Rapport Projet Long

ANDREOTTI Eric, BASSET Louis, BRÉGEOT Valentine, CORBELLARI Nolan,
DEHMANI Mohammed, ISTFALEN Hana, ROGER Victor, PICARD Arthur.

Département Sciences du Numérique - Première année
2022-2023

1 Rappel du projet

Notre projet long du cours de Technologie Objet porte sur une plateforme de révision à destination des écoliers, et de leurs parents, qui ont un accès privilégié au travail et aux progrès de leur enfant.

Plusieurs matières sont disponibles ainsi que plusieurs niveaux, avec à chaque fois des exercices variés et ludiques correspondant au programme scolaire actuel.

2 Actualités du projet

Nous avons rapidement compris qu'un projet concernant les élèves du CP à la Terminale était bien ambitieux et pas forcément pertinent pour un projet de ce type, nous avons donc décidé de nous limiter dans un premier temps au primaire. En effet, d'un point de vue scolaire cela nous a paru assez abordable, et il y a suffisamment de notions et de matières pour avoir du contenu.

Nous n'avons pas pu développer la plateforme au-delà du primaire, mais l'enjeu principal ici est de développer les "squelettes" des mini-jeux. Sur le principe, nous souhaitons simplement que les différents mini-jeux soient réutilisés en changeant les questions ou les éléments de question selon le niveau.

Depuis la dernière itération, nous avons pu avancer comme nous le souhaitions sur le projet, en prenant en compte les retours. Nous avons donc une version aboutie de E-Learner, incomplète par rapport à notre objectif initial, mais fonctionnelle et correspondant à nos attentes.

Depuis la première itération, nous avons revu toute l'architecture initiale afin de respecter le MVC, en concevant 2 ou 3 fichiers contenant la Vue, le Modèle, et le Controlleur.

Cette structure nous a permis de réfléchir collectivement à sa mise en place, et de favoriser le travail de groupe par la suite.

Nous avons par la suite réalisé les diagrammes UML du modèle que nous utilisons, nous avons codé le menu (paramètres, utilisateur, accès aux niveaux) et les mini-jeux (d'abord les mini-jeux de mathématiques, puis les autres, dont le quizz), et avons créé la base de donnée pour les questions et les réponses dans toutes les matières.

Ainsi, nous disposons désormais d'une plateforme permettant aux écoliers du CP au CM2 de réviser leur français, leurs mathématiques, leur histoire, leur géographie, ou encore leurs sciences. Ils ont accès à plusieurs mini-jeux variés et ludiques qui couvrent une grande partie du programme scolaire de ces niveaux.

2.1 Répartition du travail

Nous avons hésité entre une répartition par mini-jeux et une répartition plus "globale" par matière, et avons donc testé les deux options.

Le problème de la répartition par mini-jeux était que ce n'était pas assez clair pour nous, et que nous nous éparpillions trop. Nous avons donc réparti par matière pour que chacun sache précisément ce qu'il a à faire. Cela nous oblige également à communiquer énormément entre nous pour savoir les avancées des autres membres, et pour réutiliser les codes entre eux.[2]

Ainsi, nous avons tous une architecture commune à toutes les matières, ainsi que des mini-jeux communs, mais également des mini-jeux propres à une matière, réfléchies et codés par le groupe correspondant.

Par équipes de 2, 3, ou 4 (selon la quantité de travail estimée et les envies de chacun), nous nous sommes tous attribués 1 à 2 matières parmi :

1. Mathématiques (Nolan, Mohammed, Hana, et Louis),
2. Français (Valentine et Nolan),
3. Histoire (Victor, Arthur, et Valentine),
4. Géographie (Eric, Arthur et Mohammed),
5. Sciences (Eric).

Les diagrammes UML des mini-jeux ont été écrits par les personnes qui ont choisi la matière correspondante, contrairement au diagramme général qui est le fruit du travail collectif de toute l'équipe. A l'aide de *Draw.io* nous avons tous pu contribuer à son élaboration et apporter des modifications, commenter certaines sections, proposer des alternatives...

Chacun a pu mettre ses compétences au service du groupe et nous avons donc collectivement réalisé les menus, les rapports, les mini-jeux...

3 Possibles améliorations

Evidemment, nous avons simplement créé une base, et de nombreuses améliorations sont possibles. Nous n'avons pas pu les faire par manque de temps ou de compétences (par exemple, la plateforme est seulement accessible en local pour le moment), mais avons discuté ces possibilités lors de nos réunions et avons déjà plus ou moins réfléchi à la façon de les mettre en place.

