Université de Montpellier

Rapport pour le TER

Sokoban

Paola Andreu, Yanis Bled, Thiziri Harcheb, Aurélie Murcia,
Marine Pellicer, Julien Roussé

Supervisé par
Violaine Prince

11 mai 2017

Table des matières

Introduction

L'objet de ce rapport est de présenter synthétiquement le travail réalisé dans le cadre du TER. Le projet que nous avons choisi porte sur le jeu de Sokoban. Notre but était d'améliorer une version existante de ce jeu [1].

Le principe du Sokoban est de diriger un personnage dans un labyrinthe. Ce personnage doit pousser les caisses jusqu'aux objectifs indiqués sur la carte. La partie est gagnée lorsque le joueur a placé toutes les caisses sur leurs cibles.
Pour commencer, afin d'identifier les points positifs, négatifs et les limites de la version de base, il nous a fallu faire preuve d'esprit critique. Ensuite, à partir de l'ensemble des améliorations proposées au sein du groupe, nous avons construit le cahier des charges que nous avons essayé de respecter tout au long du projet. Aujourd'hui, nous pouvons dire que le cahier des charges a été respecté et que les améliorations que nous souhaitions apporter au projet ont été réalisées.
Nous nous sommes répartis le travail afin que chacun ait une partie à traiter et à présenter à chaque réunion (dont les comptes rendus sont disponibles en annexe). Nous avons désigné un chef de projet afin d'organiser les réunions et d'établir un lien avec notre tutrice. Au fur et à mesure des avancées, chacun de nous a donné son avis et a apporté ses connaissances pour faire progresser le projet.
Nous avons donc amélioré le jeu Sokoban afin qu'il propose plus de fonctionnalités. Nous avons changé l'interface du jeu, étoffé et corrigé l'éditeur, ajouté un système de score et créé des niveaux progressifs. Sur un plan technique, nous avons programmé en C/C++, ce qui nous a permis d'approfondir nos connaissances, en plus de découvrir de nouvelles possibilités de programmation.
Nous nous sommes aussi servis de la bibliothèque SDL [2], qui était la bibliothèque proposée dans la version d'origine. Son utilisation nous a permis de nous initier à la programmation multimédia en deux dimensions. En l'occurrence, nous avons appris à gérer les évènements, l'affichage et la gestion des périphériques ; ceci sera plus amplement détaillé dans la suite du rapport.

Après avoir présenté le cahier des charges de notre projet, nous expliquerons les choix effectués durant celui-ci. Nous exposerons également les problèmes rencontrés, et nous ter-

minerons par une brève conclusion qui portera sur le parcours qu'a fait notre projet.

Chapitre 1

Gestion du projet

Ce projet est une opportunité pour apprendre à travailler en équipe. La nôtre est constituée de 6 étudiants. Nous avons collectivement analysé la version existante du jeu et rédigé notre cahier des charges en conséquence. Au regard de la structure que présentait la version initiale du jeu, nous avons décomposé le projet en 3 grandes parties :

— l'éditeur de carte
— le jeu
— le tableau de bord

Nous avons réparti le groupe sur ces 3 parties en prenant en compte leur complexité. Ainsi Paola Andreu, Marine Pellicer et Thiziri Harcheb ont travaillé sur le jeu, Aurélie Murcia et Julien Roussé sur le menu et l'éditeur de cartes et Yanis Bled sur le tableau de bord. Nous avons également désigné un chef de projet afin de faciliter la mise en commun des différentes contributions.

Sur un plan organisationnel, nous avons décidé de faire une réunion bilan toutes les semaines afin de déterminer où en étaient les objectifs et si des problèmes s'étaient posés durant l'avancée des différentes parties.

Sur un plan technique, nous avons utilisé plusieurs outils informatiques afin de gérer le projet. En ce qui concerne la programmation, nous avons retenu l'outil de versioning *Git* [3]. Il nous a permis de mettre en commun et gérer plus aisément les différentes versions apportées par nos contributions. Nous avons de plus échangé des informations par l'intermédiaire d'outils classiques, comme la messagerie électronique ainsi que la messagerie instantanée Facebook (chat).

