Projet POO – The Island

Du 22/04/2022 au 23/05/2022

THE ISLAND

Enseignant encadrant : Mme.Abouda

Maud – IATIC3

Rayane – IATIC3

Mehdi AISSI – IATIC3

Raphaëlle – IATIC3

Gabin– IATIC3

Groupe 3

UVSQ université PARIS-SACLAY

ISTY Institut des Sciences et Techniques des Yvelines

CAMPUS DE MANTES EN YVELINES

CAMPUS DE SAINT-QUENTIN-EN-YVELINES

Sommaire:

Introduction :

Dans le cadre de notre cursus à l'ISTY, nous avons reçu le projet d'adapté un jeu de société en une version numérique à l'aide du langage de programmation java et des techniques de programmation orientées apprises dans le cours dédié.

Nous sommes un groupe de 5 personne à réaliser ce projet, avec des niveaux différents en programmation java.

Ce projet nous a demandé beaucoup d'investissement, en temps et en énergie. Il a été réalisé sur une période de 7 semaines de travails. Semaines durant lesquels nous avons dut faire face à de nombreux défis et imprévus

Dans ce rapport sera décrit notre gestion du projet, son implémentation en fonction des 3 modules : vue, modèle et contrôleur, et enfin un manuel d'utilisation.

Gestion de projet :

Système à réaliser :

Nous avions à réaliser le jeu de société « The Island » en java. Dans ce jeu, les joueurs dirigent des explorateurs qui doivent s'échapper d'une île avec leur trésor dans le but de se sauver. Le joueur qui réussit à s'échapper avec le plus de trésor et le plus d'explorateur gagne la partie.

Une partie de jeu commence par chacun des joueurs plaçant chacun leur tour leur pions d'explorateur et leurs bateaux.

Ensuite chaque tour de jeu les joueurs peuvent poser une tuile rouge pour profiter de leurs effets. Ils peuvent ensuite déplacer leurs explorateurs dans une limite de coups disponibles. Ils doivent ensuite retirer une tuile du plateau, celle-ci est soit ajoutée dans la main du joueur soit activée automatiquement. Enfin avant de passer au joueur suivant, le joueur peut tirer au dé le déplacement d'une des créatures présente sur le plateau. Ces créatures ont elles aussi des effets qui vont nous permettre d'obtenir un avantage ou de pouvoir gêner les adversaires…

Le jeu se termine lorsque la tuile avec l'effet volcan est retirée. A ce moment-là, la partie se termine et l'on fait les comptes des trésors sauvés par les joueurs. La personne avec le plus de trésor gagne la partie.

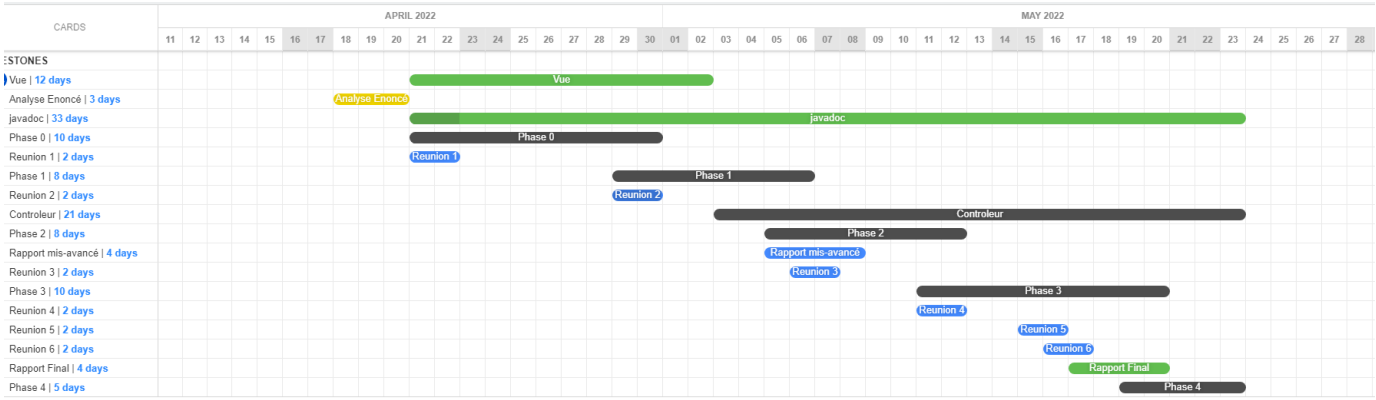
Notre objectif est donc de réaliser ce jeu de société, avec une interface graphique programmée en Java.

Notre approche pour réaliser ce système :

Dans un premier temps, nous nous sommes réunis, pour prendre connaissance du sujet tous ensemble. Cette première réunion nous a permis de partager nos points de vue sur la conception du jeu. A la suite de cela, nous avons conçu un diagramme de classe afin de mettre au clair nos idées. Nous avons décidé d'utiliser le modèle MVC (modèle, vue,

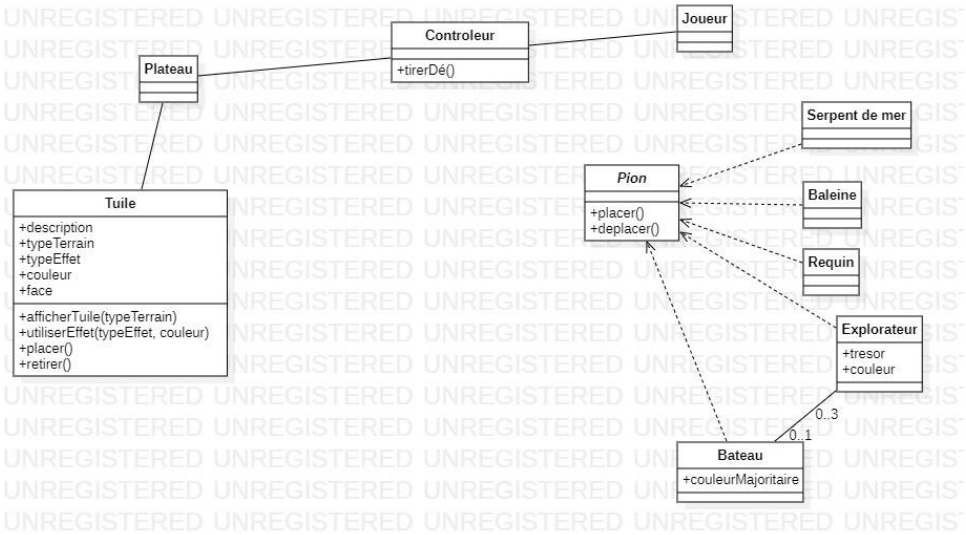
contrôleur) pour l'implémentation du jeu. De ce fait, nous avons pris la décision de faire deux groupes, un travaillant sur la vue et l'autre sur le modèle. Les membres des groupes sont Medhi AISSI et Maud LESTIENNE, pour la vue, Gabin FORRAT, Raffaele GIANNICO et Rayane HAMMOUMI pour le modèle. Pour le contrôleur, il a été décidé que les deux groupes se rejoindraient pour l'implémenter.

Nous avons ensuite réparti les tâches à faire, pour cela nous avons fait un planning et un diagramme de Gantt.



Nous avons décidé de nous réunir une fois par semaine, pour faire un point sur l'avancement et ce qui doit être fait la semaine suivante.

