Patrick Alexandre Sardinha
Patrick.Sardinha@etu.unige.ch
Numéro étudiant : 16-320-988

PROJET DE BACHELOR

SPACE ENCODERS

Sous la direction de Didier Buchs

Assistants:

Dimitri Racordon
Stefan Klikovits
Damien Morard

Université de Genève
Faculté des Sciences
Département de Sciences Informatiques

Année 2019

TABLE DES MATIERES

Pages

INTRODUCTION

Le but du projet "BSc-educative-gaming-2019" consiste à créer un jeu éducatif lié au monde de la programmation informatique et dédié à un public relativement jeune. Il est donc intéressant de se demander pourquoi nous voulons mettre en place pour notre projet un jeu éducatif, c'est-à-dire un *Serious Game*. En effet, comme le but du projet est de faire apprendre et comprendre les mécanismes de base de la programmation informatique, nous allons donc nous focaliser sur ce concept.

De nos jours, les jeux de société sont présents sur divers supports. Nous pouvons distinguer les jeux physiques et les jeux informatisés. Cependant, pour un jeu donné, il existe souvent une version sur chacun d'entre eux, ce qui permet ainsi de pouvoir tirer les avantages des deux plateformes. Pour notre projet, le développement du jeu aussi bien au niveau physique qu'informatique est intéressant à mettre en place. La partie informatisée permet ainsi de lier les concepts de *Serious Game* et de *Game-based learning* puisque nous voulons faire apprendre un savoir en passant par un jeu vidéo.

Afin de mener à bien notre projet, nous allons dans un premier temps définir les concepts essentiels et comprendre quels sont les éléments à mettre en place. Dans la deuxième partie, nous parlerons du jeu éducatif que nous avons créé. Nous discuterons de l'idée globale du jeu, des bases sur lesquelles il s'appuie, de ses buts et de ses règles, ainsi que des éléments qui le composent. Nous aborderons également les évolutions de notre jeu tout au long du projet, pour enfin voir en quoi il répond au concept de *Serious Game.* Nous verrons dans la troisième partie, la partie informatisée du jeu. Nous y détaillerons l'organisation du projet et du code ainsi que les étapes de développement. Nous parlerons ensuite dans la quatrième section des problèmes que nous avons rencontrés ainsi que des limites du travail pour enfin discuter dans la dernière partie, des améliorations qui pourraient être mises en œuvre tout en abordant les perspectives pour le projet.

1. ETAT DE L'ART

Dans la partie précédente, nous avons évoqué le concept de *Serious Game* concernant la création de notre jeu éducatif. Il est ainsi primordial de définir ce qu'est un *Serious Game*. Il s'agit d' *"un jeu dont le but principal n'est pas de divertir ou de s'amuser"* [1]. Ce type de jeu n'est pas uniquement dédié à divertir les participants, ce qui diffère des jeux classiques, mais permet aussi aux joueurs d'acquérir l'apprentissage d'un savoir. Comme le dit J. Alvarez en faisant allusion aux jeux vidéo, *"[…] l'objectif est de combiner à la fois des aspects sérieux (Serious) tel que, de manière non exhaustive, l'enseignement, l'apprentissage, la communication, ou encore l'information, avec des ressorts ludiques issus du jeu vidéo (Game)"* [2]. Notre jeu se basera ainsi sur ce concept puisque le but principal est de faire apprendre aux joueurs des notions de base de l'informatique.

Ainsi, pour créer un *Serious Game,* il semble que certains éléments soient indispensables à mettre en place [3]. Tout d'abord, il faut que ce dernier raconte une histoire. Cela permet à la fois de faciliter l'immersion des joueurs et également d'accentuer leur motivation. Il y a ensuite le concept de *Gamification*. Il s'agit d'une dynamique de jeu mettant en place des classements, des récompenses, etc., qui permet ainsi aux joueurs de s'appliquer davantage et de se surpasser. Le *feedback* individualisé est aussi un autre élément important à mettre en place. Le fait d'interagir directement avec le jeu permet aux joueurs d'apprendre de leurs erreurs et ainsi de s'améliorer puisqu'ils sauront où et pourquoi ils se sont trompés. Enfin, l'élément le plus important est l'objectif d'apprentissage. En effet, un jeu ne peut pas être considéré comme un *Serious Game* si celui-ci ne vise pas à enseigner un savoir.

---

[1] Damien Djaouti et al., « Origins of serious games », in *Serious games and edutainment applications* (Springer, 2011), 25–43.

[2] Julian Alvarez, « Du jeu vidéo au serious game, approche culturelle, pragmatique et formelle » (2007)

[3] GameLearn, Game-based Learning, ludification et Serious Games, [en ligne]. Disponible sur : <https://www.game-learn.com/ce-que-vous-devez-savoir-sur-serious-games-game-based-learning-exemples/>

Après avoir défini les *Serious Games* et leurs éléments principaux, il est à présent intéressant d'introduire le concept de *game-based learning* [4]. Ce dernier est le nom de la méthode qui consiste à utiliser des jeux vidéo pour l'apprentissage. L'idée est donc d'utiliser ce concept pour la version informatisée du jeu et permet ainsi de préserver le but principal du jeu, c'est-à-dire celui de faire apprendre, tout en changeant la plateforme.

L'apprentissage à travers les jeux vidéo implique de nombreux avantages [4] par rapport à un apprentissage "classique" tel que celui fait par des livres ou en classe par exemple. Comme nous l'avons mentionné précédemment, l'interaction des joueurs avec le jeu est primordiale. Or, le principe même des jeux vidéo est l'interactivité puisque tout au long de la partie les joueurs sont obligés de faire des choix. Ainsi, le *game-based learning* est certainement l'un des meilleurs moyens pour apprendre car il s'agit d'un apprentissage basé sur la pratique. De plus, le fait que les joueurs s'impliquent totalement dans le jeu permet d'augmenter la qualité de l'apprentissage - qui est le but principal - et la mémorisation des connaissances acquises.

Il existe de nombreux jeux vidéo basés sur le concept de *game-based learning.* Nous pouvons par exemple citer *"Dragon Box Element"* [5] ayant pour but d'apprendre aux enfants les principes de base de la géométrie ainsi que le théorème d'Euclide. Nous pouvons également parler de *"Pulse !!"* [6] qui est, quant à lui, axé sur le secteur médical et qui vise à entrainer des infirmiers à des situations réelles.

Afin de développer un jeu vidéo, il est nécessaire de définir certains éléments qui sont liés à la base du développement [7]. Il faut tout d'abord choisir à quel genre de jeu vidéo celui-ci appartiendra (par exemple aventure, puzzle, ...) mais aussi pour quelle plateforme le jeu sera

---

[4] GameLearn, Game-based Learning, ludification et Serious Games, [en ligne]. Disponible sur : <https://www.game-learn.com/ce-que-vous-devez-savoir-sur-serious-games-game-based-learning-exemples/>

[5] DragonBox, Dragon Box Elements, [en ligne]. Disponible sur : <https://dragonbox.com/products/elements>

[6] Pulse !!, First Person Healthcare System Simulation, [en ligne]. Disponible sur : <https://www.gamasutra.com/view/news/102373/SGS_Feature_Pulse_First_Person_Healthcare_System_Simulation.php>

[7] WikiHow, Make your own Video Game, [en ligne]. Disponible sur : <https://www.wikihow.com/Make-Your-Own-Video-Game>

destiné (portable, PC, …). Après avoir défini ces deux éléments, il est nécessaire de créer un premier design du jeu dans le but de percevoir ces concepts fondamentaux et ainsi juger si l'idée est viable. Il faut enfin lister les idées et les concepts qui seront à mettre en place tout au long du développement.

Pour ce projet, nous nous baserons donc sur les deux concepts présentés ci-dessus afin de créer une version physique et une version informatique de notre jeu éducatif, que nous avons décidé d'intituler *"Space Encoders"*.

2. Le jeu *"Space Encoders"*

2.1 Idée du jeu et inspirations

Avant de commencer la création du jeu que nous avons intitulé *"Space Encoders"*, nous nous sommes renseignés sur les jeux de société actuels afin de prendre connaissance de ce qui est proposé sur le marché et quels sont les types de jeu les plus en vogue en ce moment. Nous avons ainsi constaté que les jeux de société collaboratifs sont des jeux qui plaisent énormément à tout type de joueur. En effet, ces jeux permettent aux joueurs une approche différente que celle proposée par des jeux classiques. Nous avons donc pensé intéressant de mettre en œuvre ce type de collaboration pour notre jeu.

Pour la création de notre jeu éducatif, nous avons décidé de nous baser sur un jeu déjà existant afin de ne pas débuter de zéro. En effet, cela permet de récupérer des mécanismes de jeu déjà créés et ainsi de les réutiliser mais aussi de les réadapter à nos besoins tout en créant nos propres mécanismes.