Qu'est-ce que la bibliothèque SDL ?

➟ La bibliothèque SDL est une bibliothèque de bas niveau particulièrement utilisée pour créer des jeux en 2D. Il s'agit d'une bibliothèque tierce, c'est-à-dire qu'elle n'est

pas initialement installée sur l'ordinateur. Il faut donc la télécharger et l'installer avant de pouvoir l'utiliser. Comme elle est écrite en C, elle peut être utilisée par des programmes en C, C++ (mais existe également en Java). C'est une bibliothèque libre et gratuite, multi-platforme.
Pour notre projet Sokoban, nous avons utilisé la SDL pour la création de fenêtres, la gestion des images, du son et du temps. En effet, la bibliothèque SDL permet de créer des fenêtres de la dimension souhaitée, elle permet également d'affecter une image donnée à une partie de la fenêtre créée en spécifiant les coordonnées (en pixels) de l'endroit où on souhaite voir l'image.
Cette bibliothèque a également des fonctions permettant d'effectuer des actions lors d'un évènement. Un *évènement* a lieu quand l'utilisateur appuie sur une touche du clavier, quand il clique avec la souris, quand il bouge la souris, quand il réduit la fenêtre, quand il demande à fermer la fenêtre... Elle prend donc en charge la gestion du clavier et de la souris.

Tous les développements ont été réalisés en C/C++ avec gpp *comme compilateur.*

Chapitre 2

Cahier des charges

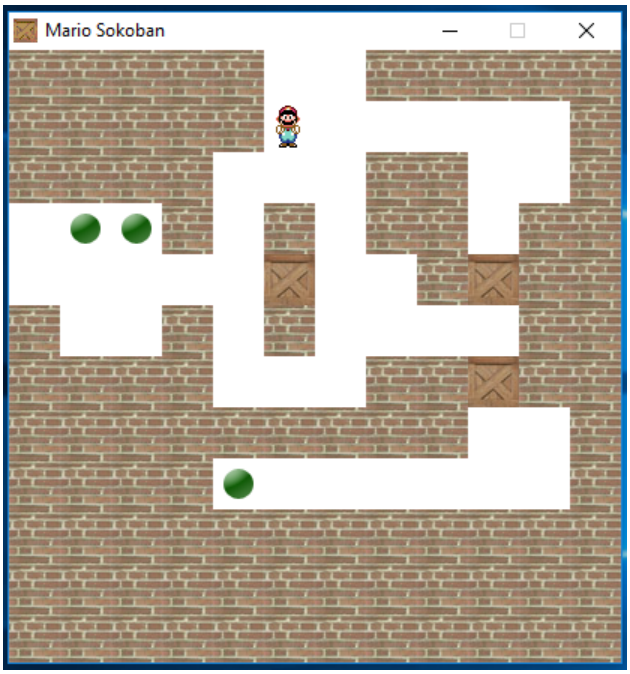
Dans cette partie nous allons présenter les différents objectifs que nous nous sommes fixés au début du projet.

Tout d'abord, voici quelques captures de la version sur laquelle nous avons travaillé :

✧ **Voici l'écran d'accueil :**

✧ **Voici l'écran de jeu :**

✧ **Voici l'écran de l'éditeur :**



(Comme vous pouvez le constater, sur la version de départ, l'éditeur et le jeu sont identiques en tous points)

Étant donné que le projet visait à améliorer les différents aspects du jeu, le cahier des charges peut être divisé selon plusieurs domaines, à savoir :

