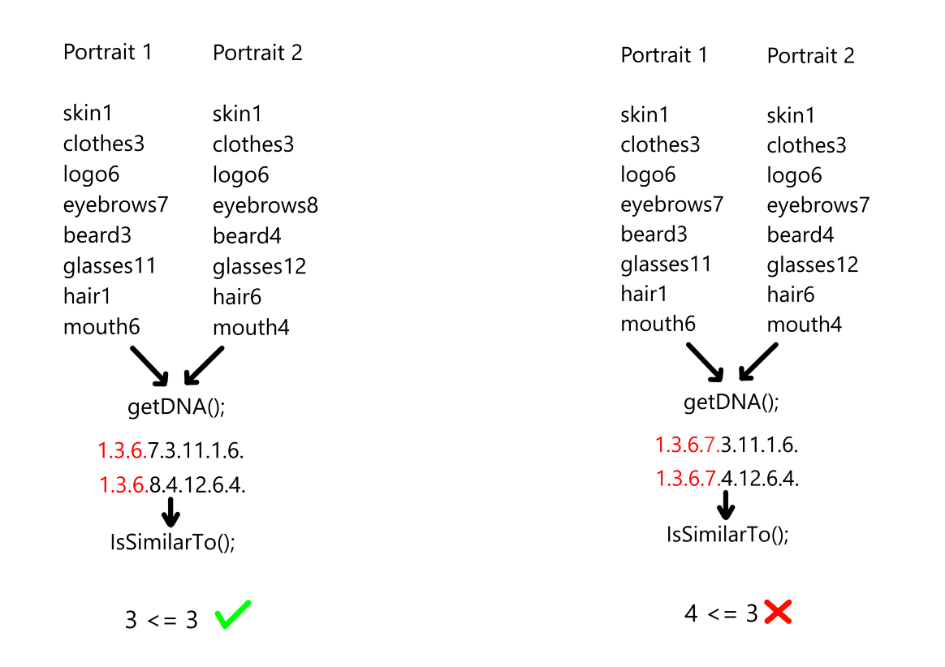
# 28.04.2025

## Étapes réalisées

La mise en place de l’algorithme de randomisation de portrait a été commencée et terminée, va suivre un aperçu de son fonctionnement.

Un portrait est défini par les propriétés suivantes :   
Id, Nom, Skin, Clothes, Logo, Eyebrows, Eyes, Beard, Glasses, Hair et Mouth.

Lorsque je choisi une texture aléatoirement dans un répertoire pour créer son portrait, je vérifie qu’il n’a pas plus de 3 propriétés en commun avec un autre portrait déjà créé. Pour récupérer les similitudes j’ai créé une méthode « GetDNA » qui renvoi le DNA ou la combinaison de chaque portrait. Puis je teste ses similitudes avec une méthode « isSimilarTo ».

Ces deux méthodes sont utilisées lors de la génération aléatoire, si un portrait a plus de 3 gènes en commun on regénère un portrait. Ci-dessous, un schéma qui illustre tout ce cheminement :

J’ai commencé par afficher des textes comme sur le schéma au lieu des textures, pour obtenir d’abord le fonctionnement souhaité pour ensuite le complexifier.

## Problèmes rencontrés

Aucun problème rencontré.

## Étapes suivantes

* Affichage des 48 portrait. 24 pour chaque joueur.
* Choisir un portrait à faire deviner.

# 29.04.2025

## Étapes réalisées

J’ai mis à disposition l’algorithme de création de portraits aléatoires sans sosies à ma nouvelle classe « PortraitGenerator » qui contient les méthodes :

* GeneratePortraits
* CreateRandomPortrait
* GetRandomAsset

« GeneratePortraits » appelle « CreateRandomPortrait » 48 fois pour les 48 portraits différents.

Si la génération respecte le maximum de gènes similaires alors on l’ajoute à la liste de portrait qui sera affichée plus tard.

« CreateRandomPortrait » lui, s’occupe de la récupération aléatoire **d’un fichier** **par** catégorie. (« GetRandomAsset ») Lorsque tout ces fichiers sont récupérés ils sont inscrits dans un nouveau portrait, qui sera vérifié à son tour.

## Problèmes rencontrés

* Oubli de passer les fichiers images en « Toujours copier » dans Visual Studio 2022  
  pour tester en Debug.
* Chemin d’accès différents, « \\ » au lieu de « / » à cause des fonctions de Raylib.

## Étapes suivantes

* Choisir un portrait à faire deviner.
* Travail UX/UI