En nous renseignant auprès de professionnels de la vente de jeux, nous avons eu plusieurs propositions de jeux collaboratifs mettant en œuvre divers mécanismes de jeu. Parmi différentes propositions, nous nous sommes tournés vers *"Le Ciel Interdit"* [8], un jeu créé par

[8] Cocktail Games, 2018 « Le Ciel Interdit », Réalisé par Matt Leacock.

Matt Leacock, ce qui nous a permis de reprendre certains de ses mécanismes, tel que les cartes proposées ou les différentes actions proposées aux joueurs pour ne citer qu'eux.

"Le Ciel Interdit" est un jeu collaboratif dont le but est de faire décoller une fusée avant qu'un des joueurs ne meurt. Le principe consiste à construire le plateau du jeu en posant des tuiles, sur lesquelles il faut créer un chemin électrique permettant ainsi de générer un courant. Ce courant permet le décollage de la fusée. Cependant, les joueurs doivent faire face à des événements négatifs qui arrivent au fur et à mesure de l'avancement de la partie, il s'agit des cartes "orage". De plus, les joueurs possèdent deux caractéristiques qu'ils doivent gérer en permanence afin de ne pas mourir, ce sont les "points de vie" et les "points de corde". A chaque tour, chaque joueur possède quatre actions. Ces actions permettent aux joueurs de piocher une carte dans la pile des tuiles, d'explorer la carte en posant une tuile sur le plateau, de se déplacer sur le plateau et aussi de raccorder les tuiles pour créer le courant électrique. Au cours de la partie, les joueurs peuvent aussi récupérer des cartes "équipements" qui leur donnent la possibilité d'obtenir certains bonus. Ces bonus permettent principalement de contrer les effets des cartes "orage" mais aussi de leur donner des avantages. Les cartes "orage" sont jouées à chaque fin de tour d'un joueur (plusieurs cartes peuvent être jouées en même temps), leur but étant de ralentir la progression des joueurs, voire de les faire perdre la partie.

2.2 Buts et règles du jeu

Comme le jeu *"Le Ciel Interdit"*, *"Space Encoders"* se base également sur le principe du jeu de société totalement collaboratif, c'est-à-dire que les joueurs ne jouent pas les uns contre les autres mais qu'au contraire, ils jouent tous ensemble afin de gagner face au jeu. Ce mécanisme rend le jeu d'autant plus intéressant puisqu'il oblige les joueurs à s'entraider afin de gagner la partie, ce qui implique ainsi une prise de conscience qu'il n'est pas possible de gagner seul mais qu'au contraire, tout comme en informatique, un travail d'équipe est indispensable.

Le but du jeu de *"Space Encoders"* est le même que celui de *"Le Ciel Interdit"*, à savoir faire décoller une fusée. Cependant, la façon de faire décoller la fusée diffère. En effet, pour *"Space Encoders"* le principe est de remplir des objectifs. Ces derniers sont des algorithmes à construire avec des tuiles qui vont constituer le plateau du jeu qui est à construire tout au long de la partie (comme pour le jeu *"Le Ciel Interdit"*). De plus, les joueurs devront faire face à un compte à rebours, il s'agit de la *deadline*, et devront donc finir leur tâche avant que celle-ci n'atteigne zéro. Pour cela, il existe différents types de tuiles: (1) les tuiles sur lesquelles sont associées des instructions, c'est en les agençant de la bonne manière que l'on construit les algorithmes afin de remplir les objectifs; (2) les tuiles sans instruction "Empty", utiles uniquement pour les déplacements des joueurs. Ce n'est qu'une fois que tous les algorithmes sont correctement construits et qu'aucun des joueurs n'est mort jusque-là, que la fusée peut décoller.

Au début de la partie, le plateau n'est composé que d'une seule tuile - la tuile centrale - sur laquelle la fusée est située (*Rocket*). Tous les joueurs de la partie sont placés sur cette dernière.

Le jeu *"Space Encoders"* peut se jouer de deux à quatre joueurs. Le nombre d'objectifs à remplir dépend ainsi du nombre de joueurs dans la partie. Il faut créer (nombre de joueurs – 1) algorithmes pour remplir tous les objectifs. Les objectifs en question sont sélectionnés aléatoirement au début de la partie parmi un ensemble. Chaque joueur choisit ensuite cinq tuiles dans la pioche des tuiles afin de remplir sa main. En effet, chaque joueur ne peut posséder en même temps que cinq tuiles à la fois. La façon de piocher se déroule comme suit: le joueur sélectionne trois tuiles parmi celles présentes dans la pioche afin de les dévoiler. Il choisit ensuite l'une d'entre elles et l'ajoute à sa main. Les deux autres tuiles sont ensuite remises dans l'ensemble puis ce dernier est mélangé. Une fois que tous les joueurs ont rempli leur main initiale, c'est-à-dire après cinq piochages pour chacun, la partie peut débuter.

Parlons maintenant des objectifs qui sont, comme nous l'avons mentionné auparavant, les algorithmes à construire tout au long de la partie. Ils sont ainsi constructibles à l'aide des différentes tuiles présentes dans le jeu. Ces dernières représentent (1) des structures de contrôles tel que des "if", des "while" ou encore des "for"; (2) des allocations de valeur pour

des variables de la forme "☐ = valeur" où le carré représente la variable - les joueurs peuvent la nommer comme ils le souhaitent - à laquelle sera assignée la valeur; (3) des conditions avec le même principe de carrés et des tests tel que par exemple "☐ != valeur" et enfin (4) des primitives qui sont des fonctions déjà implémentées tel que "show(☐)". Les algorithmes à construire peuvent ainsi l'être de différentes manières puisque nous avons créé plusieurs variantes pour chacun. Ceci permet aux joueurs de faire comme bon leur semble.

2.3 Les différentes actions

A chaque tour, chaque joueur peut faire quatre actions parmi les actions citées ci-après. Les actions possibles sont les suivantes: (1) Piocher une tuile; (2) Se déplacer; (3) Poser une tuile; (4) Retirer une tuile; (5) Echanger une tuile; (6) Passer son tour.

L'action (1) permet au joueur de piocher une tuile dans la pioche du jeu de la même façon qu'en début de partie lors des cinq premiers piochages (décrit auparavant). Dans le cas où le joueur a déjà sa main pleine, c'est-à-dire qu'il possède cinq tuiles dans sa main, le fait d'utiliser cette action lui permettra de dévoiler trois tuiles de la pioche du jeu et d'en échanger une avec l'une des siennes si cela lui convient. Dans le cas contraire il peut tout simplement décider de ne pas procéder à l'échange mais l'action sera quand même décomptée.

L'action (2) permet au joueur de se déplacer sur les différentes tuiles posées. Cette action ne peut s'effectuer que sur une tuile adjacente à la position actuelle du joueur.

L'action (3) est le fait de poser une tuile sur le plateau. Le joueur peut poser une tuile uniquement sur un emplacement adjacent à sa position et, bien évidemment, sur aucune tuile déjà placée. De plus, le fait de poser une tuile sur le plateau doit satisfaire certaines conditions. En effet, lorsque le joueur veut poser une tuile autre que "Empty" à la suite d'un algorithme en partie construit, il doit s'assurer que celle-ci respecte la sémantique ainsi que la syntaxe de l'algorithme. En ce qui concerne les tuiles vide "Empty", elles peuvent être posées à n'importe quel endroit, cependant si elles bloquent la pose d'une tuile pour la construction d'un algorithme, elles devront alors être enlevées.

L'action (4) est justement le principe de retirer une tuile posée sur le plateau. Le joueur ne peut retirer une tuile que sur une case adjacente à sa position et seulement si aucun autre joueur n'est présent sur la case qu'il veut enlever. De plus, il est impossible de retirer la case centrale (*Rocket*).

L'action (5) permet à un joueur d'échanger une des tuiles qu'il possède (sa main ne doit pas être vide) avec l'une présente dans la main d'un autre joueur. Les joueurs doivent ainsi se mettre d'accord afin d'optimiser la rentabilité de l'action.

Enfin, l'action (6) permet simplement à un joueur de passer son tour. Il peut l'utiliser à n'importe quel moment de son tour.

2.4 Les cartes "orage"

A chaque fin de tour d'un joueur, un certain nombre de cartes "orage" sont exécutées. Le nombre de cartes exécutées dépend de la jauge orage. L'initialisation de cette jauge est relative au nombre de joueurs présents dans la partie, elle vaut au début (nombre de joueurs - 1). La jauge est ainsi divisée en différents paliers qui représentent le nombre de cartes à être exécutées à chaque fin de tour complet (par exemple au palier 3, après que tous les joueurs aient joué leur tour, trois cartes seront exécutées). Les paliers sont les suivants: [1,2] = palier 1, [3,7] = palier 2, [8,12] = palier 3, [13,16] = palier 4, [17,19] = palier 5 (les valeurs de la jauge appartiennent à [1,20]). Une fois la jauge orage arrivée à hauteur de 20, la partie est perdue pour les joueurs.