- "Jouabilité " :
  - créer un système de niveaux progressifs : dès que l'on réussit un niveau, le jeu enchaîne sur le niveau suivant
  - améliorer la facilité d'utilisation du jeu, son ergonomie, notamment avec des graphismes de meilleure qualité
  - proposer les touches « retour » et « recommencer »
  - ajouter la possibilité de charger une sauvegarde
  - placer les cartes créées dans l'éditeur de carte dans un niveau « personnalisé »
  - faire en sorte que le jeu puisse tourner en même temps que d'autres applications en améliorant les performances
- Fonctionnalités :
  - créer un mode d'emploi avec une aide ergonomique
  - prendre en charge la gestion de la souris
  - mettre en place un compteur de temps
  - en se basant sur le temps, installer un système de score
  - permettre au joueur de s'identifier et enregistrer ses scores
  - faire des vérifications avant de valider une carte créée dans l'éditeur
  - proposer un menu **options** (couper le son, changement des touches)
  - gérer des bonus/malus en fonction du temps mis à résoudre une carte
  - proposer des indices si le joueur bloque (avec possibilité de malus sur le score final)
  - ajout d'une bande son
  - ajout d'un choix pour la difficulté
- Graphismes :
  - rajouter un message de félicitations à la fin de chaque partie gagnée
  - mise en place d'un tableau de bord intuitif
  - mise en place d'un menu « Accueil »

Chapitre 3

Choix effectués

Comme vous avez pu le constater, de nombreuses améliorations et changements étaient nécessaires, nous allons dans cette partie vous décrire ce que nous avons mis en œuvre et pourquoi.

Dès le lancement du jeu, nous avons choisi d'animer une petite introduction, ce n'est pas un choix anodin. En effet, nous n'avons finalement pas fait de tutoriel, le jeu étant simpliste et relativement intuitif. Nous avons donc décidé de faire cette petite animation en introduction qui reprend le principe du jeu : un personnage déplace une caisse vers un objectif. Nous voulions ainsi donner un avant-goût du jeu et en rappeler le principe.

Passé l'introduction, nous arrivons sur l'écran d'accueil. Musique calme et design minimaliste ont été choisis ici pour mettre le joueur dans une ambiance de réflexion. D'ailleurs le design minimaliste, très épuré, a été choisi très tôt dans le projet pour suivre l'idée de rendre le jeu intuitif et agréable pour l'utilisateur.

À la base, nous voulions permettre à un joueur de s'identifier dès le début du jeu. Le problème est qu'il aurait alors fallu stocker toutes les données de tous les utilisateurs et leurs cartes personnalisées. C'est pourquoi nous avons décidé de ne demander le nom de l'utilisateur que si celui-ci faisait un nouveau meilleur score. De plus, mettre les cartes personnalisées en commun permet également de rajouter du contenu, et il est intéressant de pouvoir jeter un œil sur les créations des autres joueurs.

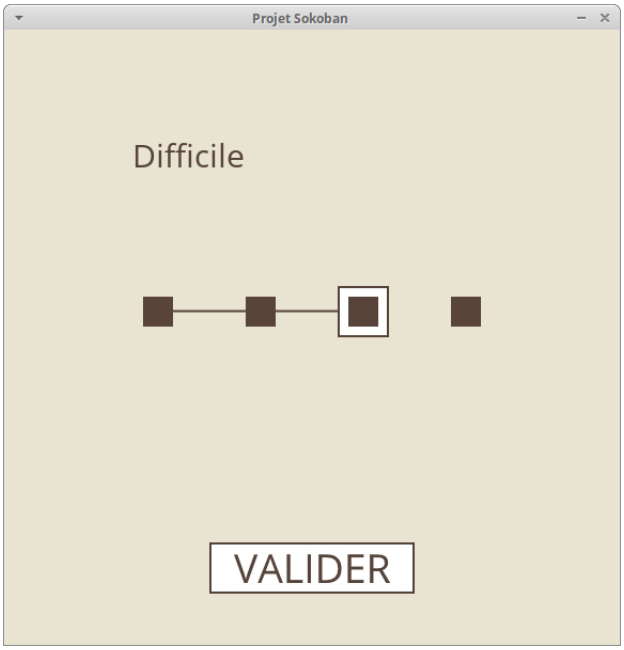
Puisque la bibliothèque SDL que nous avons utilisée ne permet pas de créer des boutons, nous avons dû utiliser une image pour indiquer à l'utilisateur quel choix était sélectionné (par défaut sur Jouer).

Nous avons, par souci d'ergonomie, rendu possible de naviguer dans le menu avec la souris aussi bien qu'avec les touches directionnelles. Il en va de même pour le reste du jeu.