Diagramme de classe :



Premier diagramme de classe effectué

Voici le premier diagramme de classe que nous avons effectué. Il est assez basique, mais c'est ce que nous obtenions après avoir analysé le cahier des charges. En effet, le plateau contenait des tuiles avec quelques attributs et méthodes. Et, les différents monstres héritaient de pions. Mais nous nous sommes très vite rendu compte que ce n'était pas suffisant c'est pourquoi il était nécessaire d'ajouter certaines classes, et d'en compléter certaines. Nous obtenons le diagramme de classe ci-dessous :

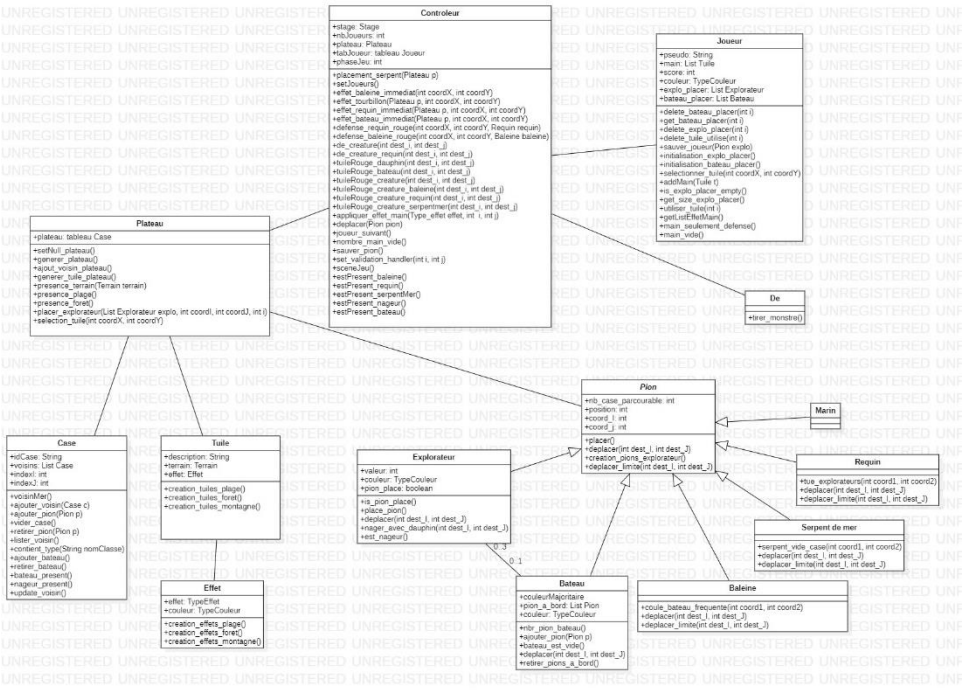


Diagramme de classe final

Ce diagramme est beaucoup plus conséquent que le premier et nous pouvons y remarquer quelques changements. Notamment que la classe plateau contient aussi une nouvelle classe appelé Case. Aussi, la classe tuile est associé à une classe Effet abstraction d'un effet d'une tuile. Et enfin, de nombreuse méthodes et attribut supplémentaire ont été ajouter à chaque case.

Planification et répartition des tâches :

Pour ce projet nous sommes 5 étudiants. Nous avons donc décidé de mettre plus de personnes sur la partie modèle car elle nous semblait plus conséquente. Mehdi et Maud étaient assignés à la vue. Et pour la partie modèle, Gabin, Rayane et Raffaele. Enfin le contrôleur a été fait par les deux parties.

Nous faisions en sorte de faire au moins une réunion par semaine, pour que chacun détaille les changements qu'il a effectués depuis la précédente réunion. De plus, nous assignons les prochaines tâches à exécuter. Enfin, nous communiquions via Discord pour toujours rester en contact, dans l'éventualité d'une incompréhension.

Outils utilisés :

Pour réaliser ce projet nous avons utilisé différents outils pour couvrir nos besoins.

Figure 1- Logo Eclipse

Nous avons décidé d'harmoniser l'utilisation d'IDE, pour ce faire nous avons opté pour Eclipse. Eclipse est un IDE spécialement adapté pour convenir aux projets en Java. Cette harmonisation fut importante pour s'affranchir des problèmes d'installation liés à nos libraires, ainsi, nous pouvions nous aider dans l'installation des librairies mais aussi lorsqu'une personne rencontre un problème lié à l'exécution du code/l'installation. Ainsi pour s'assurer que le groupe utilise les mêmes outils de développement nous avons décidé d'utiliser l'outil qui nous convenait en majorité et qu'on utilisait.

Figure 2- Logo Discord

Pour mener nos réunions à bien, nous avons choisi Discord. En effet, nous avons considéré qu'il était important de nous réunir comme nous l'avons vu précédemment. L'avantage de Discord était l'historisation de message mais aussi la possibilité de créer plusieurs salons afin de partager des ressources. Grâce à cela, l'utilisation d'un outil pour sauvegarder nos documents n'était plus nécessaire. En utilisant Discord, nous faisions d'une pierre deux coups, nous avions à la fois un outil de communication mais aussi un outil de suivi et de sauvegarde.

Figure 3- Logo Trello

Afin de faciliter la gestion de projet et des tâches, nous avons utilisé l'outil de gestion Trello, grâce à celui-ci nous avons pu organiser la répartition, la création et le suivi sur l'avancement des tâches. Trello nous a aussi permis de générer un diagramme GANTT à partir des tâches que nous avons ajoutées au fur et à mesure du projet. Trello permet aussi l'assignation de tâche et envoie également un rappel à l'utilisateur lorsqu'une tâche arrive à échéance.

Pour l'échange de code nous avons utilisé GitHub. GitHub est un outil permettant d'avoir une historique des modifications du code, il permet également de faciliter le partage et la synchronisation du code entre les membres du groupe, notamment via la gestion de conflits.

Figure 4 - Logo GitHub

Vue :

Design Graphique du jeu :

Afin de définir le style graphique du jeu nous nous sommes inspirés du r/place de Reddit où la France a occupé une très grande place au sein de cet évènement. Cet évènement arrive tous les 4 ans faisant participer des millions d'internaute sur une toile de 4 000 000 de pixels afin de produire les plus beaux pixel art[1]. Chaque utilisateur de Reddit peut alors placer un pixel toute les 5 minutes. La France a fini premier de cet évènement avec 185 000 pixels placés. Nous avons ainsi voulu rendre hommage à cet évènement et proposer une interface faite en pixel art.

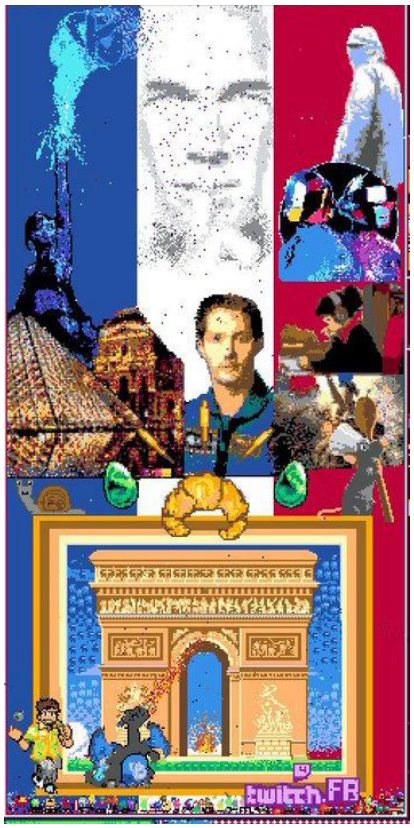


Figure 6 - Pixel art de la France durant le r/Place 2022

Pour cela nous avons créé notre propre carte en pixel art, ainsi que nos propres tuiles et même les éléments de l'interface de jeu.

Nous avons voulu redesigner l'entièreté du jeu afin de crée une version « pixel art » de *The Island.* Pour ce faire nous nous sommes inspirées du thème « des fonds marins » pour l'interface de jeu, et le thème de « l'ile déserte » pour les tuiles.

Figure 5 - Tuile par defaut

[1] Le **pixel art** désigne une composition numérique de carrés, utilisant un nombre de couleurs limité et une définition d'écran basse.

Librairie utilisée :

Pour mener à bien ce projet nous avons décidé d'utiliser la librairie JavaFX qui est la librairie graphique principale de JavaSE.

Figure 7 - Logo JavaFX

Ce choix a été motivé par le fait que les membres de l'équipe Vue avait déjà fait des projets en SWING et désirais tester une nouvelle librairie durant ce projet pour apprendre l'utilisation d'une librairie en plus.

De plus, JavaFX permet une portabilité sur différentes plateformes, ordinateur, mobile et même application web. Nous pensions alors qu'il était pertinent d'utiliser cette librairie afin de pouvoir proposer au client une évolution possible en proposant le portage de son application sans devoir repartir de 0.

Nous avions cependant démarré le projet en utilisant SWING, cependant nous rencontrions parfois certains problèmes lors de son utilisation qui n'était pas très bien documenter sur des sites d'entre aide tel que StackOverFlow. Nous avons remarqué que sur ces mêmes forums il y avait enrobement de sujet avec de l'activité concernant la librairie JavaFX.

Afin d'utiliser JavaFX il est nécessaire d'installer la librairie préalable sur le site de JavaFX et de l'implémenter en tant que «user library». Sur Eclipse il est notamment nécessaire d'installer e(fx)clipse dans le « marketplace » d'Eclipse.