Revenons aux cartes "orage". Celles-ci sont situées dans une pile où tout au long de la partie les cinq premières sont dévoilées. Les joueurs peuvent donc anticiper les 5 prochains malus qui leur seront appliqués. Il existe différents types de carte "orage": (1) L'éclair; (2) La bourrasque; (3) Le vent tourne (sens horaire ou sens antihoraire); (4) L'orage se déchaine; (5) L'orage se déchaine et mélange de la pile des cartes "orage".

L'éclair (1) est un malus qui réduit la *deadline* (définie auparavant) d'un point. Dans la pile orage, ces cartes sont celles qui sont le moins présentes puisqu'il s'agit du malus le plus contraignant car les joueurs n'ont aucun moyen de le contrer.

La bourrasque (2) a pour effet de décaler tous les joueurs d'une case. Ce décalage dépend de la direction du vent qui est donnée par la girouette. Au début de la partie, cette dernière indique l'Est, mais au cours du jeu le sens peut changer. Lorsqu'une bourrasque va prendre effet, les joueurs doivent donc faire attention à leur position car s'ils tombent dans un vide, c'est-à-dire si la bourrasque les pousse sur une case où il n'y a pas de tuile, ils perdront un point sur la *deadline*. Cette perte de points s'applique à chaque joueur mais aussi à chaque bourrasque. Par exemple, si un joueur est au bord d'un vide et que deux bourrasques vont être appliquées (la jauge orage est au palier 2 minimum), la *deadline* sera alors décrémentée de 2 points puisque le déroulement est le suivant: le joueur tombe dans le vide (-1 point), le joueur remonte sur la même case, le joueur retombe (-1 point) puis remonte à nouveau.

Il existe deux types de cartes Le vent tourne (3): dans le sens horaire et dans le sens antihoraire. Ces cartes impactent la direction du vent en faisant tourner la girouette citée plus haut ce qui implique ainsi un effet différent sur la bourrasque. Pour chaque carte, la rotation du vent est de -Pi/2 ou Pi/2 (selon sens horaire ou sens antihoraire).

L'orage se déchaine (4) est un malus qui fait augmenter la jauge orage de 1. Comme expliqué précédemment, plus la jauge est haute, plus le nombre de cartes "orage" à être jouées à chaque fin de tour de jeu complet est élevé.

Enfin, le dernier type de cartes "orage" est L'orage se déchaine et mélange de la pile (5). On retrouve les mêmes effets qu'avec la carte (4) (L'orage se déchaine) mais avec en plus la particularité que la pile des cartes "orage" est mélangée. Cette carte implique donc des imprévus dans le déroulement de la partie.

De plus, à chaque fin de tour complet de jeu (tous les joueurs de la partie ont joué une fois), la *deadline* est décrémentée de 1. Cet effet s'applique tout au long de la partie et ne peut être contré.

2.5 Les principales évolutions du jeu

Dans cette partie, nous allons détailler les principales évolutions du jeu depuis le commencement de sa création.

Tout d'abord, nous pensions récupérer l'idée des points de vie associés à chaque joueur. Nous avons cependant jugé intéressant de mettre en place une jauge de vie globale (*deadline*) à tous les joueurs ce qui renforce l'aspect collaboratif du jeu.

A propos du placement sur le plateau en fin de partie, nous aurions voulu garder le principe que tous les joueurs doivent être situés sur la case de la fusée pour ainsi gagner la partie. Cependant, après plusieurs tests nous avons dû renoncer à cela car cette règle rendait la fin de partie trop compliquée.

Au départ, nous avions aussi mis en place une action qui permettait de verrouiller une tuile posée. En effet, à ce moment la bourrasque - une carte "orage" - ne déplaçait pas les joueurs mais avait pour effet de souffler l'une des tuiles du plateau qui n'était pas verrouillée. Cependant, en testant le *gameplay*, cela n'était pas très intéressant; nous l'avons donc supprimée et par la suite avons modifié l'effet de la bourrasque.

En effet, la bourrasque a ensuite subi certaines modifications puisqu'elle n'avait plus aucun effet. Nous avons donc décidé de reprendre le concept de base du jeu *"Le Ciel Interdit"* en réutilisant le principe de faire déplacer les joueurs d'une case selon la direction du vent. Nous avons également remis en place la girouette pour ce faire.

En ce qui concerne les cartes "équipements", nous avons pu constater que ces dernières ne représentaient pas un bonus assez intéressant. Nous avons donc décidé de les supprimer totalement.

En testant le jeu, nous avons de plus constaté que le temps de jeu d'une partie était trop long. Nous avons donc décidé de mettre en place des modifications principalement sur les actions proposées aux joueurs ainsi que sur les cartes "orage".

Nous avons ainsi modifié l'action qui permet de choisir une tuile en apportant le principe que si un joueur possède le nombre de cartes maximal autorisé dans sa main, il a alors la possibilité d'en échanger une avec l'une d'un tirage (expliqué dans la partie 2.3). En plus de cela, nous avons augmenté ce nombre de trois à cinq. Nous avons aussi ajouté une nouvelle action qui permet un échange d'une tuile entre deux joueurs. Ceci évite aux joueurs d'être bloqués dans certaines situations et, lors des tests suivants, nous avons ainsi pu observer une fluidification du jeu.

Nous avons au cours du développement mis en place plusieurs versions constructibles pour un même algorithme. En effet, cela était nécessaire puisque lors de certains tests les joueurs étaient bloqués: ils n'avaient pas les tuiles nécessaires étant donné qu'ils avaient décidé de construire l'algorithme d'une façon que nous n'avions pas prévue. Ainsi, l'augmentation du nombre de tuiles a permis de réduire considérablement les blocages.

Une des dernières modifications majeures est liée à l'exécution des cartes de la jauge orage. Cette dernière se faisait à la fin de chaque tour de jeu d'un joueur. Or, ce principe est trop punitif lorsque le nombre de joueurs dans la partie augmente (pour trois et quatre joueurs) et que la jauge orage commence à atteindre des valeurs assez élevées (palier 3 et plus). En effet, le fait de prévoir son placement pour contrer les bourrasques par exemple n'est quasiment plus possible. Nous avons donc appliqué une modification en changeant le moment d'exécution de ces cartes qui se fait maintenant à chaque fin de tour de jeu complet.

Toutes les modifications qui ont été mises en place ont permis d'obtenir une version du jeu plus jouable et ainsi d'améliorer l'expérience de jeu des joueurs.

2.6 L'aspect *Serious Game*

Le jeu *"Space Encoders"* peut être considéré comme un *Serious Game* étant donné que son but principal est de faire apprendre des principes liés à l'informatique. En effet, en jouant au jeu, une personne novice pourra comprendre comment construire un algorithme tout en respectant aussi bien la syntaxe que la sémantique de ce dernier. Elle comprendra non seulement l'importance de travailler en équipe afin de résoudre des problèmes de plus grande taille, mais aussi de résoudre ces derniers dans un temps limite. De plus, le principe de la planification a été mis en place dans le jeu. En effet, les joueurs doivent anticiper leurs actions notamment en prenant en compte les cartes "orage" afin de pouvoir gagner la partie. Enfin, le fait de créer une version informatisée du jeu répond bien au concept de *game-based learning* puisque nous pouvons faire apprendre les éléments cités ci-dessus à l'aide d'un jeu vidéo.

3. Le développement informatisé de *"Space Encoders"*

3.1 Création du projet

L'idée de la partie informatisée de *"Space Encoders"* est de créer une application en ligne de commandes utilisables dans un terminal avec un graphisme relativement simple. Le principe d'une application *"command line tool"* est que pour chaque interaction entre le joueur et la machine, il est proposé au joueur différents choix à sélectionner. Ce concept est très bien adapté à notre jeu puisque son déroulement se base totalement sur les choix faits par les joueurs. Les participants sont ainsi focalisés sur le jeu en lui-même ce qui leur permet d'intégrer au mieux les idées et l'enseignement qu'il apporte. Ainsi, nous lions ici les concepts de *Serious Game* et de jeu vidéo ce qui nous rapproche de celui de *game-based learning*.

Le langage choisi pour coder le jeu est Swift et pour le développement, l'environnement de développement Xcode a été utilisé. Le projet Xcode créé a donc été du type *"command line tool"*.

3.2 Organisation et étapes de développement du code

Dans un premier temps, après avoir discuté de l'orientation que nous voulions faire prendre au jeu, il a été possible d'entrevoir les différentes étapes du développement informatisé. Les questions essentielles étaient les suivantes: comment le jeu pourrait se dérouler en une application "ligne de commande" ? Comment faire pour afficher le plateau à l'écran ainsi que le mettre à jour selon l'avancement de la partie ? Mais aussi comment les joueurs pourraient sélectionner les actions voulues ?