Comme le jeu propose deux fonctionnalités qui sont l'éditeur (permettant de créer des cartes), et le jeu en lui-même, nous avons fait le choix d'un modèle hiérarchisé. C'est donc à partir du menu que l'on accède au jeu et à l'éditeur.

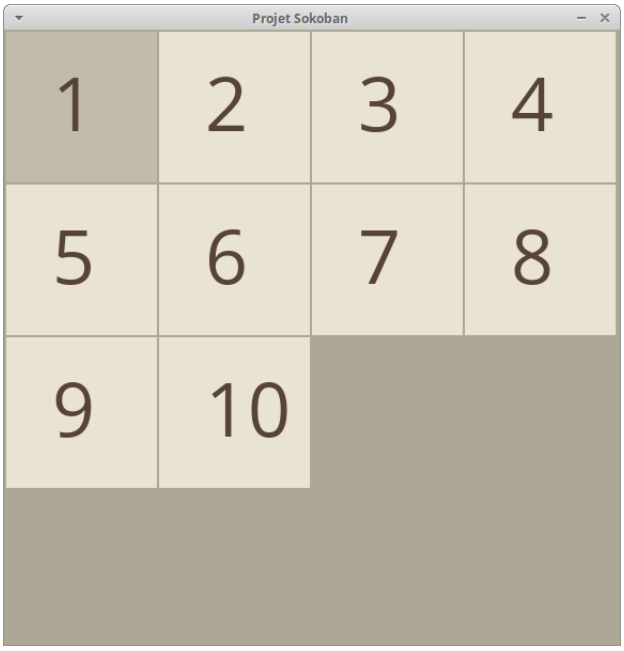
Nous allons détailler chaque améliorations apportées à ces deux aspects, séparément, et en commençant par la partie qui concerne le jeu Sokoban en lui-même.

✧ **Dans le jeu :**

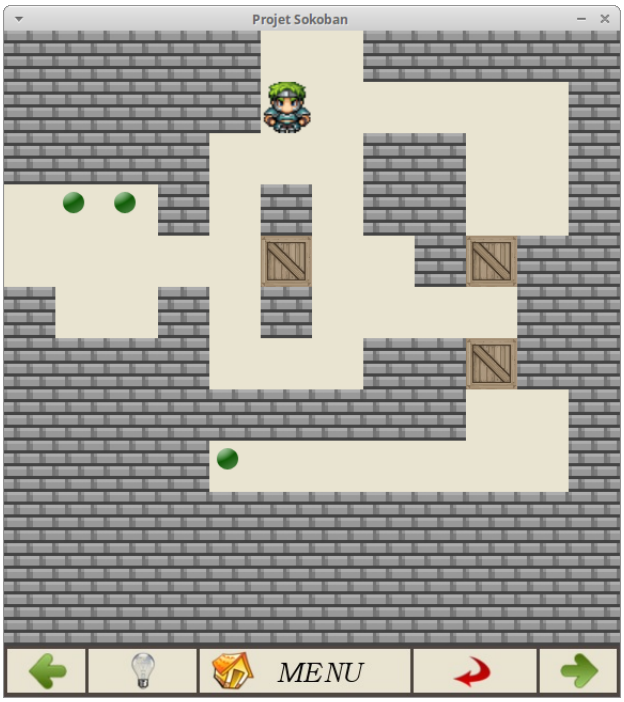


Si le joueur lance une partie, nous lui proposons d'abord de choisir le niveau de difficulté entre Facile, Normal, Difficile et Personnalisé – dans lequel il peut retrouver toutes les cartes créées via l'éditeur (sujet que nous aborderons par la suite).

Une fois le niveau de difficulté choisi, nous proposons à l'utilisateur de sélectionner une carte parmi les 10 qui sont proposées par défaut pour chacun des niveaux de difficulté.



Une fois le niveau et la carte choisis nous voilà donc en jeu.