Parmi ces idées d'amélioration, nous retrouvons :

— la partie parentale que nous voulions faire au départ, avec un mot de passe et des comptes-rendus réguliers des progrès de l'enfant,
— des points de cours si il y a trop d'erreurs aux exercices (Simplement de l'affichage, nous ne l'avons pas fait car cela prend du temps et que nous n'avons pas les compétences pédagogiques),
— Un système de score et de récompenses sur la plateforme de façon globale,
— rajouter des niveaux pour le jeu "Bitmap",
— Un correction pour chaque question,
— Un affichage en cas de victoire ou de défaite (la plupart des jeux ne l'ont pas par manque de temps, mais les compteurs de bonnes/mauvaises réponses sont déjà existants),
— l'utilisation des timers dans les jeux (les méthodes sont déjà définies mais ne sont pas encore utilisées),
— Un design de la plateforme plus attractif et moderne (Nous ne l'avons pas fait car nous voulions simplement une plateforme qui fonctionne, mais le squelette est déjà intégralement fait),
— Une optimisation du panel de réponses acceptées.

4 Méthodes agiles

Ce projet a également été l'occasion de découvrir le principe des méthodes agiles et d'apprendre à les mettre en place et à les appliquer correctement.

Nous avons mis en place la méthode SCRUM, qui favorise la collaboration, la livraison itérative et la flexibilité dans le développement de produits comme E-Learner.

4.1 Adaptabilité au changement

Comme vu précédemment, nous avions pour projet initial de créer une plateforme ludique à destination des enfants du CP à la terminale. Nous nous sommes rapidement rendus compte que ce n'était pas forcément judicieux, pour plusieurs raisons :

— La réalisation : Cela nécessite de créer une base de données conséquente, et des designs différents pour chaque niveau.
— Le niveau : Nous ne nous sentions pas vraiment légitime de créer du contenu pédagogique pour des lycéens notamment, alors que le niveau primaire est beaucoup plus atteignable

De plus, nous avons changé en cours de projet de manière de fonctionner. Au début, chacun faisait une partie du projet dans son coin, sans réelle organisation, puis nous nous sommes répartis le travail de façon à ce que chacun participe et apporte ses points forts au groupe.

4.2 Livraison itérative et progressive

Nous avons rendu un total de 4 itérations (dont l'itération 0), contenant à chaque fois notre avancée dans le code, un rapport du projet, et des rapports personnels.

Ces itérations étaient l'occasion à chaque fois de faire un point quant à l'avancée du projet, nos difficultés, nos idées... Cela nous motivait également à progresser régulièrement, et nous permettait d'ajuster notre travail pour rester dans le cadre fixé au départ.

Dans notre cas, ces itérations étaient des sprints, car les itérations étaient assez courtes (entre 1 et 3 semaines environ), aux cours desquels nous avons développé, testé et livré un incrément de fonctionnalités livrables.

Lors des premières itérations, nous avions rendu les itérables *Fonctionnalités* et *Manuel utilisateur*, correspondant au Backlog de E-Learner. En effet, ces documents contiennent les principales fonctionnalités de la plateforme, mais aussi différents scénarios, des images de l'interface graphique utilisée, ou encore comment utiliser correctement E-Learner.

Nous l'avons fait évoluer tout au long du projet afin de l'adapter au mieux à notre travail et aux idées proposées.

Afin de faire un bilan assez global sur notre projet, nous avons établi notre graphique d'avancement figure 1 en reprenant le principe du Burn Down.

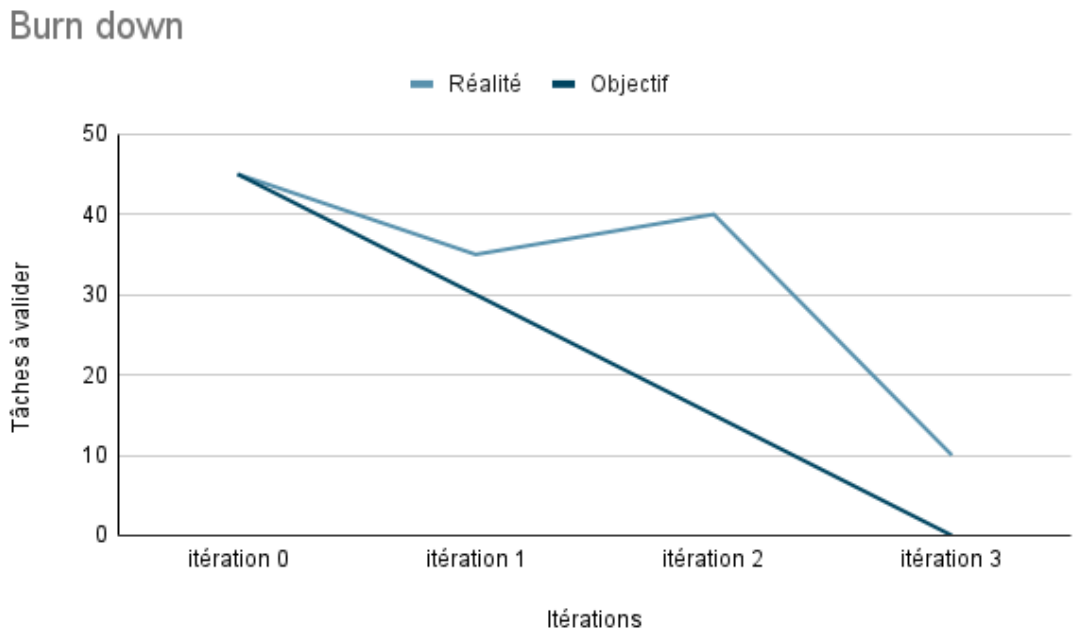


FIGURE 1 – Graphique d'avancement de l'équipe au cours du temps

on peut remarque que la quantité de tâches à accomplir augmente de nouveau après la première itération. Cela s'explique car cette itération correspond au moment où nous avons pris conscience que nous n'avions pas réalisé notre squelette en suivant le modèle MVC...

