Rapport Projet Informatique

Valentin DREANO
Antoine MAN
Quentin RAMSAMY- -AGEORGES
Ramzy ZEBRIR

Tuteur : Dimitri Watel

Sommaire

I. Introduction

L'objectif de cette seconde version de prim est :

- d'améliorer l'expérience de jeu;
- d'améliorer l'immersion;
- d'optimiser le jeu en rendant le code plus modulable;
- de permettre une expérience de jeu personnalisée.

II. Création des tâches

A. Tâche C.1 - Mode graphique

Cette tâche est probablement l'une des plus longues. Vous devez porter le code existant dans un autre langage pour permettre sa diffusion plus facilement. Le JS a été retenu car le code n'aura en grande partie pas besoin d'être réécrit.

B. Tâche C.2 - Application

Ajoutez les petits détails à la version graphique comme des menus ou de la musique pour permettre une meilleure immersion. Coder également la configuration qui permettra de compiler une application (.exe par exemple sous Windows) depuis le code.

C. Tâche C.3 - Tests

Transvaser les tests de la version C en JS.

D. Tâche C.4 - Translation et réglage du son de la musique

Vous devez permettre de jouer en plusieurs langues (anglais par défaut), les réglages de la langue peuvent se faire depuis un nouveau paramètres. Si vous ajoutez un menu paramètres, vous pouvez également coder le réglage du son de la musique ou encore un menu qui règle la musique jouée.

E. Tâche C.5 - Logger

Alternativement, debug l'interface est parfois très compliqué, on peut donc mettre en place un logger qui log l'état du jeu. Vous pouvez mettre en place une interface graphique pour analyser tour par tour les infos du logger. Prenez bien l'exemple les loggers comme sur internet, donc avec des niveaux de debug (warning, error, critical, info, par exemple). Ces méthodes permettront de ne pas afficher tous les logs et finir par en avoir trop. On peut choisir le niveau de Logs avec un argument au lancement. Les logs sont ensuite sauvegardés dans un fichier avec par exemple la log-.log comme nom (et ms la milliseconde à laquelle le programme a été lancé). En particulier, en Javascript, vous avez la console avec console.log, console.info, console.error, . . .

F. Tâche C.6 - Staffs

On trouve un peu bizarre de pouvoir acheter plusieurs fois le même Staff, ce serait plus marrant d'avoir un système de financement, donc qui provoque un effet d'amélioration de l'effet du professeur à chaque financement (équivalent à l'acheter plusieurs fois). On attend donc dans l'interface graphique, pour chaque staff pouvant être financé, un bouton ***Financer*** avec le coût (qui devra être croissant et non fixe) qui permet d'améliorer l'effet d'un Staff. Il serait pertinent de voir en direct l'effet de l'amélioration d'un professeur.

G. Tâche C.7 - Prim Master

On voudrait rendre le jeu un peu plus évident au joueur. Des icônes ou images ou couleurs pour l'orientation et les machines pourraient aider. Tester si c'est possible et faites en sorte que même dans la liste des machines nous voyons les icônes, . . . Vous pouvez également améliorer la légende en ajoutant les 3 commandes suprêmes qui sont m (voir la map), r (voir la map des ressources) et g (voir la map des déchets). On pourrait également améliorer le menu d'aide en ajoutant par exemple des exemples d'utilisation de certaines commandes, add_machine n'étant pas très évidente à cause de l'orientation. Enfin vous pouvez en profiter pour améliorer le menu des règles, pour y ajouter des détails sur ce que sont les actions. . .

Alternativement, c'est peut être un peu embêtant d'avoir m r et g pour les versions de la map, . . . On pourrait ajouter un système qui permet de cliquer sur une case, ce qui affiche toutes les informations comme :

- détails sur le type de machine (nom + id)
- détails sur l'orientation (ex: un texte qui dit ça rentre de bas et gauche et sort du côté opposé par exemple)
- détails sur le niveau : level (int)
- nombre de ressources/déchets sur la case/machine et cliquer n'importe ou ailleurs ferme le popup/popover.

H. Tâche C.8 - Documentation

Assurez-vous que les fichiers sont bien à la norme imposée au début du projet, les noms respectent nos normes et la documentation est claire est bien rendue en doxygen. Réalisez le diagramme de GanTT et le rapport.

