Auxilien Katia
El Haddad Nassim

Rapport SAé de développement partie 2 : Donjon Infini

Dans notre version du jeu du Donjon Infini, vous incarnez un héros qui aura pour objectif de progresser dans un plateau de dimensions personnalisées, composé d'une multitude d'éléments que nous avons ajoutés, parmi lesquels de nouveaux monstres, de nouvelles armes, des nouvelles fonctionnalités permettant d'améliorer l'expérience utilisateur, comme un menu, avec des choix de thèmes, de difficultés et de dimensions, une musique et un meilleur score enregistré dans les fichiers du jeu. U*ne javadoc a également été réalisée.*

Bestiaire : **(des nouveaux éléments)**

- Le Chef des abysses, un boss, puissant, et ayant une assez faible probabilité d'apparaître durant la partie, mais dont le combat vaut le détour : il permet d'obtenir énormément d'or.
- Le Gloculaire, un monstre de puissance moyenne, mais qui a une particularité très puissante : il réduit grandement votre champ de vision. En effet, durant un nombre de tours égal à la dimension du plateau, la portée de votre vision est limitée aux cases adjacentes au héros.
- Le Gobelin, un monstre vil et sournois, qui malgré sa puissance limitée peut être bien contraignant à combattre : avant de mourir, il emportera avec lui une partie de votre or.
- Le Squelette, l'un des monstres les plus fragiles et basiques.

Ces nouveaux monstres héritent de la classe Monstre().

Armes : **(des nouveaux éléments)**

- L'épée légendaire, une arme redoutable, très puissante, mais rare. Certainement l'un des seuls moyens de pouvoir contrer le chef des abysses.
- La hache, une arme puissante et solide, vous permettant de tenir tête à la plupart des monstres que vous rencontrerez sur le plateau.
- Les dagues, des armes légères, fragiles, mais communes : De très bonnes armes à obtenir en début de partie, mais qui deviennent vite insuffisantes face à des monstres puissants.

Les classes héritent de la classe Arme.
Les classes ont pour attribut valMax. L'intérêt de valMax réside dans la gestion de la durabilité de l'arme, elle sert à réparer l'arme sans dépasser sa valeur initiale.

Soins :

- Le feu de camp, un objet de soin supérieur à la potion, puisqu'il rétablit les PV's du joueur et répare entièrement l'arme que le héros possède actuellement.
- La potion, un élément de soin commun qui vous permettra de vous régénérer avant de petits combats. La potion à l'attribut valeur venant de Case, et détermine la quantité de pv qu'elle va régénérer

Ces classes héritent de CaseBonus, héritant lui-même de Case.

Autres :

- La classe "Case" est une superclasse qui représente une case dans un jeu. Chaque case a une valeur et peut être visible ou invisible. La méthode "getLabel" retourne une chaîne de caractères vide, car la classe "Case" est une superclasse. De même, les méthodes "getLabelPv" et "getIntPv" retournent respectivement la chaîne de caractères "0" et le nombre 0, parce qu'elles sont également des méthodes de la superclasse. La méthode "setVisibilite" permet de définir la visibilité d'une case, tandis que la méthode "getVisibilite" renvoie la visibilité actuelle de la case. La méthode statique "newRandomCase" génère une nouvelle case aléatoire en fonction d'un réel "diff" qui représente la difficulté du jeu. Selon des probabilités prédéfinies, elle peut créer un monstre, de l'or, une arme ou une potion. La méthode "getDescription" retourne une chaîne de caractères décrivant la case comme étant une case simple.

  Côté technique, la classe "Case" est la superclasse de différentes classes représentant des types de cases spécifiques, tels que les monstres (ChefDesAbysses, Gloculaire, Gobelin, Squelette), l'or (Or), les armes (EpeeLegendaire, Hache, Dague) et les potions (Potion). La méthode "newRandomCase" utilise un générateur de nombres aléatoires pour créer une instance d'une classe enfant de "Case" en fonction de probabilités définies. Cela permet de générer des cases variées et intéressantes dans le jeu.