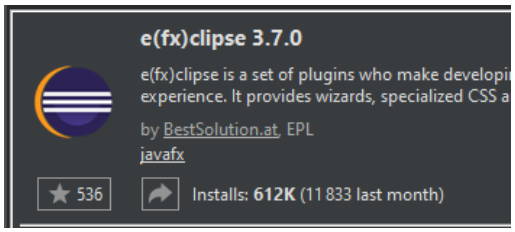


Figure 8- Extension du market place

La carte :

Toujours dans la même volonté qu'au départ nous avons donc proposer notre propre fond d'écran pour le jeu à l'aide de *Tiled* et *Photoshop*.

Figure 9 - Fond de fênetre en .png

Dans ce fond nous retrouvons à gauche la partie qui servira de fond pour la grille de jeu et sur la partie droite nous avons les emplacements réserver aux instructions, à la main et à l'affichage de la tuile de prévisualisation.

Une fois l'image en .png générer avec Photoshop il nous suffit d'ajouter cette image comment fond de notre fenêtre pour commencer à y ajouter les éléments du plateau, de la main, du texte, du dé, de l'hexagone de prévisualisation et du bouton pour passer son tour.

```
public void init_fenetre(AnchorPane tileMap)
{

    // lien vers le background.
    String filePath = "images/bg.png";
    File file = new File(filePath);
    String localUrl = file.toURI().toString();
    Image img = new Image(localUrl);
    BackgroundImage bImg = new BackgroundImage(img,
                                               BackgroundRepeat.NO_REPEAT,
                                               BackgroundRepeat.NO_REPEAT,
                                               BackgroundPosition.DEFAULT,
                                               BackgroundSize.DEFAULT);
    Background bGround = new Background(bImg);
    tileMap.setBackground(bGround);
```

Figure 10 - Code pour mettre un fond de fênetre

La zone de prévisualisation :

Nous avons en haut à droite la zone de prévisualisation qui est une zone dans laquelle nous affichons des informations utiles pour l'utilisateur.

Nous allons définir pas à pas chaque élément de la zone de prévisualisation afin d'expliquer leur utilité et certaines fonctions clés qui lui sont associés.

Nous commencerons par la zone de texte, puis nous continuerons par la tuile de prévisualisions pour finir nous parlerons du dé.

Figure 11- Zone de prévisualisation

La zone de texte :

La zone de texte est une zone dans laquelle nous affichons les instructions, nous pouvons voir sur la figure ci-contre qu'une instruction est afficher demandant au joueur de placer un pion sur le plateau. Afin de permettre cela nous avons initialiser une variable « Text » en lui assignant une position, une couleur, une police qui est une police libre de droit nommé « Broken Code Console » et enfin un « wrapping [2]». Nous avons aussi la fonction changeText(String) qui permet de mettre à jour le texte affiché.

```
public void init_zone_text(AnchorPane tileMap){
    textInstruction.setText("");
    textInstruction.setX(730);
    textInstruction.setY(85);
    textInstruction.setWrappingWidth(330);
    textInstruction.setTextAlignment(TextAlignment.CENTER);
    textInstruction.setFont(Font.loadFont("file:images/brokencode.ttf", 17));
    textInstruction.setFill(Color.valueOf("#5f280f"));
    tileMap.getChildren().add(textInstruction);
```

Figure 12 - Fonction d'initialisation de la zone Texte

[2] Wrapping désigne le conteneur invisible dans laquelle est contenue le texte, sa taille agira donc comme un rectangle de largeur X pixels, que le texte ne pourra pas dépasser. La longueur du texte quant à elle est de la taille de l'écran.

L'hexagone de prévisualisation :

Nous avons ensuite la tuile de prévisualisation, qui est un hexagone avec un point d'interrogation lorsque aucune tuile n'est survolée, ou qu'un effet n'est retirer du plateau. En effet, au survol d'une tuile sur le plateau ou dans la main, l'hexagone de prévisualisation affiche cette tuile, afin de permettre à l'utilisateur de mieux voir la tuile, ou le pion sélectionner.

Figure 13- Différentes prévisualisation possible

La mise à jour de cette hexagone se fait lorsqu'un case du plateau est survolée par le joueur, lorsqu'une tuile de la main est survolée ou cliqué par le joueur lors de la phase 1, lorsqu'un effet vert est tirer lors de la phase 3 et enfin lorsqu'il survole un pion sur le plateau.

Le dé créature :

Le dé est un rectangle qui lorsque l'on clique dessus affiche l'image correspondante au resultat du dé créature obtenu.

Le résultat est obtenu lors du clique dans sur le dé lancer à l'aide de « l'event handler » du dé qui genère un nombre aleatoire entre 0 et 3 lors du clique, appliquant ainsi l'image désirer dans le dé est actualisant ainsi la variable de résultat du dé.


Figure 15 - Différentes faces du pion

```
public void set_de_handler(Vue vue) {
    Rectangle de = vue.getZonePreview().getDe();
    ZonePreview zp = new ZonePreview();
    de.addEventHandler(MouseEvent.MOUSE_CLICKED, e -> {
        de_clic = true;
        resultat_de = (int)(Math.random() * 3) + 1;
        switch(resultat_de) {
        case 1: //Requin
            vue.changeText("Tu as obtenu un requin !!");
            de.setFill(new ImagePattern(zp.get_dice_img(1), 0, 0, 1, 1, true));
            break;
        case 2:
            //vue.changeText("Tu as obtenu une baleine !!");
            de.setFill(new ImagePattern(zp.get_dice_img(2), 0, 0, 1, 1, true));
            break;
        case 3:
            //vue.changeText("Tu as obtenu un serpent de mer !!");
            de.setFill(new ImagePattern(zp.get_dice_img(3), 0, 0, 1, 1, true));
            break;
        }
    });
}
```
Figure 14 - Handler du clique du dé créature

La main du joueur :

La main du joueur peut être constitué de pion pour la phase d'initialisation ou d'hexagones qui correspondent aux tuiles rouges piochés par le joueur.

Pour ce faire, nous avons dans un premier temps initialiser la liste des 10 pions explorateur que chaque joueur doit placer en début de partie sur le plateau. L'initialisation des pions & des tuiles de la main se font avec des placeholders[3] ayant une visibilité mise à « false » dès l'initialisation.
Les pions seront alors affichés grâce à la fonction update_main_pion qui affiche les placeholders en fonction des pions dans la main du joueur.

Le même principe est appliqué pour les tuiles rouges avec la fonction update_main_joueur. Celles-ci prennent la main du joueur actuel en paramètre et affiché également les effets que contient la main du joueur sur la fenêtre. Cependant, pour les tuiles une eventhandler au survol et au clic sont crées afin de faire apparaître cette tuile dans l'hexagone de prévisualisation.

```
public void update_main_joueur(Joueur[] tabJoueurs, int joueurActif)
{
    for(int i=0;i< tabJoueurs[joueurActif].getListEffetMain().size();i++)
    {
        Type_effet effet_main = tabJoueurs[joueurActif].getListEffetMain().get(i);

        Vue v = new Vue();
        hexHand[i].setFill(new ImagePattern(v.get_effet_image(effet_main), 0, 0, 1, 1, true));
        hexHand[i].setStroke(Color.RED);
        hexHand[i].setStrokeWidth(5);
        hexHand[i].setVisible(true);
    }

    for(int j=tabJoueurs[joueurActif].getListEffetMain().size();j<6;j++)
    {
        hexHand[j].setVisible(false);
    }
```

Figure 16 - Fonction udpate_main_joueur

[3] Placeholder – Espace reservé

Les tuiles :

Dans ce jeu, les tuiles sont des hexagones, nous avons alors décomposer les tuiles en deux parties, tout d'abord la partie de l'affichage de la grille de jeu, fait en hexagone pointu et les effets qui sont eux des hexagones couchés. Pour la grille, nous avions besoin de 3 images pour représenter les tuiles terrains, les tuiles montagne, plage et forêt. Nous avions crée un design au préalable sans se soucis de la visibilité des pions sur ceux-ci ce qui nous à conduit à crée des version « pâle » de ces designs afin de facilité la distinction des pions de la case sur laquelle ils se trouvent.