De plus, on voit que nous n'avons pas réussi à atteindre le seuil de 0 tâches à effectuer, comme nous le disions dans la partie 3.

4.3 Auto-organisation et responsabilisation de l'équipe

Vers le milieu du projet, nous sommes partis sur l'application Trello, présentée en cours de Méthodes Agiles. Nous avons pu mettre en place une organisation efficace, et savions ce que chacun avait à faire pour chaque échéance. Nous nous sommes donc répartis les tâches.

Pour des raisons de lisibilité, nous avons par la suite fait le choix de ne plus utiliser Trello, mais d'utiliser ClickUp, qui permet une vision plus globale et était plus agréable selon nous à prendre en main et à utiliser régulièrement.

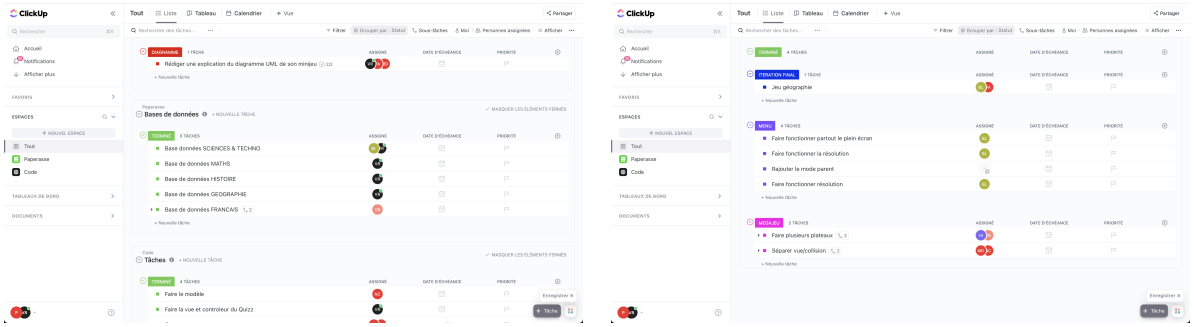
Si nous devions nommer un SCRUM-Master, ce serait sans aucun doute Nolan, car il a pris très à coeur la création d'un environnement de travail productif, collaboratif, et où tout le monde se sent à

l'aise.

4.4 Communication et collaboration face-à-face

Nous avons communiqué très régulièrement nos avancées sur le projet ainsi que nos questions, nos idées, ou nos difficultés. Chacun a trouvé sa place au sein de l'équipe.

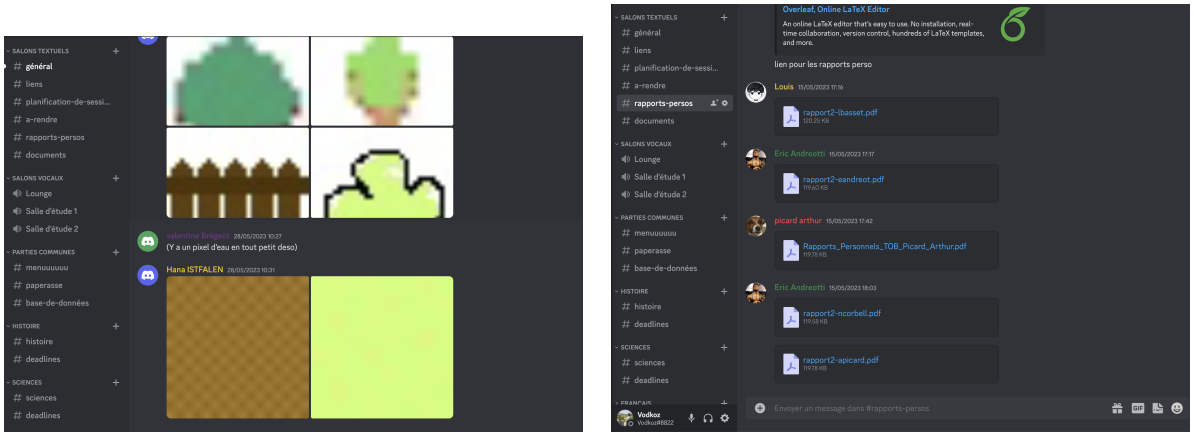
Des réunions, des "Daily SCRUM", étaient planifiées environ toutes les semaines ou toutes les 2 semaines afin d'échanger, soit en présentiel, soit à distance. Ces dernières nous permettait de faire un point sur ce qui avait déjà été fait, et sur ce qu'il nous restait à faire. Nous pouvions ainsi nous engager facilement sur des tâches et mettre à jour le ClickUp en conséquence, comme le montre la figure 2.