- La Case bonus représente un type particulier de case : ce sont les cases qui contiennent un élément spécial : une arme, de l'or, une potion, un feu de camp. Le taux d'apparition de ces éléments est déterminé par des probabilités dans la méthode newRandomCase().

  La classe "CaseBonus" est une classe enfant de la classe "Case" qui représente une case spéciale dans un jeu. Elle hérite des attributs et des méthodes de la classe parente. Le constructeur de la classe permet de créer une instance de "CaseBonus" avec une valeur spécifiée en argument, ou sans aucune valeur.

  Côté technique, la classe "CaseBonus" est une classe enfant de la classe "Case" et elle est également la superclasse de différentes classes représentant des types spécifiques de cases bonus, tels que l'or (Or), les armes (Arme) et les potions (Potion) et le Feu de camp.
- Le Contrôleur

  La classe Controleur est la classe principale dans un jeu appelé "Donjon infini". Elle contrôle le déplacement du héros sur le plateau de jeu, gère les interactions avec les différentes cases.
- Les Descriptions

  Pour un utilisateur :

  La classe "Description" est une classe qui permet d'écouter les événements de la souris. Lorsque vous survolez un objet, sa description s'affiche, et quand vous quittez l'objet, la description disparaît.

  Pour les développeurs :

  La classe "Description" est une implémentation de l'interface "MouseListener" de la bibliothèque Swing. Elle réécrit les méthodes "mouseEntered" et "mouseExited" pour gérer les événements de survol de souris. Lorsque l'utilisateur survole un

élément graphique (de type "VueCase"), la méthode "mouseEntered" est appelée, récupère la source de l'événement (l'objet survolé), et appelle la méthode "showDescription()" de l'objet "VueCase" pour afficher la description. De même, lorsque l'utilisateur quitte l'objet, la méthode "mouseExited" est appelée, cachant ainsi la description en appelant la méthode "hideDescription()" de l'objet "VueCase".

- Le Héros représente le personnage que vous incarnez lors de la partie. Il a pour attribut ses PVs, la valeur de la difficulté, l'arme qu'il possède, ses PVs initiaux (limite maximum) pour éviter que le héros obtienne plus de PVs que possible durant la partie, un attribut booléen move et un attribut booléen monstreMort, le premier nous permettant de nous déplacer, ou non selon la situation, et l'autre indiquant si le monstre qu'il combat est mort. La quantité de PVs qu'il aura au cours de la partie est déterminé par la difficulté choisie.
- Le Menu permet de choisir la difficulté de la partie ainsi qu'un thème graphique :
    - Le choix de la difficulté est ainsi stockée dans une variable diff, et cette variable est réutilisée dans les classes d'arme, influant donc sur la puissance des armes, dans les classes de monstre, influant sur leur puissance, et enfin dans la classe du Héros, déterminant son nombre de PVs. Cette variable peut prendre 3 valeurs : 1.0, 2.0 et 3.0, pour débutant, intermédiaire et expert.
    - Le choix du thème est également stocké dans des variables int, allant de 1 à 4. Des thèmes ont ainsi été créés : un thème vanilla, qui existe depuis la première version du jeu, un thème clair, un thème sombre et enfin un thème uni.
- La Musique

    Pour un utilisateur :

    La classe "Musique" est une classe qui permet de jouer de la musique en arrière-plan lors de la parti.

    Pour les développeurs :

    La classe "Musique" utilise les fonctionnalités de la bibliothèque "javax.sound.sampled" pour jouer de la musique en arrière-plan. Elle comporte les attributs "clip", "musicFile" et "audioInput" pour gérer les opérations de lecture audio. La méthode "playMusic()" charge le fichier audio spécifié dans l'attribut "musicFile", crée un flux d'entrée audio avec "AudioSystem.getAudioInputStream()", puis ouvre un lecteur audio avec "AudioSystem.getClip()". Ensuite, la méthode "loop()" permet de lire la musique en boucle de manière continue, et "start()" lance la lecture. La méthode "stopMusic()" arrête la lecture du clip et ferme le flux d'entrée audio. Cette classe offre une façon simple d'intégrer la lecture de musique dans un jeu en utilisant la bibliothèque "javax.sound.sampled".
- La Vue du plateau

    Explication pour un utilisateur lambda :

    La classe VuePlateau est une partie de l'interface graphique de votre jeu. Elle affiche visuellement le plateau de jeu et permet de mettre à jour l'apparence des cases en fonction des événements du jeu.