Concernant le plateau, nous avons crée une grille d'hexagone à l'aide d'un double décalage. Lorsque la case était pair, il fallait qu'elle soit décaler de de décalage initiale pour afficher deux tuiles successivement et le multiplié par 1.5 (le décalage est de « l » et le décalage pour une ligne sur deux et de l+(1/2*l) où l est la longueur d'un hexagone). De plus il nous fallait faire commencer chaque ligne à un point de départ diffèrent compte tenu de la non uniformité dans le nombre de cases que contient chaque ligne, cela à été gérer à l'aide de switch prenant en compte la variable j (la ligne) et retourne un décalage correspondant au début de cette ligne, il suffit ensuite de toujours appliquer un décalage de « l » tant que la ligne n'est pas fini.

Figure 17 - Grille du jeu

De plus, chaque case est en fait un hexagone auquel sont attribués des event_listener de clique et de survol affiche de pouvoir : cliquer sur une case et afficher sa prévisualisation dans l'hexagone de prévisualisons de la clique. Enfin nous avons un dernier switch qui affiche la bonne image en fonction du type de tuile dans le plateau (montagne,mer,plage et fôret).

Figure 18 - Exemple de tuile personnalisé.

Concernant, les tuiles effets nous avons dans un premier temps refait l'intégralité des tuiles du jeu en suivant le style graphique que nous avons établie lors de la phase de conception du design graphique du jeu, cependant il nous était compliquer d'intégrer directement des bordures vertes ou rouge, car le fichier lu par la vue est un fichier en .png (.jpg) et ne peux dont pas être hexagonale, si nous avons de la transparence dans une image celle-ci sera afficher dans l'hexagone rendant impossible d'avoir un hexagone proportionnelle en utilisant cette méthode. C'est alors que nous avons décider de crée directement les bordures verte et rouge au sein de la vue en utilisant les propriétés « setStroke » et « setStrokeWidth » que nous paramétrions à vert ou rouge, et à une épaisseur de bordure de 10 pixels.

Ainsi, lorsqu'un effet sera affiché dans la zone de prévisualisation ou la main, non seulement l'image remplira l'hexagone en question, mais aussi les bordures seront générées en adéquation avec la couleur de la tuile dans le modèle. Tout cela est gérer par la fonction get_effet_image() qui est un switch qui prend en paramètre l'effet de la tuile et retourne l'image correspondante, lors d'un affichage dans la main, seulement les tuiles rouges sont présente ainsi la bordure de toutes les tuiles de la main seront en rouge, lorsque l'utilisateur clique sur la plateau pour retirer une tuile du terrain une vérification de la couleur est alors effectué, si la tuile est immédiate nous l'affichons uniquement dans la zone de prévisualisions avec une bordure verte et si elle est rouge nous l'ajoutons à la main et à la prévisualisation avec es bordures rouges.

Les boutons :

Dans le jeu plusieurs boutons ont été implémentés, dans les menus d'accueil et dans le jeu.

Ces boutons sont des objets de la classe Rectangle de Java FX. Pour gérer quand l'utilisateur clique dessus nous avons ajouté un EventListener à chacun d'entre eux. Cela permet que lorsque l'utilisateur clique ou passe sur l'objet, la fonction associée à l'évènement se déclenche. Par exemple, lorsque l'utilisateur passe au-dessus des boutons "continuer" et "jouer" une ombre apparait et lorsque l'utilisateur clique dessus la vue change.

Les créatures :

Les créatures du jeu sont les serpents de mer, les requins et les baleines. Lorsqu'elles sont initialisées dans le contrôleur pour le modèle, on les initialise également dans la vue.

Dans la vue, les créatures sont associées à un objet de la classe Rectangle de Java FX. Comme pour les boutons on leur ajoute un EventListener. Celui-ci a pour effet lorsque l'utilisateur passe sur une créature, l'image de la créature apparait dans l'hexagone de prévisualisation. Et si l'utilisateur clique sur la créature celle-ci sera alors sélectionnée, la variable "creature_cliquee" dans le contrôleur va être mis à jour avec la créature sélectionnée.

Au niveau de la vue, la créature aura un contour noir pour montrer que l'on a cliqué dessus.

Les pions :

Les pions sont initialisés dans le contrôleur pour le modèle puis dans la vue. Les pions de la vue sont la représentation des Explorateurs du modèle. Dans la vue, les pions sont associés à un objet de la classe Circle de Java FX. Comme pour les créatures et les boutons on leur ajoute un EventListener. Pour les pions ce dernier agit comme pour les créatures à quelques exceptions près. L'hexagone de prévisualisation, prend la couleur du pion lorsque l'utilisateur passe dessus.

Et lorsqu'il clique dessus c'est la variable "joueur_clique" qui se met à jour avec l'explorateur que l'on a séléctionné.

Modèle :

Après mûre réflexion nous avons choisi de faire un tableau d'objet Case pour stocker le plateau de jeu. Le centre du plateau, qui contient les tuiles de terrain, sont des objets Tuiles qui héritent de la classe Case. En faisant de la sorte, il est possible de distinguer une tuile mer ou une tuile terrain. Il suffit de récupérer le terrain et vérifier qu'il ne soit pas null.

De plus, nous avons fait le choix d'ajouter une liste de voisins dans l'objet case, qui contient les voisins d'un hexagone.

Stockage du plateau

Le plateau est la première implémentation qui a été effectué au niveau du modèle. C'est un élément central puisque toutes les interactions prochaines se feront à partir de ce plateau. Le défi majeur de la représentation du plateau est son irrégularité. Comment représenter un plateau qui n'a pas le même nombre de case à chaque ligne ; les cases étant hexagonal comment connaitre le voisin en haut à gauche qui n'est pas aux mêmes coordonnées relatives en fonction de la ligne.

Nous avons décidé après plusieurs prototypes de partir sur un tableau à deux dimensions, c'est à dire un tableau qui contient des tableaux, fixe et où certaines cases du tableau seront vides et ne contiendront pas de "case de plateau". Toutes ces cases sont iniatilisés manuellement. Pour chaque case sont défi un identifiant avec une lettre pour la ligne et un nombre pour la colonne, i.e. la première case du plateau est identifiée comme "a1" ; les coordonnées dans le tableau, la liste des pions présents sur la case, un polygone nécessaire au modèle de la vue, ainsi qu'une liste des case voisine. Cette dernière liste sert à résoudre le problème de l'irrégularité des voisins. Puisque pour trouver les voisins, cela nécessite des calculs avec beaucoup d'exceptions nous avons choisi de faire un graphe de toute les cases avec chaque case initialisée avec une liste de leurs voisins pour ne pas avoir à les recherchés à chaque appel ce qui serait très demandant en ressources de calcul pendant l'exécution.

Dans notre modèle les cases sont considérées comme des cases "mer", il nous fallait donc un moyen de représenter les tuiles de terrain du jeu. Ces tuiles héritent de la case, et récupèrent donc les mêmes arguments, mais on en rajoute des nouveaux. L'information sur le terrain c'est à dire si c'est une tuile plage, forêt ou montagne. Ainsi que les effets de la tuile. Ces effets sont eux même caractérisé par deux informations, l'effet en lui-même c'est à dire ce qu'il fait, ainsi que sa couleur, verte ou rouge, en fonction de si l'action s'effectue immédiatement ou non lorsque l'on retire la tuile du plateau.

Recherche voisin :

Pour initialiser les voisins on boucle à travers tout le tableau. Au maximum une case a 6 voisins puisqu'elle est en hexagone. La recherche de voisin se fait en fonction de la ligne, si elle est paire ou impaire. On a utilisé le tableau suivant, créer à partir d'observation, pour faire les vérifications et les initialisations des voisins.

Tableau de calcul voisin

Ligne	*Case d'origine*	*Voisin en haut à gauche*	*Voisin en haut à droite*	*Voisin au côté gauche*	*Voisin au côté droite*	*Voisin en bas à gauche*	*Voisin en bas à droite*
Paire	[i,j]	[i-1,j-1]	[i-1,j]	[i,j-1]	[i,j+1]	[i+1,j-1]	[i+1,j]
Impaire	[i,j]	[i-1,j]	[i-1,j+1]	[i,j-1]	[i,j+1]	[i+1,j]	[i+1,j+1]

Ici les coordonnées i (ligne) et j (colonne) sont celles du tableau.On peut voir qu'en fonction de si la ligne est paire ou impaire, le décalage n'est pas le même.