Ce questionnement a ainsi permis de penser aux différentes classes à implémenter, aux constantes et aux variables à mettre en place ainsi qu'à leur initialisation.

Dans un premier temps, l'affichage du début de partie a été créé. Ce dernier nous présente le titre du jeu et demande le nombre et le nom des joueurs.

```
WELCOME TO SPACE ENCODERS

ENTER THE NUMBER OF PLAYER (2-4): 3
Name of player 1: Jean
Name of player 2: Pierre
Name of player 3: Michel
```

Fig. 1 - Entrée dans le jeu *"Space Encoders"*.

Un affichage présente ensuite les buts du jeu ainsi que les règles. Il permet aux joueurs de prendre connaissance des différentes actions qu'ils peuvent exécuter au cours de la partie ainsi que les effets des cartes présentes dans le jeu. Cet affichage a été dupliqué dans un fichier texte auxiliaire au jeu ce qui permet ainsi aux joueurs d'y accéder à tout moment lors de la partie [CTRL + tab]. Il en est de même pour les aides concernant les algorithmes à construire qui sont également disponibles dans un autre fichier.

```
The effect of action are the followings:
[Draw a tile] : choose a tile from the draw stack
[Move on the map] : move on tiles
[Explore the map] : place a tile on a case adjacent to its position
[Remove a tile] : remove a tile on a case adjacent to its position
[Swap a tile] : swap a tile between 2 players
[Skip turn] : simply skip your turn

The effect of storm cards are the followings:
[Wind Turn clockwise] : change the wind direction clockwise
[Wind turn counter cw] : change the wind direction counter clockwise
[Storm] : remove 1 HP of deadline
[Squall] : push all player (1 case) in the wind direction
[Storm rages on] : increase by 1 the storm gauge
[Storm rages on & shuffle] : same + remix all storm cards

You can find the whole goals and rules in the file [goalsNrules]

You can also find the algorithms in the file [objectives]


Press [ENTER] to begin the game !
```

Fig. 2 - Présentation des actions et des cartes.

Afin de gérer les joueurs ainsi que leur ensemble de tuiles, la classe Player a été mise en place. Cette classe possède les attributs suivants: [nom, cartes, position, id]. L'attribut cartes

permet de renseigner les tuiles présentes dans la main du joueur en question, l'attribut position renseigne sa position sur le plateau et l'id, unique à chaque joueur, permet de les identifier.

En parlant du plateau, la classe Board permet de gérer son affichage ainsi que ses mises à jour. Le plateau est dans un premier temps initialisé avec une taille de sept par neuf et reste fixe tout au long de la partie. Il s'agit d'une des seules différences avec la version physique car ici il est important de préserver l'affichage et de faire attention aux retours à la ligne.

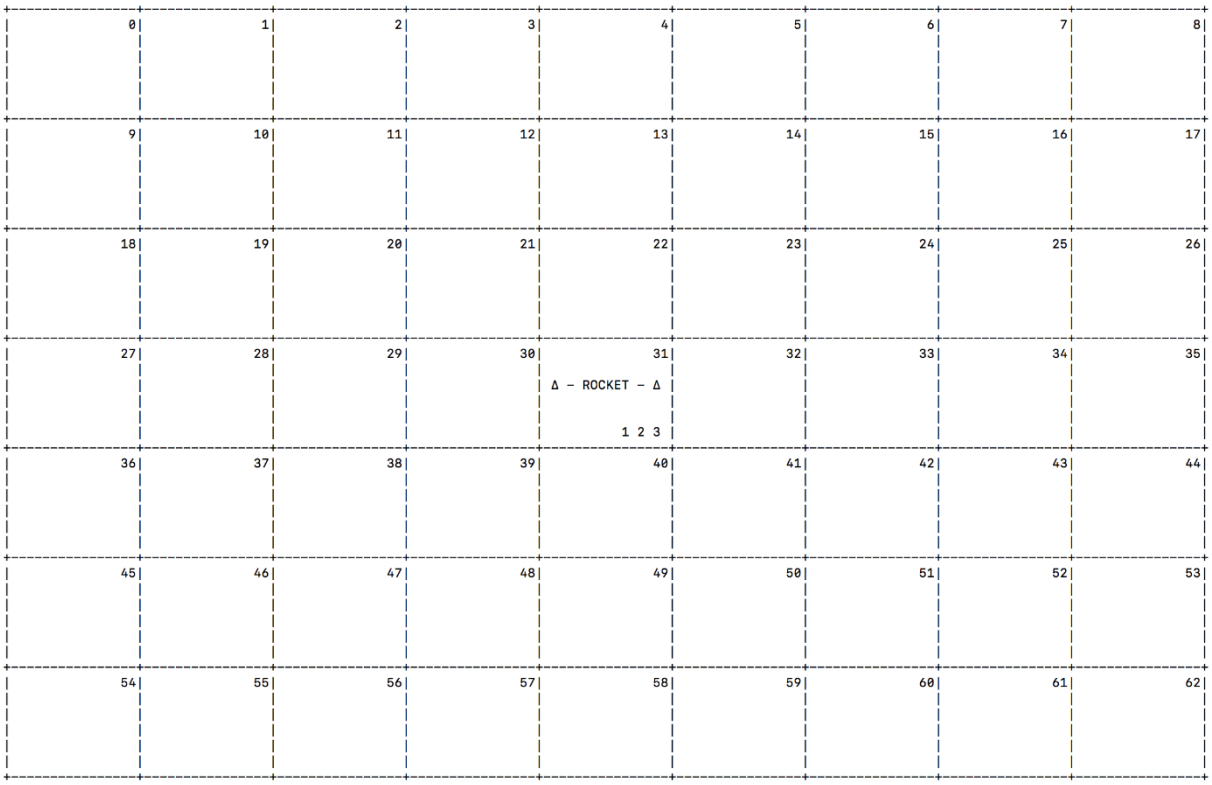


Fig. 3 - Plateau initial du jeu *"Space Encoders"*.

Le principe général du plateau est simple. Chaque case est identifiable avec un id unique affiché en haut à droite. Au centre d'une case, on y représente les instructions contenues sur les tuiles (la valeur est "Empty" dans le cas d'une tuile vide comme vu dans la partie 2.2). Dans le cas où rien n'est marqué dans une case, cela signifie qu'il y a un vide, c'est-à-dire

qu'aucune tuile n'a été encore posée à cette position. Dans la partie inférieure des cases, on y affiche les joueurs présents en utilisant leur attribut id.

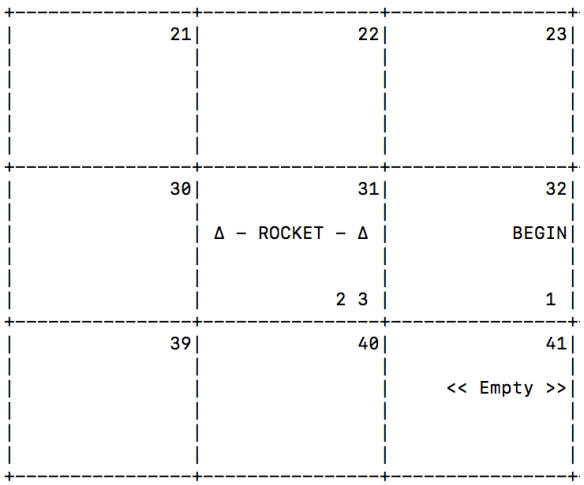


Fig. 4 - Différents types de tuile du jeu.

Le déroulement du jeu se fait à l'aide une boucle *while* et s'arrête lorsqu'une variable est passée à *true*. Cela peut se faire de deux manières différentes, soit par le fait que la *deadline* prenne la valeur de zéro, dans ce cas les joueurs perdent la partie, soit par la victoire des joueurs. Dans cette boucle, une variable indique le joueur qui doit faire son tour de jeu. A chaque action, un affichage des cartes orage dévoilées, de la direction du vent, de la jauge orage, du nombre d'actions restant pour le tour du joueur, des cartes possédées par tous les joueurs de la partie ainsi que du menu du choix de l'action est présenté.

```
TURN OF PLAYER 1 : Jean

The next storm cards are : → [Squall] → [Storm] → [Wind turn counter cw] → [Storm rages on] → [Squall] → ...
The wind direction is:  →
The storm gauge:

1    2        3        4      5    X
---↑

ACTIONS REMINING FOR PLAYER 1 (Jean): 4
Your set of tiles:  ["FOR", "<< Empty >>", "ENDWHILE", "ENDIF", "[VAR] = FALSE"]
Pierre's set of tile:  ["SHOW([VAR])", "<< Empty >>", "<< Empty >>", "<< Empty >>", "[VAR] = 10"]
Michel's set of tile:  ["END", "WHILE", "BEGIN", "[VAR] = [VAR]-1", "IF"]
```

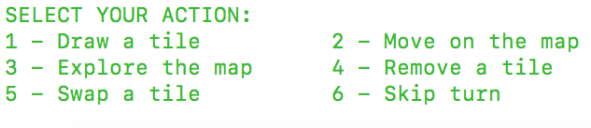


Fig. 5 - Exemple d'affichage des éléments pour chaque action.