I. Tâche C.9 - Fun tester

Vous devez essayer de faire des améliorations rendant le jeu plus amusant. Et d'empêcher des actions pouvant nuire au fun (acheter 500 fois un staff et finir avec 15 milliard de staffs sans acheter une seule machine ni joueur au jeu juste pour battre le meilleur score par exemple). Il serait par exemple intéressant de calculer une nouvelle fonction du score qui prend en compte plein de paramètres et détermine le score des actions du joueur. Il peut également être intéressant de mettre en place un système de succès. Les succès peuvent

être sauvegardés dans le stockage local. Afficher le temps de jeu et l'intégrer dans le calcul du score peut aussi être intéressant.

III. Répartition des tâches

Valentin DREANO : Tâche C.3 et C.5

Antoine MAN : Tâche C.7 et C.8

Quentin RAMSAMY- -AGEORGES : Tâche C.1 et C.2

Ramzy ZEBRIR : Tâche C.4

IV. Fonctionnalités mises en place

A. Tâche C.1 - Mode graphique

L'application est codée en typescript donc js, natif ou avec node.js, du html avec Boostrap et avec le module ***electron***. Electron n'est pas une invention révolutionnaire mais simplement un navigateur qui permet d'exécuter du JS. Ainsi, la console peut être ouverte avec CTRL-SHIFT-I par exemple, la page rechargée avec CTRL-R etc. C'est une raison qui fait qu'***electron*** est souvent détesté.

Le code moteur du jeu, donc toutes les classes qui constituent le code écrit en C, ont été codés en typescript. Il s'agit d'une version uniformisée du JS qui peut via une commande (npx tsc) être compilée dans n'importe quelle version de JS. Ceci nous permet d'ajouter au JS des types et de profiter d'une assistance syntaxique très soutenue.

Le code est majoritairement un copié collé du code c, modulo les changements au niveau de la déclaration des variables, et le fait que nous avons créé des classes plutôt que laisser des fonctions globales.

L'interface à étée rapidement refaite en html et js, avec du code déjà fait pour le terminal par exemple, donc qui appelle automatiquement la méthode associée à une action et Boostrap pour coder très rapidement l'interface.

B. Tâche C.2 - Application

Cette partie a été faite au début de la partie C.1, il s'agissait en faite de tests pour voir si electron était possiblement ce que nous allions utiliser. Tester la possibilité de faire des menus, et de pouvoir se déplacer entre des pages était primordial. Il s'agit en fait simplement d'un changement de fichier .html donc très finalement très simple.

La génération d'un .exe (version windows) ou d'une archive .tar.gz (version linux) se fait automatiquement avec le module electron-builder, simplement en spécifiant quelques options. Nous pourrions également faire une version macos/darwin mais cela nécessite un mac et un compte développeur apple.

Nous avons donc pu commencer à coder la partie C.1 avec electron. Le chargement de la musique se fait en 1 ligne, car javascript supporte la lecture audio donc il n'y avait pas vraiment de défi ici, juste appeler un constructeur et lancer la musique.

C. Tâche C.3 - Tests

Pour réécrire les tests de c à JavaScript, il a fallu trouver une technologie équivalente à CUnit. Pour cela, on a trouvé ***Mocha*** qui est une librairie permettant de compartimenter les test en suites.

Il a donc fallu adapter tous les tests afin de les rendre compatibles avec le nouveau code qui était légèrement différent du au changement de langage et au passage à l'objet. De plus, l'ajout de nouveau à pu être fait car certaines fonctionnalités étaient plus simples à gérer. Cependant des difficultées on été rencontré quand il a fallu gérer la partie de génération aléatoire d'une map qui ne pouvait pas prendre de seed pour faire le random. Il n'était donc pas possible de prévoir la map qui allait être générée.

D. Tâche C.4 - Translation et réglage du son de la musique

Je me suis occupé de coder une fonctionnalité permettant de pouvoir changer entre la version anglaise et française du jeu. Cela se fait à partir du menu Paramètres. Dans un premier temps il a fallu créer la vue Paramètres qui est conçue pour permettre à l'utilisateur de changer la langue du jeu à n'importe quel moment. Le changement de langue du jeu s'effectue lorsque nous quittons le menu de paramètres. Dans un second temps il était nécessaire de coder le côté événementiel de la page paramètres afin que le choix de langue de l'utilisateur soit bien sauvegardé. La plus grande difficulté de cette tâche était la traduction des différentes pages. Chaque page était codée de manière différente et nécessitait donc une approche différente afin de les traduire. C'était une tâche vraiment très longue car il fallait d'abord trouver tous les champs à traduire de chaque pages dans le code. Il y avait des champs où il fallait modifier le html, il fallait donc insérer une balise span avec un attribut id dont son contenu sera modifié par la suite par la traduction qui était stockée dans une Hashmap. Cependant il y avait également beaucoup de champs où il ne suffisait pas de remplacer le html. Il fallait intégrer des bouts de code non fixe dans les traductions, parfois il fallait directement dans la classe concernée traduire un attribut qui est affiché via un echo, les méthodes d'affichages des menus étaient différentes par conséquent il fallait à chaque fois trouver une nouvelle solution afin de traduire les différents menus. La difficulté résidait dans le fait qu'il fallait trouver chaque bout de code correspondant à l'affichage des différents champs des différents menus.