Comme vous pouvez le voir, le jeu se démarque à présent de la version d'origine. En effet nous avons remplacé le décor pour le rendre plus « moderne », mais surtout pour pouvoir agrandir la taille de la fenêtre. Comme expliqué dans le cahier des charges ; dans la version d'origine, il était impossible de redimensionner la fenêtre qui était par défaut très petite.

Nous nous sommes rendus compte que c'était dû à la bibliothèque SDL, et surtout à l'utilisation d'images fixes. Redimensionner la fenêtre aurait changé la taille des images et donc détruit totalement le rendu souhaité. Notre version ne permet pas non plus de redimensionner la fenêtre, mais nous avons toutefois augmenté la taille des sprites (48x48 pixels) et donc la taille de la fenêtre.

Ensuite, par soucis de faciliter l'accès aux nouvelles fonctionnalités, nous avons ajouté un tableau de bord. Le tableau de bord est spécifique selon s'il se trouve dans l'éditeur ou dans le jeu.

Dans le jeu, le tableau de bord possède une flèche à chacune de ses extrémités. Elles permettent de naviguer entre les cartes ; celle de gauche donne sur la précédente, celle de droite sur la suivante.

La flèche rouge permet d'annuler le dernier déplacement effectué par le joueur et de revenir ainsi jusqu'au début. Si nous avons décidé d'offrir au joueur la possibilité de revenir en arrière, c'est parce qu'il suffit qu'une caisse soit bloquée pour qu'il n'y ait plus moyen de continuer. Dans la version que nous avons testée, il fallait alors quitter la carte et la recharger, ce qui devenait rapidement fastidieux.

Cette fonctionnalité a été longuement discutée, nous pensions au début mettre un message d'échec si le joueur bloquait toutes ses caisses, mais nous avons rapidement compris que cela pouvait rendre le jeu frustrant car dans certains niveaux, il faut essayer plusieurs fois avant de trouver la solution. Nous avons donc opté pour une fonction retour.

L'icône d'ampoule quant à elle, permet d'afficher la liste des commandes, c'est une petite aide pour les joueurs novices qui ne seraient familier avec les jeux vidéos.

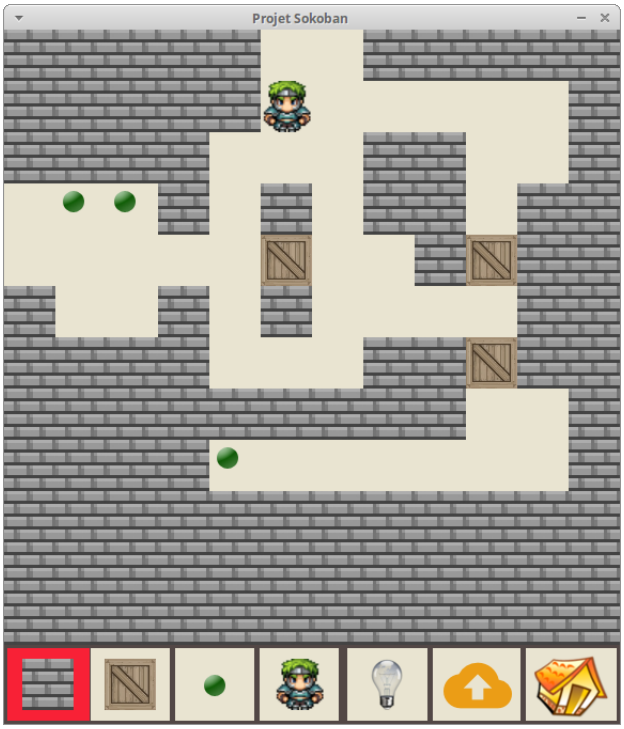
Enfin en passant le curseur sur l'icône menu, un menu déroulant offre le choix entre recommencer la carte actuelle, revenir à l'écran d'accueil ou voir les crédits du jeu.

Lors du test du jeu d'origine, nous nous sommes rendu compte que le jeu n'était pas adapté à tous les claviers. En effet, ceux ne possédant pas de pavé numérique étaient incapables d'accéder au jeu. C'est pour cette raison que nous avons rendu l'intégralité de la navigation dans le jeu possible à la souris ou au clavier. Ainsi, le joueur peut utiliser sa souris pour accéder au jeu, créer les cartes dans l'éditeur...