Après l'action effectuée, le plateau est réaffiché afin d'y présenter les modifications. Une fois que tous les joueurs ont joué leur tour de jeu, le nombre de cartes orage qui doivent être exécutées est récupéré, la jauge orage est ainsi mise à jour et les actions des cartes orage (exemple: bourrasque, éclair, …) sont exécutées. Le plateau est ensuite à nouveau affiché pour voir les modifications apportées au jeu.

La classe Action regroupe les différentes fonctions liées aux actions choisies par les joueurs. Les actions possibles sont montrées sur la figure 5.

Pour l'action (1), un menu de sélection apparait afin que le joueur choisisse la tuile de son choix parmi les tuiles dévoilées. Une fois cela réalisé, la tuile sélectionnée est ajoutée à la main du joueur.

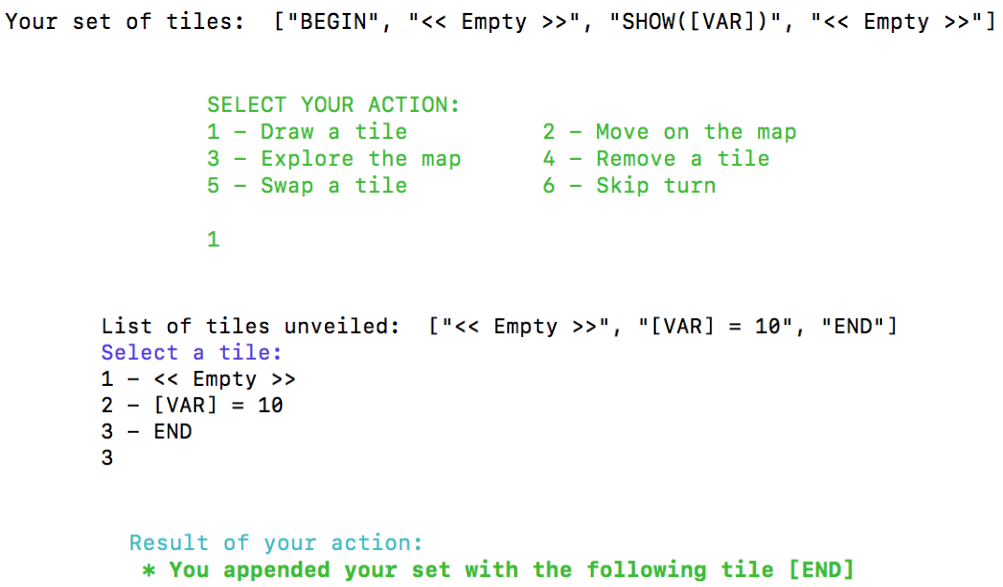


Fig. 6 - Le joueur a quatre tuiles dans sa main, il choisit l'action (1)
et choisit d'ajouter la tuile [END] à son jeu. Un log est ensuite affiché.

Il existe une alternative lorsque le joueur possède déjà cinq tuiles dans sa main. Il a le choix de sélectionner l'une des trois tuiles dévoilées ou alors, s'il n'est pas satisfait du tirage, il peut choisir l'option *Skip* afin d'abandonner l'action. Dans le cas où il veut procéder à l'échange, il doit choisir l'une des tuiles de sa main afin de la remplacer.

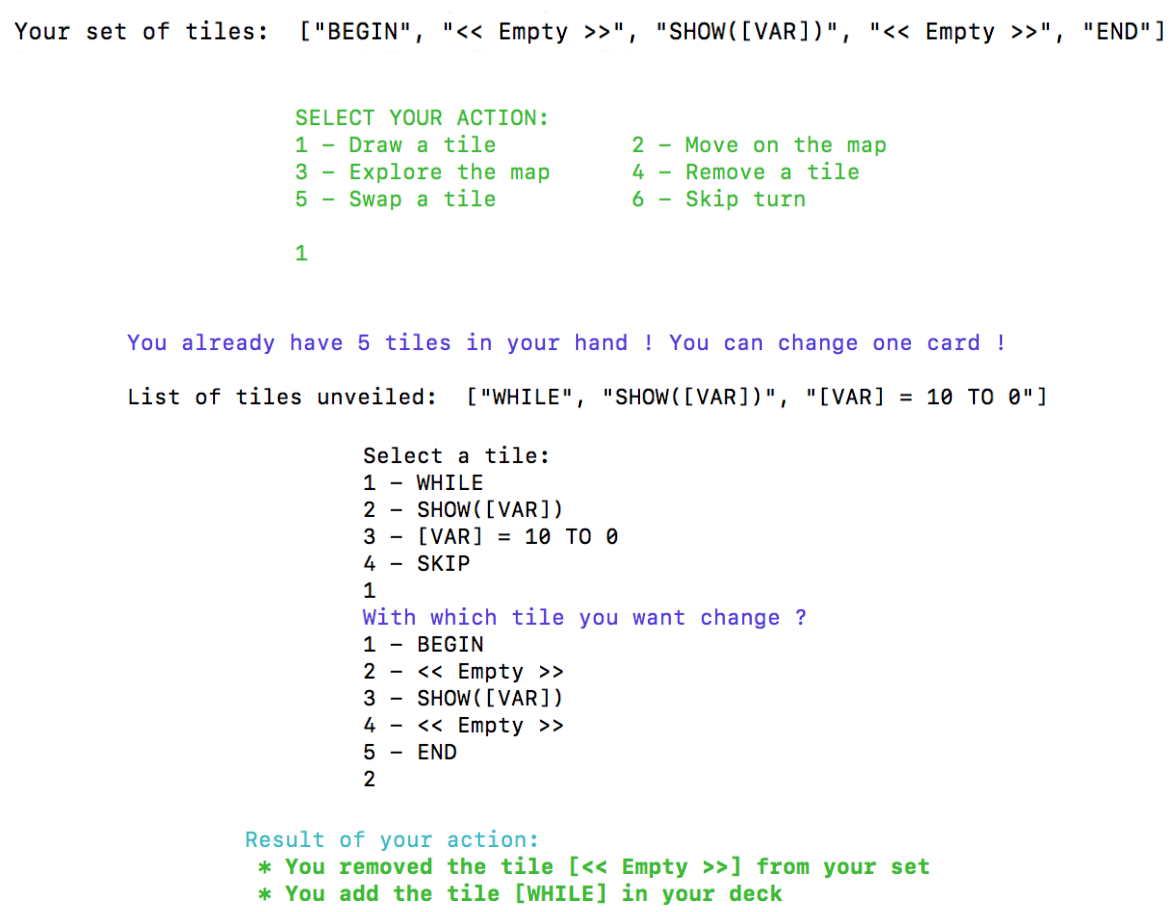


Fig. 7 - Le joueur a cinq tuiles dans sa main, il choisit l'action (1) puis choisit de remplacer sa tuile [« Empty »] par la tuile [WHILE]. Un log est ensuite affiché.

Pour l'action (2), un menu est proposé au joueur et lui demande de choisir la case où il veut aller parmi celles qui sont possibles. Ces dernières sont calculées selon la position actuelle du joueur et les potentielles tuiles adjacentes. Une fois le choix effectué, la position du joueur ainsi que l'affichage sur le plateau sont modifiés. Dans le cas où aucune tuile n'est posée autour de lui, l'action est annulée.