Lorsque l'on doit mettre à jour les caractéristiques des voisins d'une case, au lieu de réutiliser le même calcul, on utilise leur coordonnée pour aller chercher les informations dans le plateau.

Pions :

Voici une liste exhaustive des différents pions : Baleine, Bateau, Explorateur, Marin, Requin et SerpentMer.

(Deux Marins accompagnent toujours un bateau afin qu'un bateau prenne 3 cases sur les 6 de la liste de pions d'une case)

Chaque pion est représenté dans le code source par une classe. Nous avons convenu que toutes ces classes hériteraient de la classe non instanciable Pion. Cela afin de limiter la duplication de code. Ainsi, les méthodes et attributs que tous les pions possèdent, comme par

exemple setCoord_index() ou encore nombre_cases_parcourables, ne sont rédigés qu'une fois dans le code pour tous les pions.

Déplacement des pions :

Nous permettons au joueur de se déplacer seulement une case à la fois afin qu'il puisse choisir le chemin par lequel il passe.

Par ailleurs, chaque pion bouge différemment et selon différentes conditions.

Mais tous ces déplacements ont en commun de devoir vérifier si la case destination est pleine, si oui, de retirer le pion de sa case actuelle et de l'ajouter à la case destination.

C'est pourquoi nous avons opté pour l'utilisation de surcharges. Elles permettent d'ajouter des conditions avant d'appeler super.deplacer(). Notamment de permettre un déplacement uniquement dans la mer.

Exceptions pour certains pions :

1. Polymorphismes dus à des déplacements différents :

- Explorateurs : Si un joueur choisit une case destination qui contient un danger pour son explorateur, alors ce dernier est retiré de sa case mais pas ajouté à la destination.
- Bateaux : nécessitent de retirer les pions marins et de les ajouter en fin de déplacement. De plus, un bateau qui possède un explorateur qui rencontre une baleine et un requin/serpent de mer est retiré du jeu.

2. Deux méthodes de déplacement :

- Baleine, Requin, SerpentMer : ils peuvent être déplacés n'importe où sur la carte grâce aux tuiles rouges. Donc leur méthode déplacer ne limite pas les déplacements aux cases adjacentes. C'est le rôle de leur méthode deplacer_limite()

Enlever les tuiles :

Le contrôleur contient un tableau d'objet Case qui correspond au plateau de jeu.

Au début de la phase 3 chaque joueur doit enlever une tuile terrain. Mais dans un ordre précis (plage, forêt, montagne). Avec une condition supplémentaire, la tuile doit être adjacente à une case mer. C'est donc la première chose à vérifier.
Dans le contrôleur, nous avons créé 2 variables globales clic_indexI et clic_indexJ qui sont les coordonnées de la case que le joueur a cliqué. Avec ces coordonnées nous pouvons accéder à la case du tableau souhaité et récupérer un objet de type Case (si n'a pas de terrain) ou Tuile (si possède un terrain). L'objet de type Case, comme dit précédemment contient une liste de voisin. Ainsi, en parcourant cette liste, si au moins un des voisins n'a pas de terrain (null) alors c'est une case Mer.

La deuxième chose à faire est de vérifier si toute les tuiles terrain d'un type ont été enlevé avant de passer au type de terrain suivant. Pour cela, nous avons créé une méthode presence_terrain(Terrain terrain) qui prend en paramètre le terrain dont on souhaite vérifier la présence. Cette méthode parcourt toutes les cases du plateau, on récupère le terrain de l'objet Case grâce à la méthode getTerrain(). Si la méthode presence_terrain renvoie true alors cela signifie que le terrain passé en paramètre existe sur le plateau (et ceci au moins une fois). Ensuite il faut vérifier, que l'utilisateur a bien cliqué sur une tuile d'un type de terrain précis. S'il ne clique pas sur le bon terrain, on réinvite le joueur a cliqué de nouveau sur un bon terrain. Sinon, on récupère l'effet de cette tuile choisi grâce à la méthode getEffet(). En effet, si la tuile choisie est de couleur verte on active son effet associé. Mais si la tuile est de couleur rouge, nous devons l'ajouter à la main du joueur.
Enfin nous devons mettre à jour le modèle ainsi que la vue, c'est-à-dire en effacer la tuile choisie.

Nous devons répéter ces instructions pour les 3 types de terrains.

Placer les tuiles :

Les tuiles qui peuvent être jouer(placer) sont les tuiles de couleurs rouges. On peut placer les tuiles seulement au début de la phase 1.
Dans le contrôleur, un tableau d'objet Joueur a été créer. Et l'objet Joueur contient une liste de Tuile qui représente la main du joueur.
La première chose à faire est de vérifier si la main du joueur contient une tuile. S'il en contient au moins une, il est nécessaire de récupérer la tuile que le joueur a choisi. Une "Variable globale" main_clic a été créer dans ce but. Puis, il s'agit de vérifier si cette tuile est rouge en récupérant l'effet de la tuile cliqué.
Enfin, il ne reste plus qu'a appelé l'effet souhaité.

Effets:

Chaque tuile terrain, contient un effet particulier. Certaines tuiles terrain ont des effets en communs. Chaque effet correspond à une méthode. Ex : tuileRouge_dauphin(i,j) correspond à la tuile dauphin de couleur rouge qui permet de déplacer un explorateur.

Les effets "Vert" :

Les effets de requin baleine, et bateau immédiat consistent à créer un objet de type Requin Baleine, ou bateau, ainsi que de mettre à jour la vue, et ajouter à la liste de pions de la case du plateau, cet objet.

L'effet tourbillon consiste a caché tous les pions de la vue et de vider la liste des pions des cases voisine du plateau. Il suffit d'utiliser le mot clé clear.

L'effet Volcan met un terme au main, la partie est terminé

Les effets “Rouge” :

Les effets de défense du requin et de la baleine consistent a caché ces monstres de la vue et à les supprimer de la liste des pions de la Case du plateau, grâce au mot clé remove.

Les autres effets consistent quant à eux de demander aux joueurs de sélectionner le pion à déplacer, que l’on obtient grâce à la variable globale creature_clique ainsi que sa destination grâce au coordonnées clic_indexI et clic_indexJ.

Contrôleur :

Menu :

Les menus d'accueil du jeu permettent de choisir le nombre de joueurs puis de rentrer le pseudo de chaque joueur. C'est le module de la vue qui s'occupe de les créer et de leur apparence. Mais c'est dans le contrôleur qu'ils sont lancés.

Le premier menu est lancé directement, puis on attend que l'utilisateur choisisse le nombre de joueur grâce à la ComboBox, qui permet de choisir entre 2, 3 ou 4 joueurs. Quand l'utilisateur a choisi, il clique sur le bouton "continuer". On récupère alors son choix dans le contrôleur avec la ComboBox. Le deuxième menu s'affiche et les joueurs sont invités à rentrer leur pseudo. Lorsque c'est fait, ils doivent cliquer sur le bouton "jouer". Le plateau de jeu apparaitra alors à l'écran.

Phase 0 :

La phase d'initialisation est la phase de jeu pour préparer le terrain avant de commencer le jeu. Dans cette phase, les joueurs vont dans l'ordre placer leur pion explorateur sur le plateau en voyant au départ la valeur du pion explorateur, et tous les pions qu'il lui reste. Une fois un pion placer, le joueur suivant doit à son tour placer un pion et ce jusqu'à qu'aucun joueur ne puisse placer de pion à son tour.

Cette phase nécessite cependant plusieurs vérifications, tout d'abord le joueur doit d'abord sélectionner un pion depuis sa main, et choisir une case du plateau où il y'a une tuile terrain (ainsi on ne peut initialiser un pion sur une case mer). De plus, lors de la phase d'initialisation aucune case ne peut contenir plus d'un joueur, ainsi il est nécessaire pour le joueur de placer son pion sur une tuile terrain et une tuile qui ne contient pas encore de pion.

De plus, une fois placer le joueur ne peut plus voir la valeur du pion qu'il vient de placer. Pour réaliser cette phase nous avions besoin de la main de pion de chaque joueur qui sera initialiser avec les 10 pions de départ, une fois initialiser cette liste sera alors afficher dans la

main de chaque joueur avec la fonction update_main_pion permettant l'affichage des pions restant à être placer par le joueur.