    Explication pour les développeurs :

    La classe VuePlateau hérite de JPanel et représente visuellement le plateau de jeu dans l'interface graphique. Elle est composée d'un tableau bidimensionnel de VueCase qui représente chaque case du plateau. Lors de la création d'une instance de VuePlateau, la taille du plateau et le thème visuel sont spécifiés. La méthode setLayout est utilisée pour définir la disposition des cases dans le panneau en

utilisant un GridLayout. La méthode updateVue permet de mettre à jour l'apparence visuelle d'une case spécifique du plateau. Elle prend en paramètre la nouvelle case et ses coordonnées dans le tableau tabVueCases. La méthode setCase de chaque VueCase est appelée pour mettre à jour l'apparence de la case. La méthode paintComponent est utilisée pour redessiner le plateau de jeu. Elle itère à travers toutes les VueCase et appelle la méthode repaint pour chaque case, ce qui déclenche le redessin de la case. La méthode upPoint est responsable de l'augmentation du score du joueur en fonction du type d'objet trouvé dans une case. Elle utilise des conditions pour vérifier le type de case et augmente le score en conséquence. Par exemple, si la case est un monstre, le score augmente en fonction des points de vie du monstre. La méthode getVuePoints est un getter pour la variable vuePoints, qui représente le score actuel de la partie. La méthode launchAnimation est utilisée pour lancer une animation sur une case spécifique du plateau. Elle appelle la méthode launchAnimation de la VueCase correspondante. En résumé, la classe VuePlateau gère l'affichage du plateau de jeu, la mise à jour des cases, la gestion du score.

Problèmes et difficultés rencontrées :

- Diverses exceptions, notamment nullPointerException car DonjonInfini.jeu is null, donc des getter et des setter ont été créé.
- Nous avons également placé des exceptions pour la musique dans Musique(), ainsi que pour wait() dans vueCase.
- Pour l'effet du gloculaire, nous avons rencontré des difficultés à faire disparaitre et apparaître les éléments créé par VueCase().
- De grosses difficultés pour les animations, car nous n'avons pas trouvé comment mettre en "pause" ou en "attente" le programme en attendant que l'animation se termine.

Crédits :

Sprites :