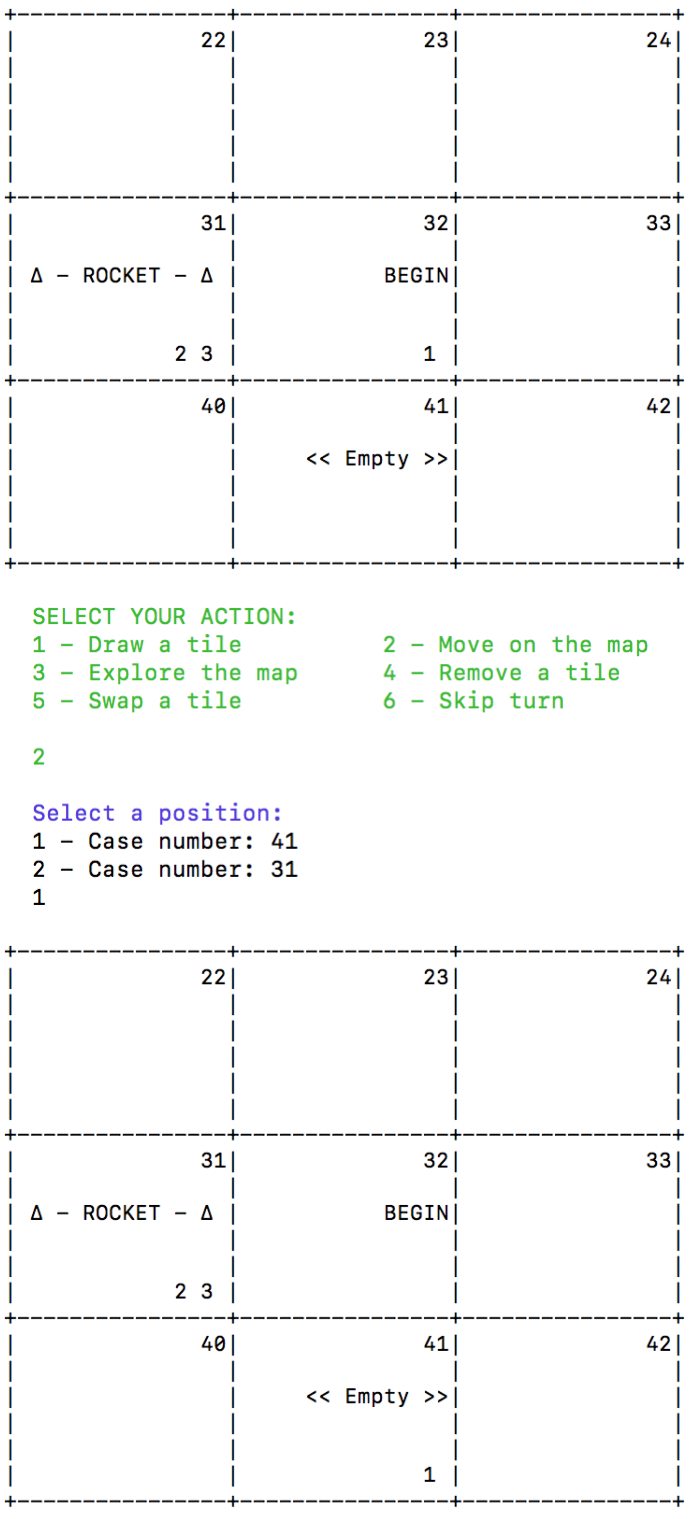


Fig. 8 - Le joueur 1 est sur la case 32, il choisit l'action (2) et sélectionne la position 41. L'affichage est ensuite modifié et un log est affiché.

L'action (3) permet de poser une tuile sur le plateau. Tout d'abord, les tuiles que possède le joueur ainsi que sa position actuelle sont affichées. Un menu demande par la suite au joueur quelle tuile il souhaite poser parmi celles de sa main, et à quel endroit parmi les positions possibles (calculées selon les tuiles déjà posées).

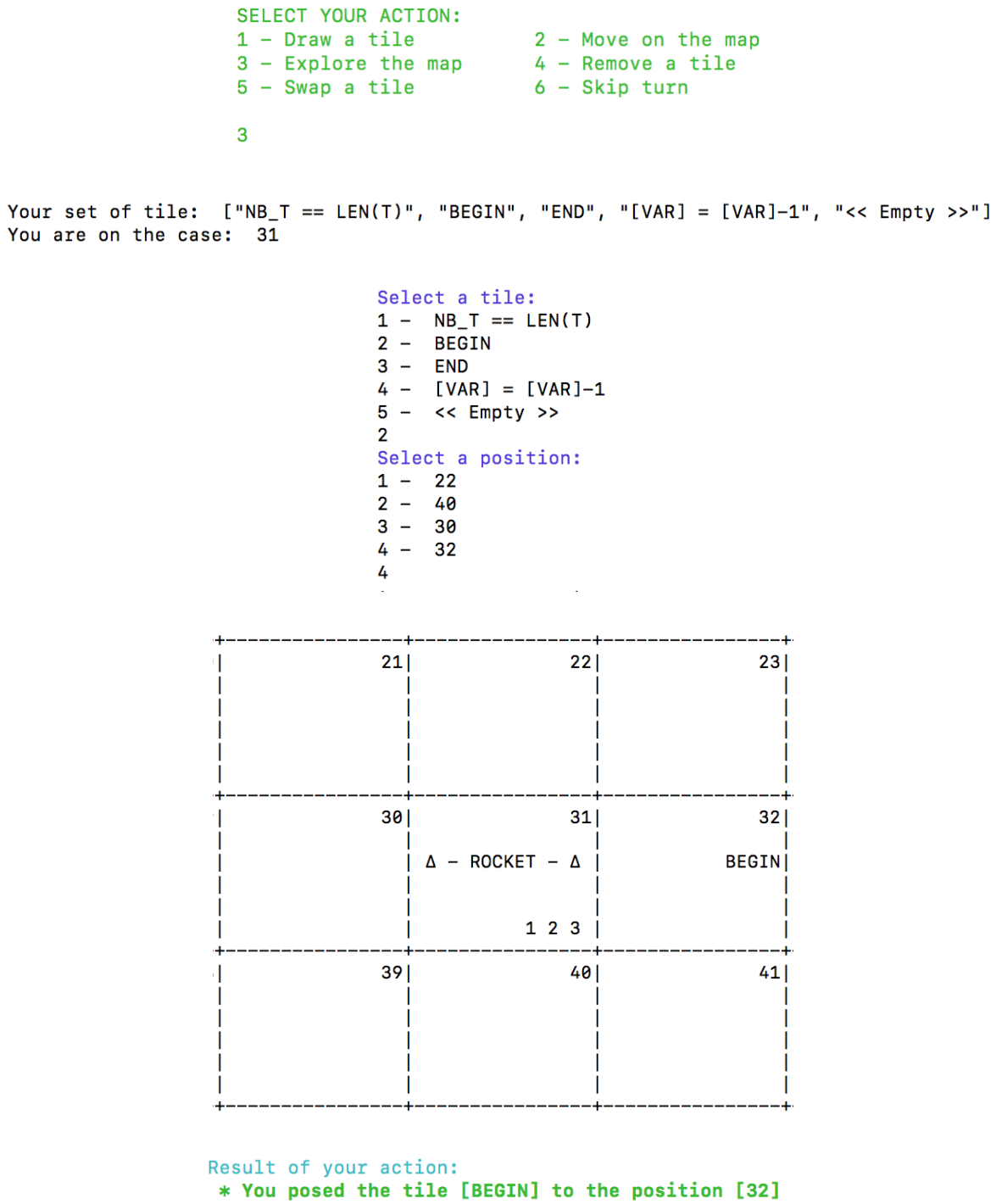
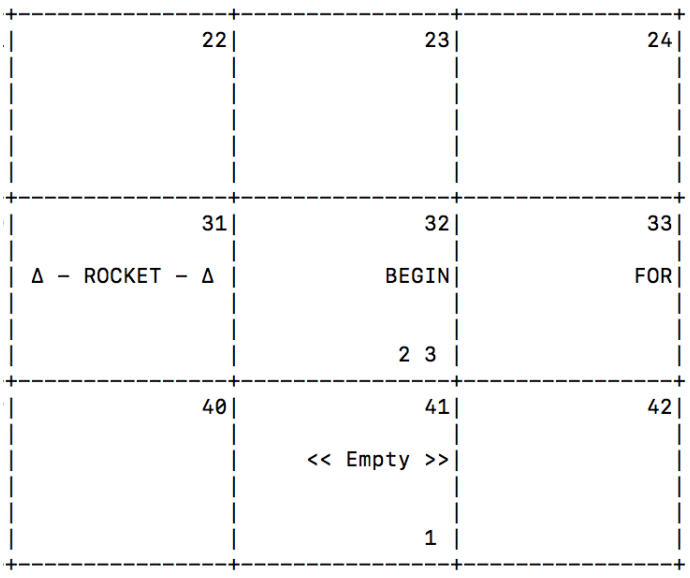


Fig. 9 - Le joueur 1 est sur la case 31, il choisit l'action (3), puis sélectionne sa tuile [BEGIN] et la position 32 afin de la poser. Un log est ensuite affiché.

Ensuite, la syntaxe et la sémantique de l'algorithme devraient être testées en faisant appel à la classe Interpretor. En effet, celui-ci permettrait ainsi de vérifier si la tuile posée coïncide avec l'algorithme en cours de construction à cet endroit. Cependant, la partie de l'interpréteur n'a pas été implémentée.