(a) Capture d'écran 1 (b) Capture d'écran 2

FIGURE 2 – Captures d'écran de notre ClickUp

Pour notre communication interne, nous avons utilisé un serveur Discord, sur la figure 3 afin de répartir les messages par thématiques et structurer notre communication. Nous avons également organisé des réunions "informelles" en appel Discord.



(a) Capture d'écran 1 (b) Capture d'écran 2

FIGURE 3 – Captures d'écran de notre serveur Discord

Nous avions également un GitHub pour partager nos codes et gérer les différentes versions.

En particulier, nous organisions des réunions de planification du sprint à chaque nouvelle itération, afin d'examiner le backlog de la plateforme et définir les objectifs à atteindre. Nous faisions également des revues et des rétrospectives de sprint, où nous effectuions le bilan du sprint et émettions des retours sur la façon dont cela s'est passé, proposions des améliorations (comme le changement vers ClickUp par exemple)... Nous pouvions aussi exprimer notre ressenti sur le projet, l'équipe, l'organisation, avec par exemple les "Weather Reports". Cela nous a permis de mieux nous connaître et d'être à l'écoute

des autres pour avancer plus facilement et sereinement.

Au cours de ces réunions, nous faisons également un point sur nos "*User Stories*", afin de rester cohérent entre notre travail et la réelle utilisation de la plateforme.

Un exemple d'User Story est : "En tant qu'enfant, je veux mettre la plateforme en plein écran afin d'être plus concentré sur mon travail." Cette phrase nous a permis de développer ce mode plein écran en aillant connaissance de l'intérêt de cette fonctionnalité.

5 Elements de conception

Nous avons fait le choix de limiter le nombre de questions posées pour chaque session de révisions. En effet, pour la plupart des jeux, ils prennent fin après 10 questions.

Nous avons choisi cette fonctionnalité dans un but pédagogique, en estimant que c'était suffisant pour l'élève pour se rendre compte de son niveau de maîtrise de la notion. Il peut ainsi passer à un autre exercice ou recommencer, au lieu de continuer à l'infini.

2 compteurs sont affichés sur l'écran, celui en haut à gauche correspond au nombre de succès, et celui en haut à droite au nombre total de questions.

Cependant, certains jeux, comme en géographie, doivent atteindre les 10 erreurs pour quitter, afin de laisser le temps à l'élève d'essayer, sans pour autant rester bloqué sur la question.

De même, le jeu "de la Guillotine" prend fin au bout de 4 erreurs, nous avons estimé que c'était cohérent par rapport au contexte du jeu.

Le jeu du "Bitmap" se distingue des autres car il ne dépend pas directement du nombre d'erreurs, mais plutôt de la "barre de vie". En effet, il faut donner 5 réponses correctes pour gagner, et à l'inverse, 5 réponses fausses pour perdre. Après 5 mauvaises réponses, l'élève est invité à réviser sa leçon et à rejouer. Nous avons fait ce choix pour que le jeu soit plus court et plus ludique, et pour ne pas décourager trop l'élève si ce dernier enchaîne les mauvais réponses.

Nous avons choisi de ne pas laisser la possibilité à l'élève de quitter le jeu en pleine partie, c'est pourquoi il faut atteindre les 10 questions pour changer de jeu. Ainsi, l'élève reste concentré sur le jeu et est obligé d'atteindre le seuil de questions.

5.1 Menu de la plateforme

Le menu que nous proposons repose sur le diagramme UML de la figure 4.

Nous avons modifié ce diagramme pour l'adapter à l'architecture MVC, cette version n'est pas définitive.