Nous avons décidé d'ajouter une fonctionnalité permettant à l'utilisateur de modifier le son du jeu. Cette fonctionnalité se trouve également dans le menu Paramètres, il a fallu tout d'abord s'occuper de sa conception en html, de son apparence en CSS pour ensuite coder tout le côté événementielle du menu ayant pour but de permettre à l'utilisateur de changer le volume du jeu. Pour ce faire, il fallait dans un premier temps créer une barre de son dont la valeur et l'apparence s'actualisaient lorsque nous la bougions. Il a fallu trouver un moyen de récupérer le paramètres audio du programme pour ensuite le modifier et stocker sa valeur afin que le volume de jeu de l'utilisateur ne se réinitialise pas lorsqu'il quitte le menu. Pour effectuer cette tâche il a donc fallu stocker les paramètres audio de l'utilisateur dans le

programme et faire en sorte qu'il puissent s'actualiser lorsque l'utilisateur souhaitait modifier le volume audio.

E. Tâche C.5 - Logger

La création du logger à été une partie simple à réaliser. Il permet d'écrire dans un fichier de log les différentes informations qui ont un niveau d'importance. Le niveau d'importance est défini au départ par le développeur qui a besoin de debug.

Il y a 4 niveaux : **debug, info, warning** et enfin **error.**

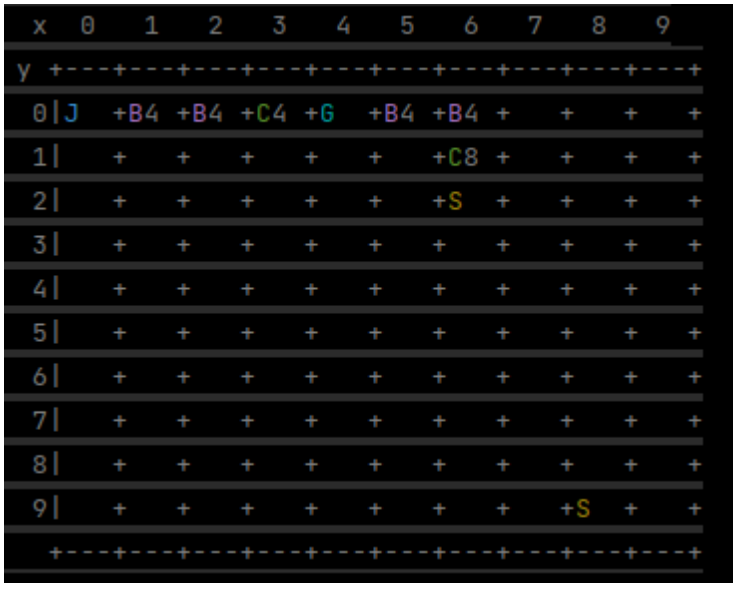
Le logger peut également être désactivé. De base, si le développeur ne change pas la valeur dans sa main le logger affichera seulement les erreur de type warning et error.