Seul le déplacement du personnage se fait uniquement avec les touches directionnelles. En effet, nous ne permettons pas de jouer avec la souris car, si le joueur désigne une position où le personnage doit se déplacer à l'aide de la souris, celui-ci va alors se téléporter à cet emplacement, ce qui rend impossible la gestion des caisses.

Enfin, afin d'ajouter un peu de compétitivité entre les différents joueurs, nous avons instauré un système de score. Celui-ci est basé sur le temps mis pour finir la carte. Pour chaque carte, les cinq meilleurs joueurs sont enregistrés dans un fichier texte à part, que le jeu va lire. Si le score est meilleur que ceux sauvegardés, alors le joueur a la possibilité d'entrer son nom directement dans le jeu.

✧ **Dans l'éditeur :**



Pour continuer avec l'éditeur, il a fallu corriger plusieurs problèmes avant de penser à ajouter des fonctionnalités. Premièrement, un bug permettait au joueur de poser plusieurs fois le personnage sur la carte. Deuxièmement, il pouvait créer des cartes inexploitables, avec un nombre d'objectifs différent du nombre de caisses. Pour résoudre le problème, nous empêchons la sauvegarde de la carte tant que le nombre de caisse(s) et le nombre d'objectif(s) ne sont pas égaux, et nous affichons un message d'erreur sur l'écran pour prévenir le joueur.

Ensuite, dans la version améliorée du Sokoban, sauvegarder une carte l'enregistrait à la place de la première carte. Afin de permettre au joueur de créer autant de cartes qu'il le souhaite, nous avons décidé de les stoker dans un niveau à part, appelé « personnalisé ». Mais pour éviter le stockage de doublons, nous avons réutilisé le système de message d'erreur afin de prévenir le joueur s'il tente d'enregistrer une carte déjà existante (nous vérifions si elle n'existe pas déjà).

Nous avons aussi effectué un travail d'optimisation. En effet la façon dont était programmé l'éditeur dans le code original a rapidement posé problème. À chaque fois qu'un bloc était ajouté ou supprimé, chaque case de l'écran était réaffichée, ce qui provoquait rapidement des ralentissements. Nous avons donc changé cela pour que seule la case modifiée soit réactualisée.

Chapitre 4

Problèmes rencontrés

Lors du projet nous avons rencontré plusieurs problèmes dont trois étaient directement liés à la bibliothèque SDL elle-même et donc impossibles à résoudre. Tout d'abord, nous souhaitions permettre un affichage en plein écran, mais les images utilisées par SDL ont une taille fixe. Il devenait alors impossible de s'adapter aux différentes résolutions d'écran. À la place, nous avons donc légèrement amélioré la qualité des images et agrandi la fenêtre de jeu, tout en gardant une taille qui convienne pour tous.

Le deuxième problème posé par la bibliothèque SDL que nous avons utilisée vient du fait qu'elle n'est pas compatible avec toutes les souris. En effet, certaines fonctionnent très mal avec SDL et provoquent d'importants ralentissements (ce dont nous nous sommes rendus compte durant les tests de notre version).

Le troisième point est également lié à la souris et aux performances. Bien qu'il n'empêche pas le jeu en lui-même, il peut se révéler problématique pour des ordinateurs à faible capacité. En effet, la gestion de la souris avec SDL est catastrophique. À chaque déplacement, le processeur est utilisé à 25%, ce qui est beaucoup trop gourmand en ressources.

Durant la programmation, nous avons également eut une incompatibilité de la fonction retour avec le score. En effet le score devait normalement s'afficher au cours du jeu mais ne s'actualisait que lorsque la souris ou le personnage étaient déplacés. L'alternative pour laquelle nous avons finalement optée est de ne montrer le temps réalisé qu'une fois la carte terminée.

Enfin, certains objectifs du cahier des charges n'ont pas été remplis. Notamment l'ajout d'une option qui aurait permis de changer la configuration des touches. En effet, cette proposition était censée palier au problème d'absence de pavé numérique, par exemple, ou des claviers qui ne disposaient pas de certaines touches. Mais la gestion de la souris a justement été ajoutée pour régler ce problème.