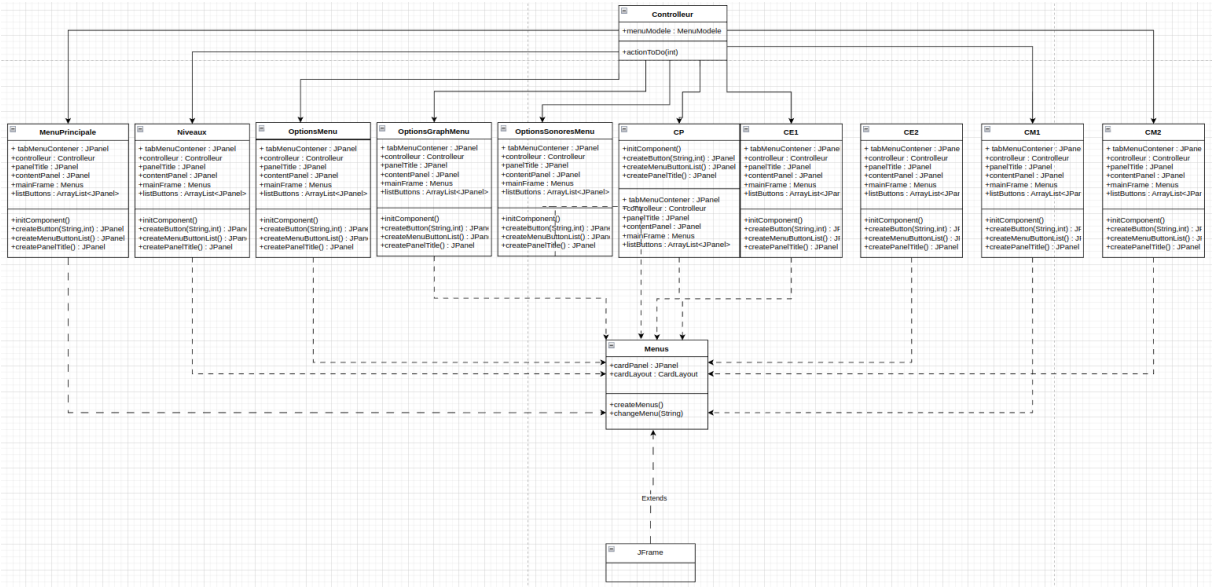


FIGURE 4 – Vue globale du Menu

5.2 Jeu du corps humain

Voici le diagramme UML 5 pour le jeu du corps humain.

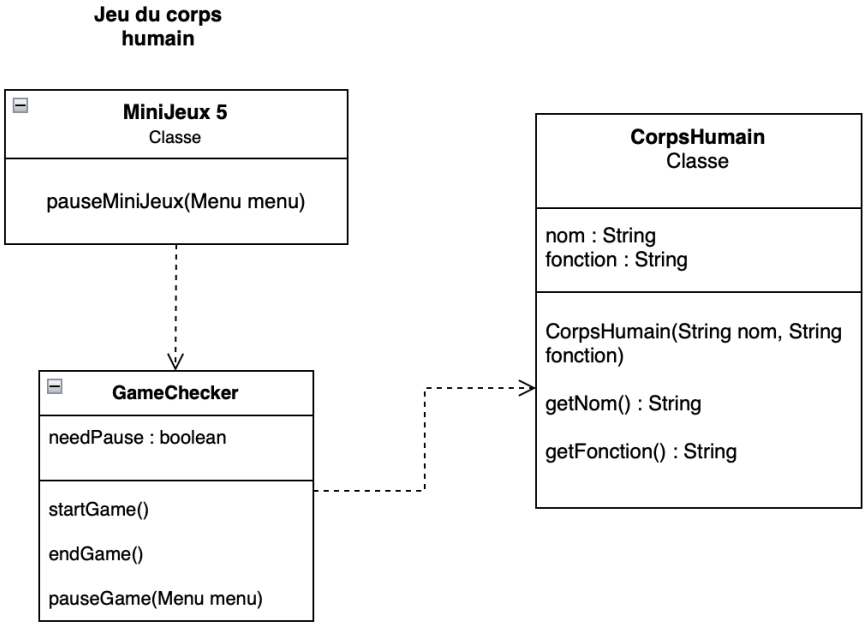


FIGURE 5 – Diagramme UML du jeu du corps humain

5.3 Quizz

Voici le diagramme UML 6 pour le quizz. Ce jeu est réutilisé dans toutes les matières ou presque, car il est simple à mettre en place, il faut simplement créer une base de données conséquente derrière.

Ce mini-jeu est composé de plusieurs classes, dont un controlleur et une interface graphique (la Vue). Les questions sont générées aléatoirement grâce au générateur.