Une fois afficher il nous suffit alors de vérifier les conditions précédemment citées. Pour ce faire lorsque nous cliquons sur un pion de la main la variable « main_clic » s'initialise et contient l'indice du pion (dans le tableau de pion explorateur de chaque joueur) est alors assigné à la variable « main_clic » cela permet de s'assurer que le joueur a bien sélectionner un pion avant de vouloir réaliser son action. S'il clique sur le terrain sans avoir sélectionner de pion un message d'erreur s'affiche lui indiquant qu'il doit d'abord sélectionner un pion.

Ensuite, lorsque l'utilisateur clique sur le terrain plusieurs conditions sont à vérifier, tout d'abord que l'utilisateur à sélectionner une tuile terrain ce qui se vérifie grâce à « instanceof Tuile » puis enfin nous vérifions si la case ne contient pas plus d'un pion grâce à la fonction getSize() qui permet de retourne le nombre de pion contenu dans la case.

Pour sortir de cette phase du jeu il nous reste plus qu'à vérifier à chaque placement s'il reste une main de pions vide chez un joueur grâce à la fonction « nombre_main_vide » et il reste un joueur dont la main n'est pas vide nous passons le tour à ce joueur et nous affichons sa main. Si aucun joueur ne possède de pion explorateur dans sa main, nous passons alors à la phase suivante.

Phase 1 :

Cette première phase de jeu, est celle durant laquelle un joueur qui a une tuile rouge dans sa main peut la jouer. C'est une phase que l'on peut passer, et on indique aux joueurs si sa main est vide de le faire. La main du joueur est une liste de maximum 6 tuiles. Elles permettent de déplacer les pions de différentes manières, les tuiles dauphins et bateau, permettent respectivement de déplacer les nageurs et les bateaux de 1 à 3 cases mer. Alors que les tuiles serpent de mer, requins et baleines, permettent de déplacer vers n'importe quelle case sur le plateau.

Pour se faire, le joueur doit d'abord sélectionner la tuile qu'il souhaite utiliser parmi celles disponible dans sa main et dont on récupère l'effet. Il devra ensuite choisir le pion sur lequel il veut utiliser l'effet en question. Le choix se fait en un simple clic sur la pièce que l'on stock dans une variable "joueur_clique" pour les explorateurs et "creature_clique". Une fois tout mit en place la personne qui joue peut cliquer sur la case de destination pour son pion et si c'est valide on déplace le pion.

Beaucoup de verification sont faite pour s'assurer du bon déroulement de cette oppération, on verifie que le ou la joueuse ont bien sélectionné tous les éléments nécessaires, que cela respecte la distance de déplacement ou bien que la case soit vide dans le cas des créatures. Si une de ces choses n'est pas bien effectué, on affiche un message et on attend les nouvelles instructions pour refaire les tests jusqu'à ce que la tuile soit bien utilisé ou que la phase soit passé.

Phase 2 :

La phase 2 correspond au déplacement d'un pion explorateur ou un bateau. Le joueur est autorisé a effectué seulement 3 déplacements. Ce nombre de déplacement est stocké dans une variable appelé nombre_deplacements_restants.
Tout d'abord, il est nécessaire de récupérer le pion choisi par l'utilisateur. Ce pion est stocké dans joueur_clique.
Ensuite, il faut procéder à certaines vérifications. Notamment, vérifier si l'utilisateur a cliqué sur un pion (non null) et que celui-ci est un pion explorateur ou un bateau (grâce à instance of d'une classe). De plus, il faut que le joueur courant soit autorisé à déplacer seulement les pions de sa couleur. Grâce à la méthode getCouleur nous pouvons comparer la couleur du pion de joueur_clique et du joueur courant.
Enfin, il ne reste plus qu'a demandé les coordonnés de destinations au joueur. Ces coordonnés sont stockés dans les variables coordIndexI et coordIndexJ. Aussi, il faut vérifier si le joueur courant peut encore jouer. Si tout est bon nous pouvons appeler la méthode déplacer et décrémenter le nombre de déplacement restant.

Une fois les 3 déplacement effectuer (ou si le joueur a décidé de passer), on passe à la phase suivante.

Phase 3 :

La troisième phase de jeu, est la phase dans lequel on retire les tuiles de terrain. Mais on ne peut pas retirer une tuile n'importe comment. Les tuiles ne peuvent être retiré que si elles sont à côté d'une case mer. On vérifie donc avec la fonction "voisinMer()" que la tuile cliquée est bien voisine avec une case mer. La fonction passe en revue les voisins de la case et retourne un booléen vrai si elle trouve une case.

Une fois la première vérification effectuée, une seconde est mise en place qui verifie ensuite de quel type de terrain la tuile est. En effet le jeu requiert que l'on ne puisse pas enlever les cases dans n'importe quel ordre. On doit commencer par la plage, puis les forêts et enfin la montagne. On vérifie donc à chaque fois s'il reste des tuiles plage, et si c'est le cas on vérifie que la tuile sélectionnée en soit bien une, et ainsi de suite avec les autres types de terrain.

Lorsque les vérifications sont faites, on retire la tuile du plateau, on récupère son effet et on la transforme en case. On doit vérifier quel est la couleur de l'effet. Si celui-ci est rouge on l'ajoute simplement à la main du joueur. Mais si c'est un effet vert, alors on appelle la fonction correspondante. En dernier on met à jour les listes voisins des cases voisines pour leur donner l'informations que la case est maintenant une case de mer. C'est nécessaire pour qu'elles puissent être enlevé aux prochains tours.

Phase 4 :

La phase 4 correspond au lancé de dé de créature. Ce dé nous permet de déplacer une créature marine si elle est sur le plateau.

Tout d'abord, on vérifie que le joueur a cliqué sur le dé. Ensuite, nous devons tirés un nombre aléatoire en 1 et 3 (Requin, Serpent de Mer, et Baleine) et selon ce nombre, nous pouvons mettre à jour la vue en affichant l'icône de la créature tirée.
La première chose à faire est de vérifier si ce monstre existe sur le plateau. C'est ce que fait la

méthode estPresent_requin() (resp estPresent_baleine() et estPresent_serpentMer()). Cette méthode parcourt la liste de pions de chaque case du plateau. Si au moins un pion est une instance d'une créature alors on renvoie un booléen true. Sinon on affiche un message et on passe au tour suivant.

Ensuite, il est nécessaire de récupérer la créature que le joueur souhaite déplacer. L'objet creature_clique contient cette entité. Puis il faut vérifier que cette créature est bien du même type que celle que nous avons tiré avec le dé (grâce à instanceof).

Enfin, il ne reste plus qu'a demandé au joueur les coordonnées de destination de ce monstre. Nous allons les stocker dans les variables coordIndexI et coordIndexJ. Et nous pouvons appeler la méthode deplacer_limite qui va se charger d'effectuer les modifications du modèle.

Une fois le déplacement réussi on passe au joueur suivant. Ces instructions sont à répéter pour les 3 créatures

Gestion du clic :

Afin de gérer les cliques sur les objets du plateau chaque élément est doté d' « event handler » à l'écoute du clique sur cette objet, une fois l'objet cliquer les fonctions présentes dans cette event handler seront exécuter, et certaines variable globales seront mise à jour en fonction des instructions écrite dans l'eventHandler du clique.

Afin d'effectuer une vérification des objets précédemment cliquer, les cliques sur le plateau servirons à la fois de clique de validation afin de passer à la phase suivante mais aussi de clique sur une case, car nous pouvons crée plusieurs eventHandler sur un seul objet, qui s'exécuterons l'un après l'autre en fonction de leur ordonnancement dans le code.

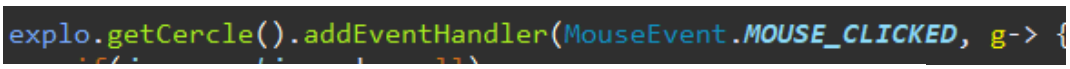


Figure 19 - Exemple d'eventHandler sur le clique

Conclusion

Qu'avons-nous appris ?

De la première utilisation réelle de Java, en passant par la découverte d'une nouvelle bibliothèque, jusqu'au renforcement de notre gestion de projet, The Island fait partie des expériences dont nous sommes ravis d'avoir pris part.