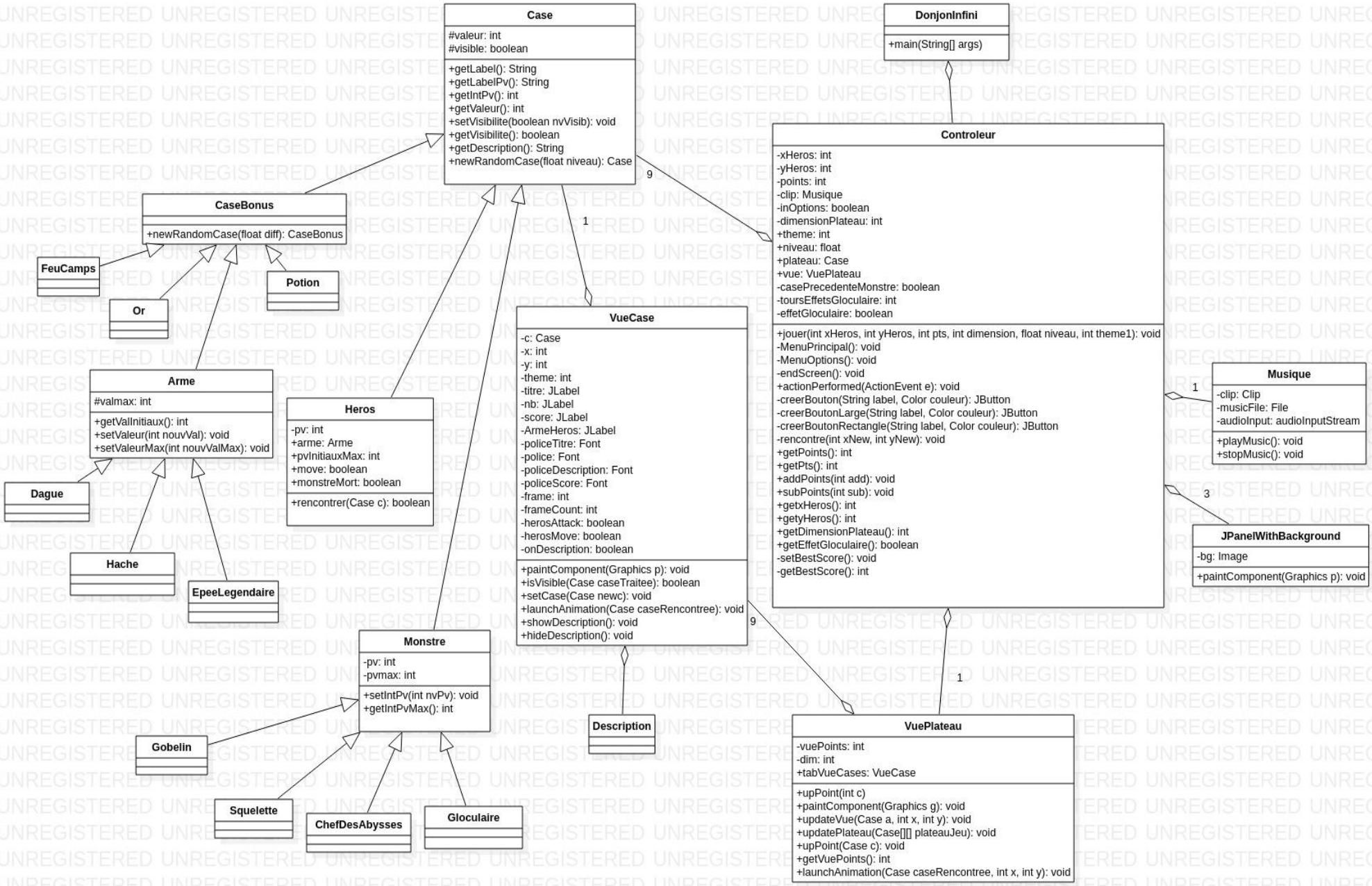
- Dague provenant de l'asset "fantasy weapons" par YohkGames (lien : https://yohkgames.itch.io/fantasy-weapons-pt2)
- Fond des descriptions, EpeeLegendaire, Hache, Arme, Pieces et Sac d'or provenant de l'asset "Shikashis fantasy icons pack" par cheekyinkling (lien : https://cheekyinkling.itch.io/shikashis-fantasy-icons-pack)
- Feu de camp de l'asset "Animated pixel campfire" par SMStudios (lien : https://simon-develop.itch.io/2d-campfire-free)
- Hero provenant de l'asset Fantasy Knight de aamatniekss (lien : https://aamatniekss.itch.io/fantasy-knight-free-pixelart-animated-character)
- Chef des abysses provenant de l'asset EVil Wizard 2 par LuizMelo (lien : https://luizmelo.itch.io/evil-wizard-2)
- Gloculaire, Gobelin et Squelette provenant de l'asset "Monsters creatures fantasy" par LuizMelo (lien : https://luizmelo.itch.io/monsters-creatures-fantasy)
- Potion provenant de l'asset "Animated Potion Assets Pack" par Flip (lien : https://flippurgatory.itch.io/animated-potion-assets-pack-free?download)
- Fond du menu par The Outlander https://the-outlander.itch.io/free-dungeon-backgrounds