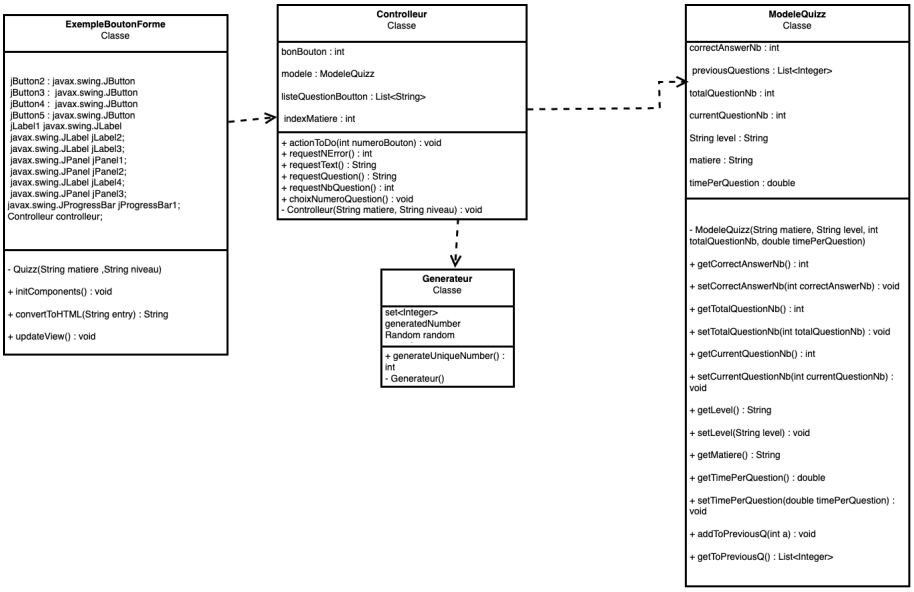


FIGURE 6 – Diagramme UML du quizz

5.4 Jeu du Boss

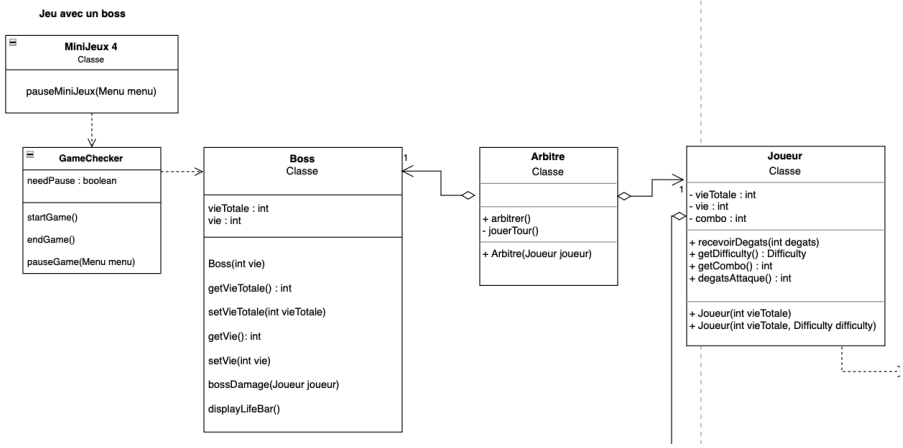


FIGURE 7 – Diagramme UML du jeu du boss

5.5 Jeu de la guillotine

La vue

Dans le groupe Vue, il y'a deux classes Vue et Scene.
— La classe Vue permet de gérer la fenetre d'affichage du jeu.
— La classe Scene, elle, permet de de gérer les composants de Swing qui vont s'afficher sur la fenêtre.

Le constructeur permet d'initialiser les affichages mais aussi de les actualiser : **Scene(int etat, String question, String truth, int nbQuestion)**. Pour les arguments, "etat" désigne le nombre d'erreurs commises par l'utilisateur et donc la hauteur de la guillotine, "question" est la question posée, "truth" est la véracité de la réponse de l'utilisateur, et "nbQuestion" est le nombre de questions passées.

La méthode **changePaintComponent(int etat)** permet de choisir l'image png correspondant au niveau de la guillotine.

La méthode **displaytruth(String truth)** permet d'afficher la véracité de la réponse.
Finalement, la méthode **handleQuestion(String question** permet d'afficher la question sur plusieurs lignes en fonction de sa longueur.

Le modèle

— La classe Modele : La classe Modele est, par définition, le modèle du mini-jeu avec les données correspondant au modèle. Cela signifie que c'est elle qui va stocker le nombre de questions, d'erreurs, le numéro de la question actuelle mais également le nombre de fautes maximales.
— La classe Generator : Cette classe permet de générer un entier correspondant au numéro de la question en json pour ensuite la récupérer.
— La classe Joueur : Joueur ici va gérer la vérification des réponses grâce à l'interface AnswerGameChecker, mais aussi la gestion du temps de réponse.

Le controlleur

La partie controlleur est composée de la classe Controlleur, qui permet de faire la liaison entre le modèle et la vue.

La méthode **actionToDo** permet, elle, de mettre à jour le modèle.

Finalement, les méthodes avec request permettent à la vue d'actualiser son interface en fonction des donées du modèle.