L'action (4) permet de retirer une tuile présente sur le plateau. Un menu est présenté pour sélectionner la case associée à la tuile à enlever. Les cases proposées dans ce menu doivent satisfaire toutes les conditions présentées dans la partie 2.3. Une fois la case sélectionnée, l'affichage est modifié en supprimant l'instruction qui lui est associée.




```
SELECT YOUR ACTION:
1 – Draw a tile          2 – Move on the map
3 – Explore the map      4 – Remove a tile
5 – Swap a tile          6 – Skip turn

4

Select a position to remove the tile on:
1 –  33
1
```

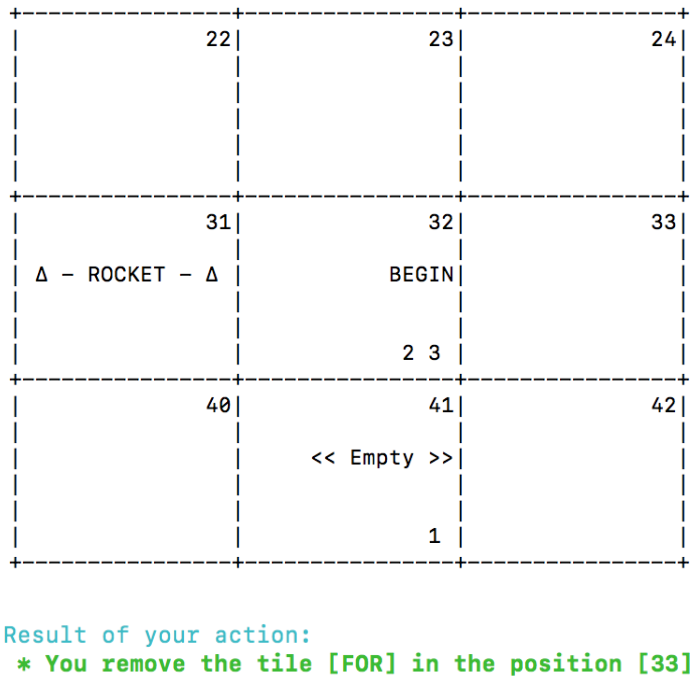


Fig. 10 - Le joueur 2 est sur la case 32, il choisit l'action (4)
et choisit de retirer la tuile de la position 33. Un log est ensuite affiché.

L'action (5) propose de faire un échange de tuiles entre deux joueurs. Il est dans un premier temps indiqué avec quel joueur l'échange peut se faire. Puis, un menu est affiché afin de sélectionner quelles sont les tuiles à échanger.

```
SELECT YOUR ACTION:
1 – Draw a tile          2 – Move on the map
3 – Explore the map      4 – Remove a tile
5 – Swap a tile          6 – Skip turn

5
```

```
The set of tiles of the others players (no empty):
Select a player to swap a tile:
1 (Jean) –  ["NB_T == LEN(T)", "END", "[VAR] = [VAR]-1"]
2 (Pierre) –  ["BEGIN", "SHOW([VAR])", "<< Empty >>", "END", "WHILE"]
1
```

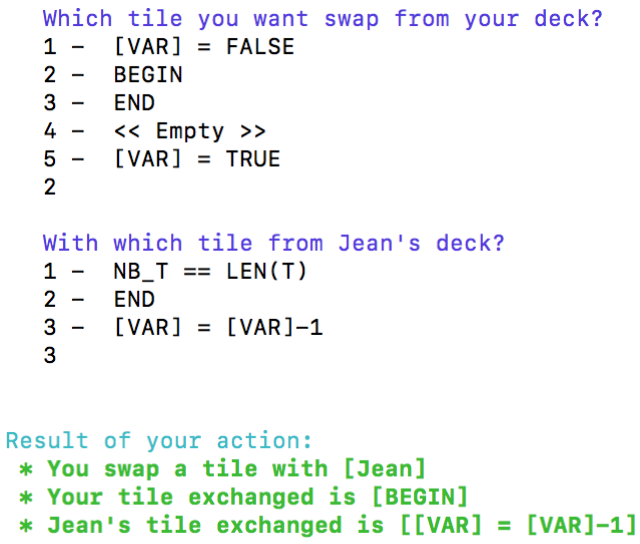


Fig. 11 - Le joueur possède cinq tuiles dans sa main, il choisit l'action (5) et choisit de faire un échange avec Jean (1), ils interchangent ainsi les tuiles sélectionnées. Un log est ensuite affiché.

La dernière action (6) permet de passer son tour. La partie continue ainsi avec le joueur suivant.

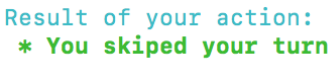


Fig. 12 - Le joueur choisit l'action (6). Un log est affiché.

Pour chacune des actions, quelques tests de robustesse ont été faits afin de limiter des erreurs d'entrées de la part des joueurs ou de contrer les joueurs mal intentionnés. Cependant, ces tests peuvent être grandement améliorés et augmentés.

Enfin, pour éviter de décompter des actions qui ont été mal sélectionnées (erreur de clic ou inattention de la part des joueurs), un *flag* permet de savoir si l'action sélectionnée est "valide", c'est-à-dire que celle-ci va modifier un élément du jeu. Par exemple, si un joueur veut se déplacer sur le plateau avec l'action (2) mais qu'autour de lui aucune tuile n'est posée, il reviendra alors au menu de la sélection des actions sans pour autant avoir dépensé un point d'action puisque le *flag* aura été mis à *false*.

La classe Card possède les attributs [stormStack, jaugeStorm]. Le premier attribut définit la pile des cartes "orage", le second indique la valeur de la jauge orage. A la fin d'un tour de jeu complet, les cartes "orage" qui ont été récupérées (selon la jauge) sont exécutées.

Les cartes *Storm* (1) représentent les cartes Eclair dans la version physique du jeu. Elles décrémentent de 1 la variable de la *deadline*.

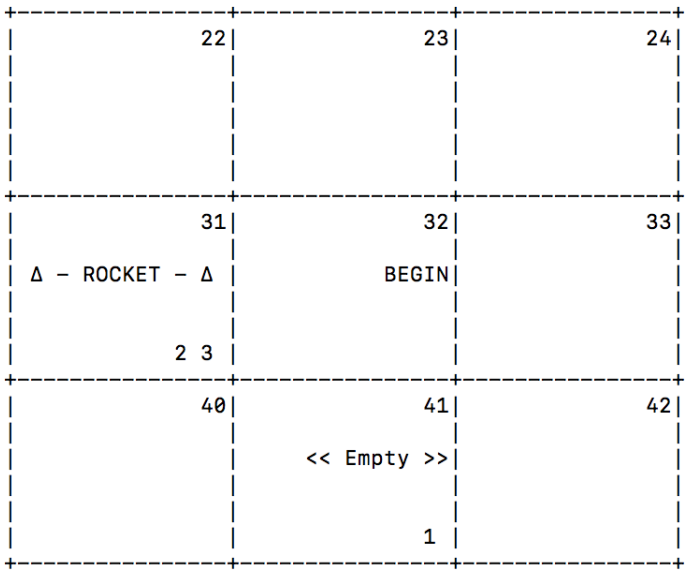
Les cartes *Squall* (2), qui sont les bourrasques pour la version physique, déplacent tous les joueurs d'une case selon la direction du vent. L'idée est ainsi de savoir pour chaque joueur s'il tombe dans un vide ou non. Dans le premier cas, une décrémentation de 1 de la *deadline* est appliquée mais la position n'est pas modifiée; par contre, dans le second cas, la *deadline* n'est pas décrémentée mais en contrepartie la position est changée.

Voici un exemple pour l'exécution d'une bourrasque:

```
TURN OF PLAYER 1 : Jean

The next storm cards are : → [Squall] → [Storm] → [Wind turn counter cw] → [Storm rages on] → [Squall] → ...
The wind direction is:  →
The storm gauge:

1    2        3        4      5    X
---↑
```



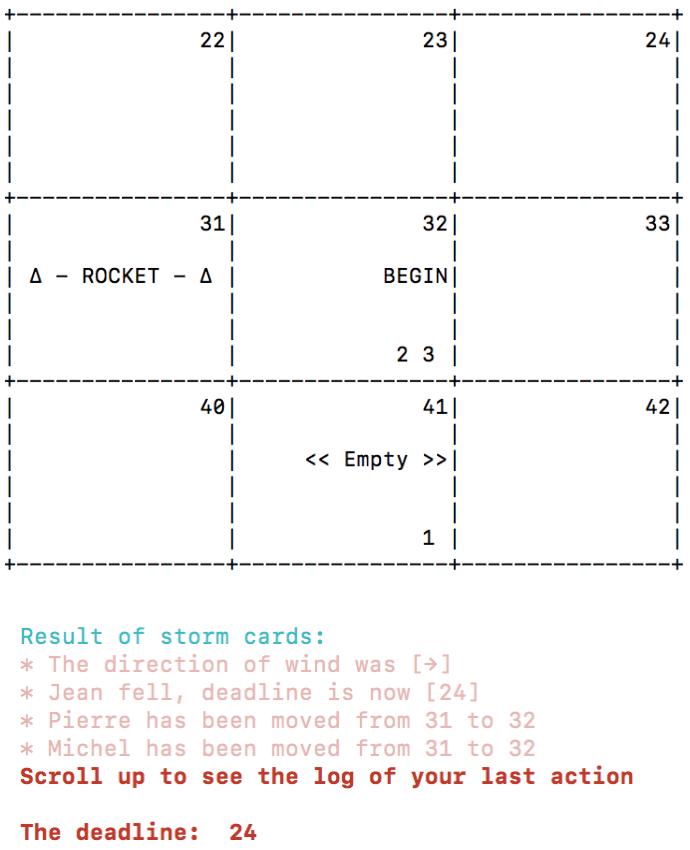


Fig. 13 - Le joueur 1 finit son tour de jeu, la bourrasque est la prochaine carte "orage" (jauge = 1), le vent va vers l'Est, tous les joueurs sont poussés d'une case. Un log est ensuite affiché.