Nous avons acquis de l'expérience dans un cadre proche du professionnel (cahier des charges, une date à respecter, une soutenance et une équipe projet)

Possibilité d'évolution

Nous avons choisi d'utiliser la bibliothèque JavaFX en partie car nous apprécierions l'idée permettre un affichage réactif du jeu. Cela permettrait de rendre le jeu compatible smartphone, appareil que tout le monde utilise aujourd'hui.

Pour le moment, nous nous sommes contentés de fixer l'affichage à un petit nombre de pixel afin que le jeu soit compatible même sur les petits écrans PC à faible résolution. Mais cela a pour conséquence l'affichage d'une petite fenêtre sur les écrans grands/à haute résolution et empêche le plein écran.

De plus, nous trouvons que le multi-joueur sur un seul écran est une expérience peu pratique car on peut aisément voir la main des autres joueurs. Un multi-joueur en ligne fournirait à la fois un nouveau défi à l'équipe et une meilleure expérience utilisateur.

En somme, nous souhaiterions adapter le jeu aux standards actuels.

Annexe :

Biblio :

Références graphiques:

https://en.wikipedia.org/wiki/R/place

Representation Hexagonal en informatique:

https://www.redblobgames.com/grids/hexagons/

https://web.mit.edu/sp.268/www/hex-notes.pdf

https://stackoverflow.com/questions/20734438/algorithm-to-generate-a-hexagonal-grid-with-coordinate-system

https://www.developpez.net/forums/d1278348/c-cpp/c/plateau-jeu-hexagonal/

Information jeu de societe:

http://jeuxstrategieter.free.fr/The_island_complet.php

Convention Sun:

https://sites.google.com/site/sureshdevang/sun-basic-code-conventions

Documentation JavaFX:

https://koor.fr/Java/TutorialSwing/comparatif.wp#:~:text=JavaFX%20est%20désormais%20l%27API,%26%20tablettes%20et%20applications%20Web).

https://fxdocs.github.io/docs/html5/

Configurer JavaFX sur VSCode

Création exécutable Java:

https://taylorial.com/cs1021/Jar.htm

https://stackoverflow.com/questions/39710286/export-javafx-project-to-a-runnable-using-eclipse

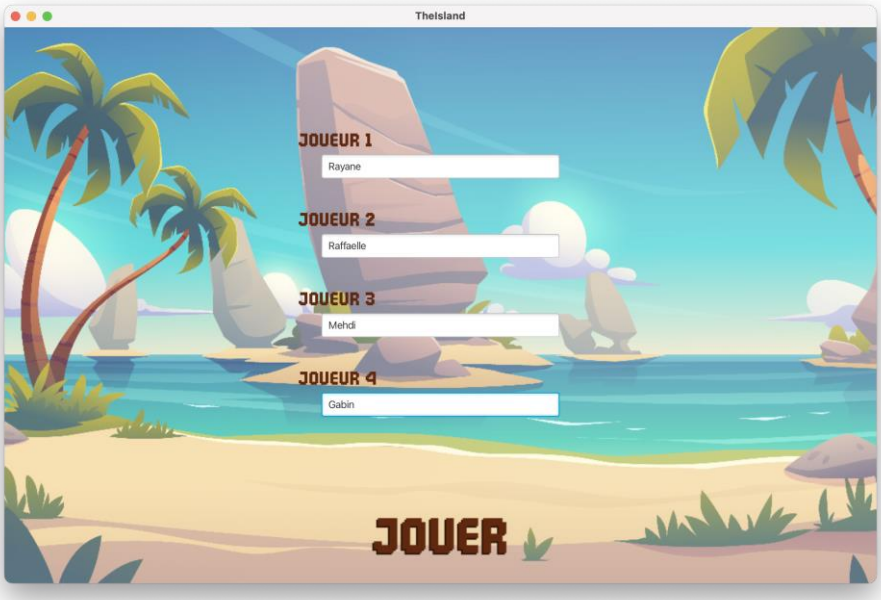
Table des figures :

Manuel utilisateur :

Menu :

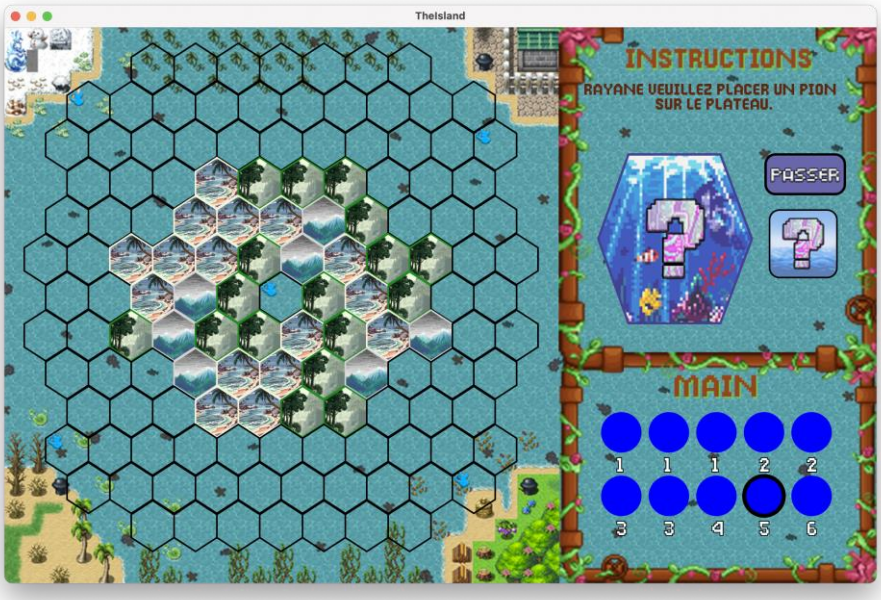
Le premier écran du menu permet de choisir le nombre de joueur. Une fois choisit, on clique sur "continuer”.

L’écran suivant permet d’entrer les pseudos. Une fois fait, on clique sur “jouer” afin de commencer une partie.



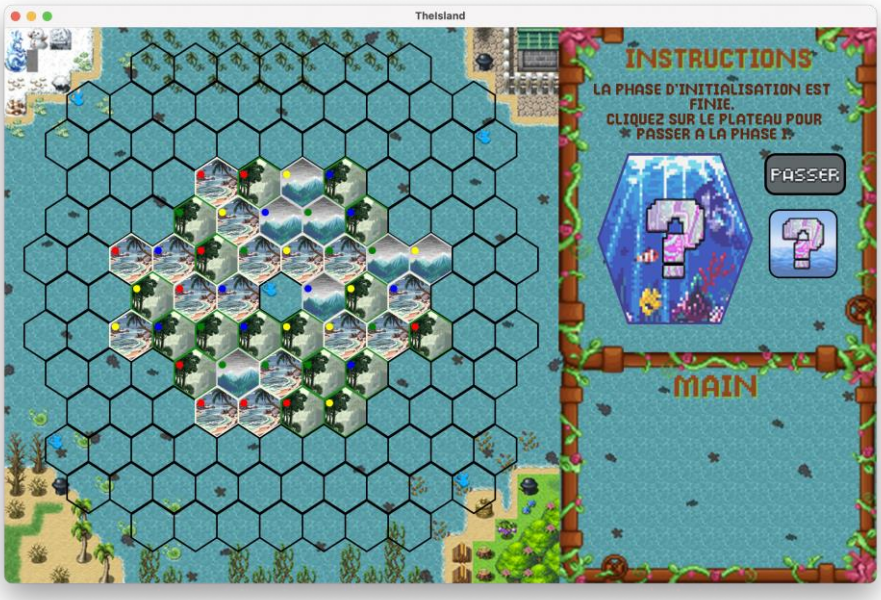
Phase 0 :

Pour placer un premier pion, on clique sur un pion de notre main, puis sur l'hexagone que l'on souhaite. Par exemple, sur la photo ci-dessous, je sélectionne le pion de valeur 5.



Après avoir cliqué sur la case sur la case que j'ai choisie, c'est au tour du joueur suivant de placer son explorateur.