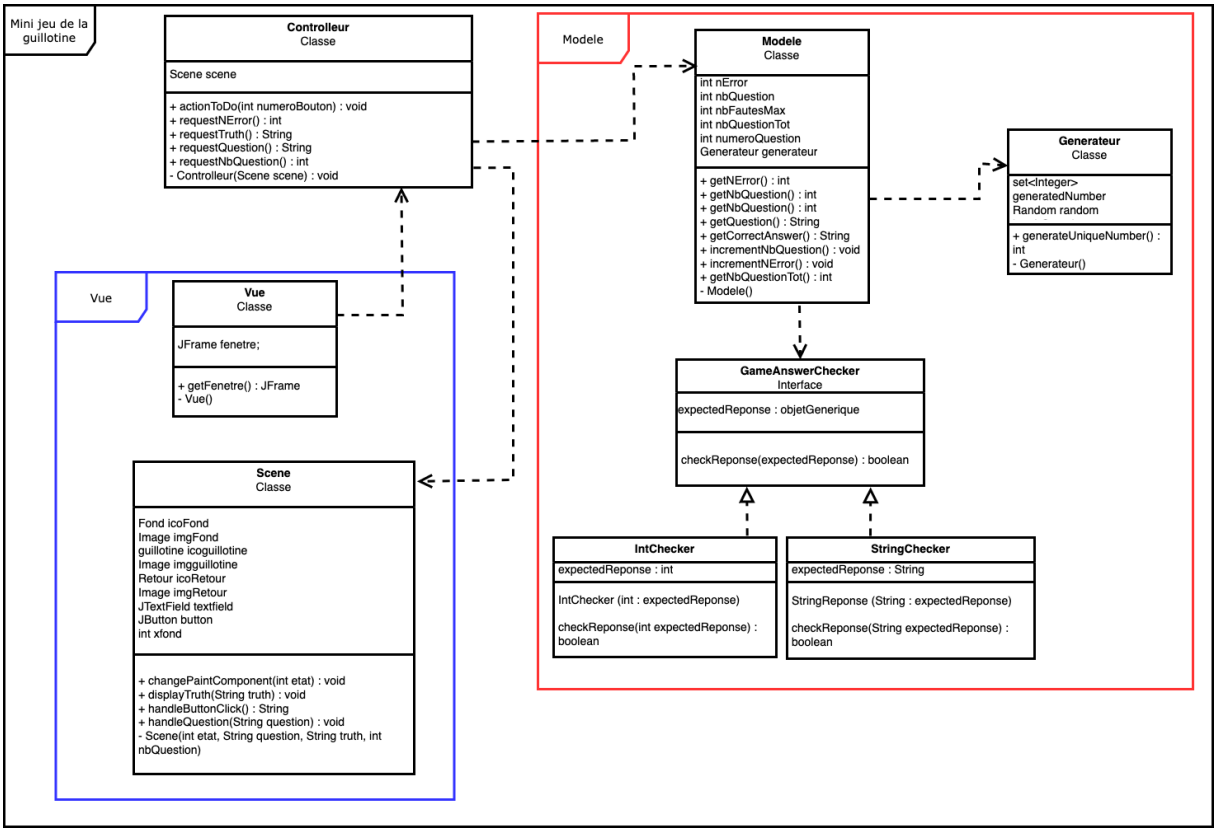


FIGURE 8 – Diagramme UML du jeu de la guillotine

5.6 Jeu de géographie

Le diagramme UML 9 représente un jeu dans lequel il faut placer correctement les villes ou les pays sur une carte. Nous voulons réutiliser sur ce jeu, comme pour le jeu du corps humain, le *MouseClicker* vu en TP.

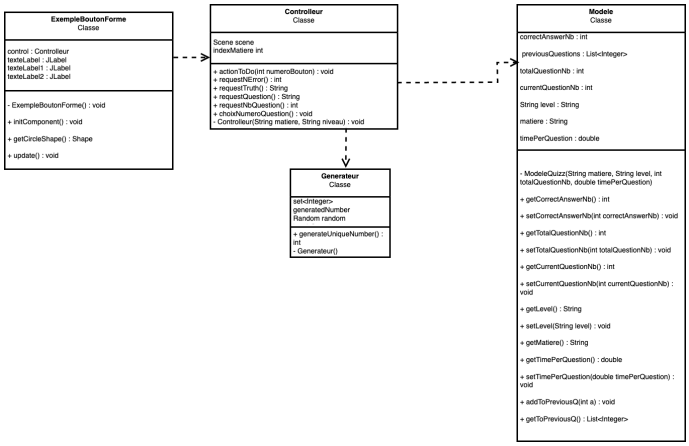


FIGURE 9 – Diagramme UML du jeu de géographie

5.7 Jeu Bit Map

Bit Map est le jeu le plus avancé pour l'instant.

Son diagramme UML est très volumineux et pas forcément lisible pour le moment, nous avons donc fait le choix de ne pas le mettre dans ce rapport afin de prendre le temps par la suite de l'améliorer.

Bit Map est un jeu d'aventure où le joueur incarne un jeune homme à la recherche d'un trésor caché dans une carte. Tout au long de son voyage, le joueur doit affronter des monstres qui se dressent sur son chemin et lui posent des défis mathématiques pour réussir à les vaincre.

Les joueurs peuvent explorer la carte, découvrir de nouveaux endroits, trouver des indices et résoudre des énigmes pour avancer dans l'histoire.

L'intérêt pédagogique de ce jeu est contenu dans les combats avec les monstres. En effet, ils consistent en des mini-jeux de mathématiques, qui couvrent tout le programme de primaire selon le niveau du joueur (additions, soustractions, géométrie, fractions, algèbre plus complexe...).

En progressant dans le jeu, le joueur peut collecter des indices et des objets qui l'aideront à trouver le trésor caché.

Ainsi, Bit Map est un jeu combinant l'exploration, la résolution d'énigmes et les mini-jeux de mathématiques pour offrir une expérience de jeu unique et stimulante.