La direction du vent est impactée par les cartes *Wind turn clockwise* et *Wind turn counter cw*. Il s'agit des cartes du type Le vent tourne pour la version physique. Ces dernières agissent sur la variable qui définit le sens de la flèche qui peut prendre les quatre valeurs suivantes: [↑ , → , ↓ , ←]. Elles permettent ainsi d'effectuer une rotation de -Pi/2 ou Pi/2 du sens de la flèche.

Les cartes *Storm rages on* (L'orage se déchaine) incrémentent simplement la valeur de *jaugeStorm* de 1 tant que celle-ci est inférieure à 20. Une fois arrivée à cette valeur, la partie est finie. On a le même principe pour les cartes *Storm rages on & shuffle* qui appliquent en plus un mélange de la pile *stormStack*. L'affichage de la jauge orage est ensuite modifié.

Fig. 14 - Exemple de la jauge orage.

La flèche indique la valeur de la jauge et donc le palier actuel, c'est-à-dire le nombre de cartes "orage" qui vont être exécutées la prochaine fois (les chiffres au-dessus). Le X signifie que la jauge a atteint sa valeur maximale qui est de 20.

4. Les difficultés et les limites du travail

Nous allons aborder dans cette partie les problèmes rencontrés tout au long du développement.

Au niveau du choix du jeu sur lequel nous nous sommes basés, nous pouvons dire que nous n'avons pas rencontré de problèmes. En effet, nous avons eu de bons conseils lorsque nous nous sommes renseignés, ce qui nous a tout de suite conduits dans la bonne direction. Cependant, l'une des grandes difficultés a notamment été de récupérer les concepts du jeu de base, de les réadapter selon nos besoins et d'en créer de nouveaux. Nous avions de nombreuses idées, mais cela dit, certaines d'entre elles n'ont pas abouti comme décrit dans la partie 2.5.

Une autre difficulté majeure a été de garder un jeu jouable. En effet, celui-ci ne devait être ni trop facile et ainsi que certaines stratégies *"auto-win"* puissent être mises en place, ni trop compliqué pour éviter que ce soit impossible de gagner. Pour les objectifs, il a donc fallu créer des algorithmes assez simples afin de correspondre au niveau des joueurs (novices en informatique) mais pas trop faciles non plus afin de garder un intérêt au jeu. Les différents paramètres du jeu tel que le nombre de points sur la *deadline*, le nombre d'actions par joueur, le nombre maximum de cartes dans la main des joueurs et bien d'autres ont été compliqués à mettre en place. Pour cela, il a fallu faire de nombreux tests, avec différents joueurs, afin de voir comment les parties évoluaient. Cependant, nous n'avons pas pu en faire autant que voulu puisque ceux-ci sont assez compliqués et longs à effectuer.

Au niveau de la partie informatisée, l'affichage du plateau a posé quelques problèmes. En effet, le principe de l'affichage se fait à l'aide de la fonction swift *print()*. Il a donc fallu trouver un moyen d'afficher les délimitations des cases ainsi que les éléments qui les composent tout en respectant l'affichage qui se fait séquentiellement ligne par ligne. Des fonctions de remplissages ont donc été mises en place afin de mettre à niveau les différentes valeurs des cases et ainsi d'obtenir l'affichage voulu.

En prenant en compte les actions des joueurs et les actions liées aux cartes "orage", on constate qu'il existe un grand nombre d'actions présentes dans le jeu. Ainsi, lors de la création et de la modification de l'une d'entre d'elles, il a fallu faire attention que celles-ci fassent dans un premier temps les opérations voulues, mais aussi qu'elles ne modifient en aucun cas, dans un second temps, le comportement des autres. Il a donc fallu faire énormément de tests en modifiant des paramètres tel que par exemple le nombre de joueurs présents dans la partie, le nombre de tuiles dans la main des joueurs, le palier de la jauge orage, etc…, afin de tester l'implémentation. Ceci représente une grande partie du temps de développement.

Nous allons maintenant aborder les limites du travail. Comme évoqué dans la partie 3.2, la création de l'interpréteur reste très compliquée, d'une part à cause de la difficulté que cela implique et d'autre part en raison du temps restant jusqu'à la fin du projet. Sa mise en place n'a donc malheureusement pas été un succès. Cependant, la grammaire du langage que nous avons créé en construisant les objectifs est en cours de développement et sera ainsi utilisée lors de la mise en place de l'interpréteur.

5. Les améliorations du jeu et les perspectives

Ce projet nous a ainsi permis de créer un jeu éducatif et de le développer sur différentes plateformes dans le but de faire apprendre des concepts fondamentaux de l'informatique. Toutefois, nous pourrions apporter certaines améliorations au jeu *"Space Encoders"* afin d'élargir le public auquel il est dédié. En effet, à la base du projet, il était question de créer un jeu éducatif pour des personnes débutantes dans le domaine de l'informatique. Il serait ainsi intéressant de pouvoir modifier les objectifs initiaux en d'autres objectifs plus complexes selon le niveau des joueurs. Nous pourrions par exemple demander aux joueurs de créer des

algorithmes de tri sans pour autant leur donner des informations (aides fournies dans les fichiers auxiliaires) ou alors, de construire des algorithmes représentants une suite de commandes *git* tel que par exemple, *git add*, *git commit*, *git push* dans le but de faire apprendre l'utilisation de Github. Certains joueurs pourraient ainsi apporter eux-mêmes des algorithmes et, de ce fait, étendre le jeu actuel.

En ce qui concerne la partie informatisée, l'implémentation de l'interpréteur est la dernière partie à finaliser pour rendre le jeu totalement fonctionnel. De plus, des améliorations dans l'affichage pourraient être mises en place ainsi que l'utilisation de librairies graphiques afin de rendre le jeu plus attractif et de capter davantage l'attention des joueurs. Il serait également intéressant de pouvoir utiliser cette version informatisée afin de simuler un grand nombre de partie et d'en étudier les résultats.

Toutes les informations à propos du projet sont disponibles sur notre *Github*[9]. Pour accéder au code de l'implémentation ainsi qu'au *README* qui explique l'installation du jeu, vous pouvez suivre ce lien[10].

[9] Disponible sur: <https://github.com/cui-unige/BSc-educative-gaming-2019>

[10] Disponible sur: <https://github.com/cui-unige/BSc-educative-gaming-2019/tree/devInf>

Je souhaite remercier Didier Buchs ainsi que Dimitri Racordon, Stefan Klikovits et Damien Morard pour leurs idées ainsi que leurs conseils tout au long de ce projet.

6. Bibliographie

Damien Djaouti, Julian Alvarez, Jean-Pierre Jessel, et Olivier Rampnoux. « Origins of serious games ». In *Serious games and edutainment applications*, 25–43. Springer, 2011.

Julian Alvarez « Du jeu video au Serious Game : Approche culturelle, pragmatique et formell*e* », 2007.

GameLearn, « Game-based Learning, ludification et Serious Games », [en ligne]. Disponible sur :<https://www.game-learn.com/ce-que-vous-devez-savoir-sur-serious-games-game-based-learning-exemples/>

DragonBox, « Dragon Box Eléments », [en ligne]. Disponible sur: <https://dragonbox.com/products/elements>

Pulse !!, « First Person Healthcare System Simulation », [en ligne]. Disponible sur: <https://www.gamasutra.com/view/news/102373/SGS_Feature_Pulse_First_Person_Healthcare_System_Simulation.php>

WikiHow, « Make your own Video Game », [en ligne]. Disponible sur: <https://www.wikihow.com/Make-Your-Own-Video-Game>

Cocktail Games, 2018 « Le Ciel Interdit », Réalisé par Matt Leacock.