Musique :

- "Magic Escape Room" : Instruments - Harp, Bassoon, Flutes, Celesta, Percussion, Oboe, Timpani, Cellos, Basses, Xylophone, Glockenspiel, Violins, Violas, French Horns, Trombones, Tuba, Pipe Organ, Clock Tower, Chimes, Church Bell. Artiste - Kevin MacLeod. Téléchargée sur incompetech.com (lien : https://incompetech.com/music/royalty-free/music.html?source=codeur-com-blog&utm_source=codeur-com-blog)

Les Sprites ont été créés par différents artistes et sont utilisés conformément à leurs licences. La musique a été composée par Kevin MacLeod et est disponible en tant que musique libre de droits sur incompetech.com.

Diagramme de classe (ci-dessous)


Case
#valeur: int
#visible: boolean
+getLabel(): String
+getLabelPv(): String
+getIntPv(): int
+getValeur(): int
+setVisibilite(boolean nvVisib): void
+getVisibilite(): boolean
+getDescription(): String
+newRandomCase(float niveau): Case
DonjonInfini
+main(String[] args)
Controleur
-xHeros: int
-yHeros: int
-points: int
-clip: Musique
-inOptions: boolean
-dimensionPlateau: int
+theme: int
+niveau: float
+plateau: Case
+vue: VuePlateau
-casePrecedenteMonstre: boolean
-toursEffetsGloculaire: int
-effetGloculaire: boolean
+jouer(int xHeros, int yHeros, int pts, int dimension, float niveau, int theme1): void
-MenuPrincipal(): void
-MenuOptions(): void
-endScreen(): void
+actionPerformed(ActionEvent e): void
-creerBouton(String label, Color couleur): JButton
-creerBoutonLarge(String label, Color couleur): JButton
-creerBoutonRectangle(String label, Color couleur): JButton
-rencontre(int xNew, int yNew): void
+getPoints(): int
+getPts(): int
+addPoints(int add): void
+subPoints(int sub): void
+getxHeros(): int
+getyHeros(): int
+getDimensionPlateau(): int
+getEffetGloculaire(): boolean
-setBestScore(): void
-getBestScore(): int
9
1
CaseBonus
+newRandomCase(float diff): CaseBonus
FeuCamps
Or
Potion
Arme
#valmax: int
+getValInitiaux(): int
+setValeur(int nouvVal): void
+setValeurMax(int nouvValMax): void
Dague
Hache
EpeeLegendaire
Heros
-pv: int
+arme: Arme
+pvInitiauxMax: int
+move: boolean
+monstreMort: boolean
+rencontrer(Case c): boolean
VueCase
-c: Case
-x: int
-y: int
-theme: int
-titre: JLabel
-nb: JLabel
-score: JLabel
-ArmeHeros: JLabel
-policeTitre: Font
-police: Font
-policeDescription: Font
-policeScore: Font
-frame: int
-frameCount: int
-herosAttack: boolean
-herosMove: boolean
-onDescription: boolean
+paintComponent(Graphics p): void
+isVisible(Case caseTraitee): boolean
+setCase(Case newc): void
+launchAnimation(Case caseRencontree): void
+showDescription(): void
+hideDescription(): void
Musique
-clip: Clip
-musicFile: File
-audioInput: audioInputStream
+playMusic(): void
+stopMusic(): void
1
3
JPanelWithBackground
-bg: Image
+paintComponent(Graphics p): void
Monstre
-pv: int
-pvmax: int
+setIntPv(int nvPv): void
+getIntPvMax(): int
Gobelin
Squelette
ChefDesAbysses
Gloculaire
Description
VuePlateau
-vuePoints: int
-dim: int
+tabVueCases: VueCase
+upPoint(int c)
+paintComponent(Graphics g): void
+updateVue(Case a, int x, int y): void
+updatePlateau(Case[][] plateauJeu): void
+upPoint(Case c): void
+getVuePoints(): int
+launchAnimation(Case caseRencontree, int x, int y): void
9
1