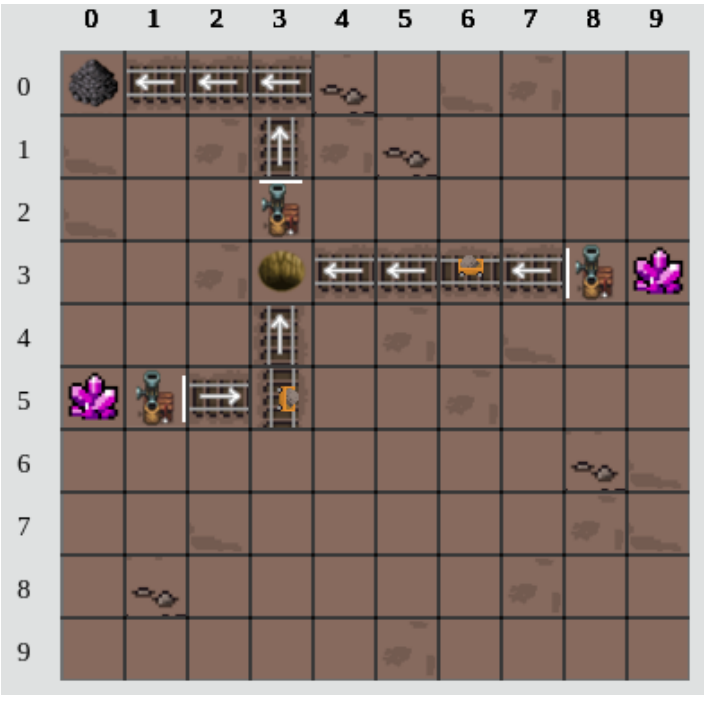
F. Tâche C.7 - Prim Master

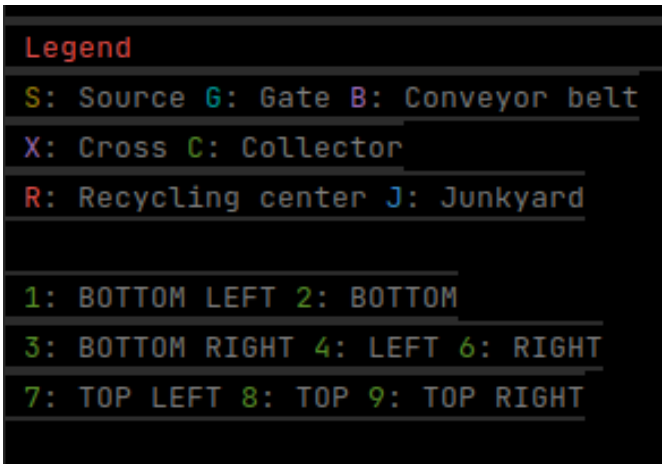
Afin de rendre le jeu beaucoup plus simple au premier coup d'œil, nous devions apporter une couche graphique au jeu. Pour cela, nous nous sommes orientés dans un premier temps vers l'affichage de caractère Unicode mais comme nous avions la possibilité d'afficher des images grâce au JS et bien nous sommes plutôt partis vers cette option. Mon travail s'est donc orienté vers la recherche d'assets de jeux vidéo (libre de droit), du photomontage et enfin la création du plateau de jeu avec ces images.

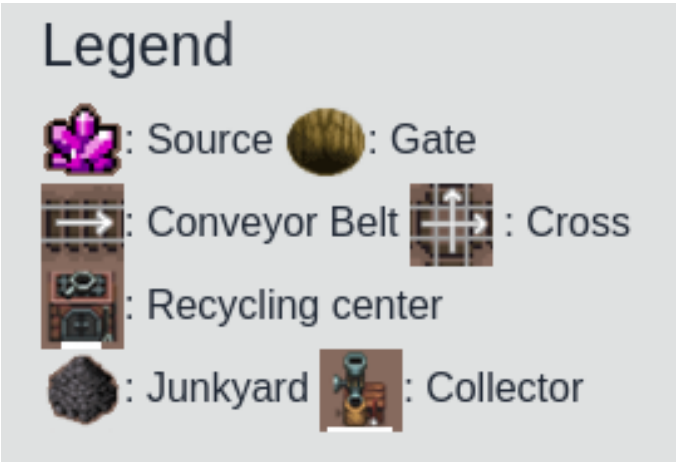
Ensuite, nous voulions donner la possibilité au joueur de suivre la progression du jeu à tout instant en lui permettant de cliquer directement sur les cases de la grille afin de connaître le nombre de ressources et de déchets ou encore de connaître le niveau d'une machine par exemple. Comme il s'agit d'une page internet, la récupération de la position de la souris a été plutôt simple. Il fallait ensuite ajouter des Events lorsque le joueur cliquait sur la grille qui enclenche derrière la récupération de toutes les infos de la case et l'affichage de ces infos à l'utilisateur.

Comparaisons :









G. Tâche C.8 - Documentation

Cette tâche consistait simplement à tenir à jour le diagramme de Gantt et à rédiger ce rapport.

V. Tuto installation du jeu

A. Sous Windows

https://lgs-games.com/assets/file/prim%20Setup%202.3.0.exe

Veuillez tout simplement installer le **.exe** et suivre les différentes étapes décrites lors de l'installation.

B. Sous Linux

Selon ce que vous préférez, vous avez pour linux :

- https://lgs-games.com/assets/file/prim-2.3.0.tar.gz

Pour cela, il suffit tout simplement de désarchiver l'archive et d'exécuter "***prim***" sous un terminal.

- https://lgs-games.com/assets/file/prim-2.3.0.AppImage

C. Sous Debian

https://lgs-games.com/assets/file/prim_2.3.0_amd64.deb