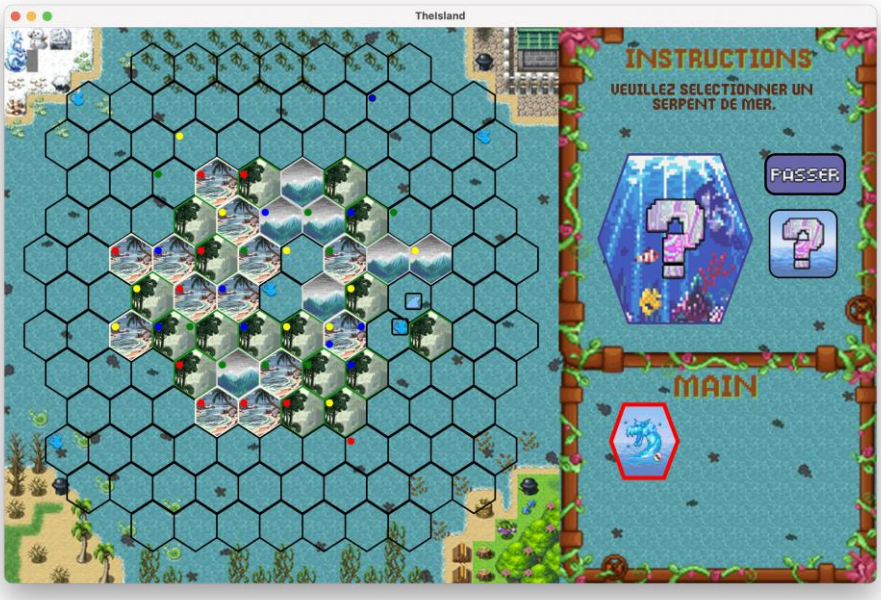
Une fois tous les pions placés, cliquez sur le plateau pour passer à la phase 1.



Phase 1 :

Pour utiliser une tuile rouge de sa main, il suffit de cliquer sur la tuile voulue dans la section main.

Une fois cliquée, l'effet de la tuile s'active.



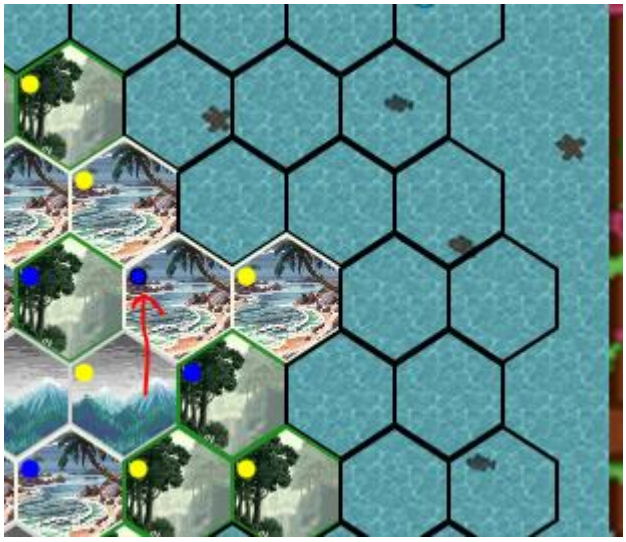
Phase 2 :

En arrivant à la phase 2, un message d'instructions apparaît à l'écran. Il marque le début de la phase.

Pour commencer cette phase, il faut cliquer n'importe où sur le plateau, et le message d'instruction changera :

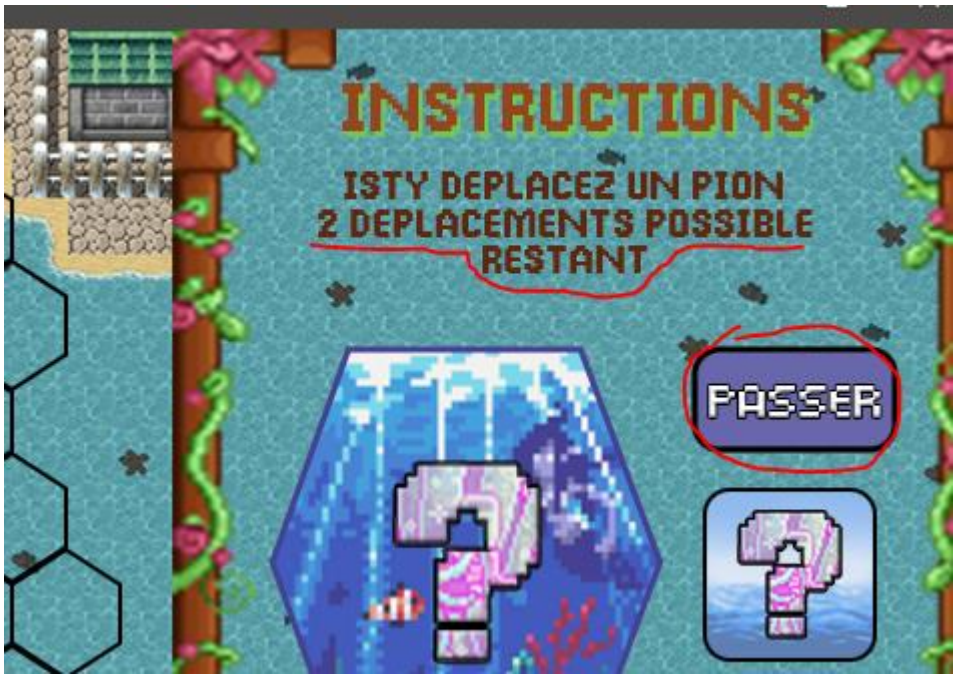
Ensuite nous devons cliquer sur le pion (cercle de couleur) que nous voulons déplacer. Il faut que ce pion vous appartienne.

Une fois le pion sélectionner un cercle noir devrait apparaitre autour du cercle pour marquer sa sélection :



Et enfin nous devons cliquer sur une case adjacente à la case du pion sélectionner :

Vous savez maintenant comment déplacer un pion. Vous pouvez le faire 3 fois au maximum. Vous pouvez aussi passer votre phase en cliquant sur passer :



Il ne reste plus qu'à cliquer n'importe où sur le plateau pour passer à la phase suivante :

Phase 3 :

Phase 4 :

En arrivant sur la phase 4 nous obtenons ce message :

Pour commencer cette phase il est nécessaire d'appuyer n'importe où sur le plateau. Le message change.
Ensuite, nous devons lancer le dé. Le dé est le carré avec un point d'interrogation qui se trouve en dessous du bouton passer. Nous devons cliquer dessus pour lancer le dé :

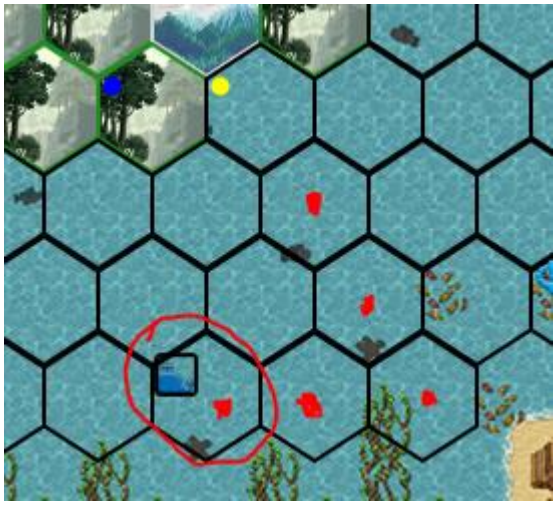
Ici nous obtenons une baleine mais vous auriez pu avoir un requin ou un serpent de mer.

Il faut cliquer n'importe où sur le plateau, et si le jeu contient au moins une créature que vous avez tiré au dé, vous êtes invité à le déplacer :

Il faut maintenant cliquer sur la créature (que l'on vient de tirer au dé) présent sur le plateau que nous souhaiterions déplacer. Une fois cliqué sur la créature, un carré noir apparait autour de celle-ci pour marquer sa sélection.

Ensuite il faut cliquer sur une case adjacente pour la déplacer :

Nous pouvons déplacer la créature d'autant de case que nous voulons. Cependant il est nécessaire de toujours cliquer sur une case adjacente. Comme le montre la capture ci-dessous, pour aller à la case entourer, il faut que l'utilisateur clique sur une case adjacente à celle où se trouve le pion. (Les clics de l'utilisateur sont représentés par des points dans la capture ci-dessous).



La raison à cela est que si nous cliquons sur une case qui n'est pas adjacente à celle du pion, la phase se termine. Comme le montre la capture ci-dessous, l'utilisateur clique sur une case non adjacente (représenter par le point rouge), et le message de fin de phase apparait.

Et c'est de cette façon que nous terminons la phase du dé et en même temps notre tour. Chaque tour du jeu suivra ces différentes phases jusqu'à la découverte du volcan.