Nous avons pour le moment inclus 15 mini-jeux de mathématiques pour couvrir tout le programme de mathématiques du primaire. Il y a 3 mini-jeux de mathématiques par niveau, ce qui permet aux joueurs de réviser différents concepts mathématiques.

Pour bien structurer notre jeu, nous avons utilisé plusieurs packages pour organiser le code de manière efficace.

- Le package "Games" : contient l'implémentation des différents mini-jeux qui composent le jeu. Cette approche permet de séparer la logique de chaque mini-jeu, rendant ainsi le jeu plus modulaire et facilement maintenable.
- Le package "Entity" : destiné à la création des différents types de joueurs selon leur niveau scolaire, il peut également inclure des ennemis, des objets et d'autres éléments du jeu qui interagissent avec les joueurs.
- Le package "Object" : contient les différents objets utilisés dans le jeu, tels que les armes, les armures, les potions, etc. Il peut également contenir des monstres.
- Le package "Tile" : utilisé pour créer la carte du jeu et la configurer selon les besoins, il permet de définir les éléments visuels de la carte tels que les murs, les portes, les chemins, etc.
- Le package "Maths" : gère le traitement des collisions entre les différents éléments du jeu, regroupe les différents éléments du jeu et la classe Math pour lancer le jeu selon le niveau scolaire.

Ce package est essentiel pour le bon fonctionnement du jeu, car il gère les interactions entre les différentes entités du jeu, la logique du jeu et le lancement des différents mini-jeux.

En résumé, en utilisant des packages pour organiser le code de notre jeu, nous avons créé une architecture modulaire et facilement maintenable, ce qui permet de faciliter la création de nouveaux mini-jeux, personnages et objets dans le jeu.

5.8 Jeu coloriage

Le diagramme UML 10, représente un jeu dans lequel l'utilisateur doit colorier de la bonne couleur les cases d'un dessin selon le code couleur donné, chaque case possède alors des opérations mathématiques afin de coder la couleur à y mettre.

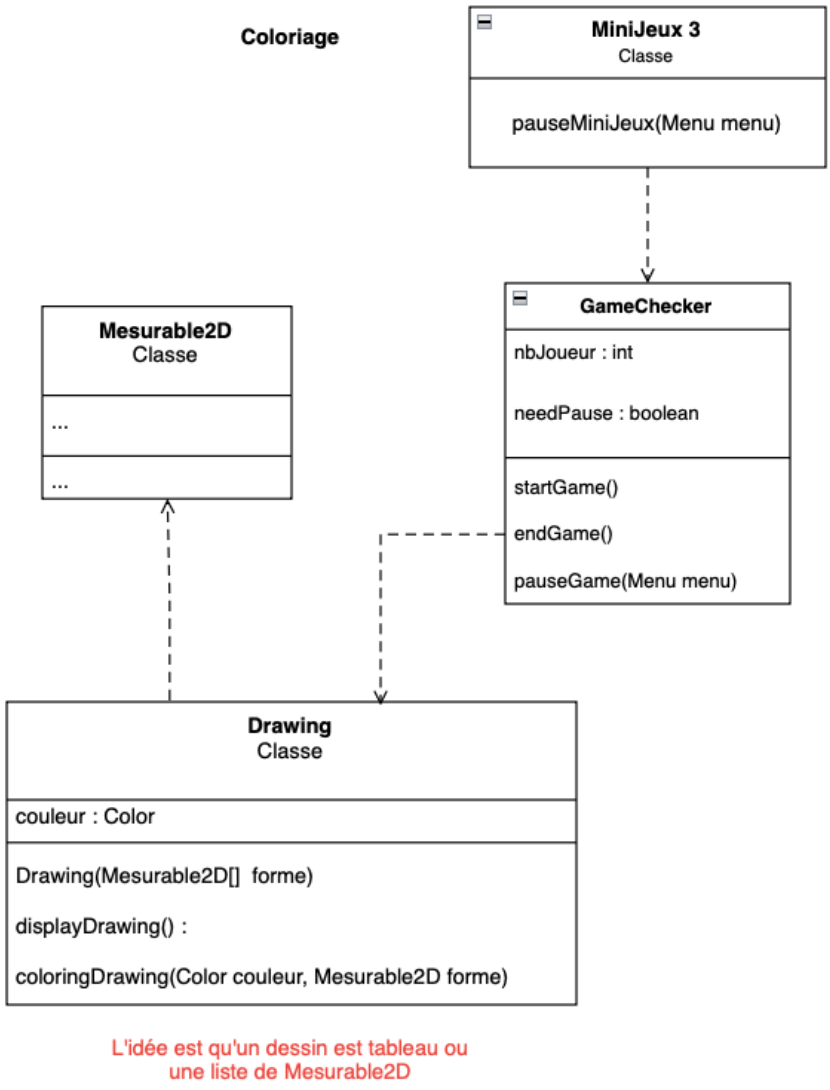


FIGURE 10 – Diagramme UML du jeu du coloriage