Le système d'indices pour guider le joueur a finalement été oublié, car combiné à l'avantage qu'offre la possibilité d'annuler un déplacement, il n'y aurait presque plus aucune difficulté au jeu et ce n'était pas ce que nous souhaitions.

L'idée de changer le décor en fonction du niveau de difficulté ne s'est pas concrétisée pour deux raisons. Déjà, le jeu gère beaucoup d'images (personnage, caisses, objectifs, fond, murs, menu, tableau de bord, ...) et nous ne souhaitions pas l'alourdir plus. Ensuite, parce que trouver des images qui s'accorderaient entre elles dans les différents niveaux de difficulté était une tâche presque impossible, à moins de les créer nous-même.

Enfin, nous devions ajouter un bouton « quitter » dans le tableau de bord mais nous avons finalement corrigé le bug de la version originale avec lequel cliquer sur la croix ramenait au menu. À présent, cela ferme directement le jeu. Nous avons donc annulé le bouton quitter pour le remplacer par un retour à l'accueil.

Annexe A

Comptes rendus

Voici les comptes rendus tenus au cours des différentes entrevues du TER :

Compte rendu du 25/01

I – Les critiques faites au jeu

I – 1) Défauts de jeu

Critique	Proposition
Le jeu continue de tourner même lorsque la pièce est bloquée	Proposer les touches « retour » et « recommencer »
In-jouabilité (éditeur de cartes)	Faire des vérifications avant de valider une carte
Possibilité de perdre la progression	Sauvegarde de la partie en cours
Pas de niveau de difficulté	Proposer un système de niveaux
Pas de progression personnelle	Créer un système de joueur avec identifiant

I – 2) Défauts d’ergonomie

Critique	Proposition
Jeu géré uniquement par des touches	Proposer une gestion par la souris
Le jeu n’est pas adapté à la souris	Création d’un menu pour les joueurs à la souris
Les règles/touches ne sont pas explicitées	Rajouter un mode d’emploi/tutoriel
Tous les claviers n’ont pas les mêmes touches	Permettre de changer la configuration des touches
Le jeu ne ferme pas une fois la partie terminée	Permettre de quitter le jeu
Cartes uniformes	Changement de décor en fonction du niveau de difficulté

I – 3) Autres

Résoudre le problème de librairie dynamique pour la portabilité Linux/Windows : possibilité de l’inclure en statique.

I – 4) Fonctionnalités supplémentaires

- Ajouter un système de scores prenant en compte le temps
- Gérer un système de bonus/malus en fonction du temps mis à résoudre une carte.
- Proposer des indices si le joueur bloque (avec possibilité de malus sur le score final).
- Ajout d’une bande son.

II – Objectifs pour la prochaine séance

Décortiquer le code existant et trouver les parties à garder, celles à modifier et celles qu’il faudra entièrement re-coder.

Compte rendu du 10/02

I – Les changements et le contenu des différents fichiers

1) À partir du programme initial

On part du code initial, on analyse ce que fait chaque partie.

Ensuite, on donne les modifications qui ont été faites à partir des objectifs listés dans le précédent compte-rendu.

En italique, les modifications à apporter

- main.c
  - Initialisation (charge l'affichage)
  - Lance des événements à travers différentes fonctions
    - Fin (accès de deux manières différentes (croix ou escape))
    - Jeu (touche 2) *On ajoute la possibilité de cliquer à la souris*
    - Éditeur (touche 3) *On va ajouter la possibilité de cliquer à la souris*
    - ✔ *On ajoute la touche 3 pour accéder aux options (que l'on veut ajouter), ainsi que la possibilité de cliquer à la souris*
    - ✔ *Ajout de la possibilité de charger une sauvegarde*
    - ✔ *Accès au mode d'emploi/tutoriel*
    - ✔ *Accès au tutoriel (touche 4 et souris)*
- jeu.c
  - *Ajout d'un choix pour la difficulté*
  - *Ajout d'une touche retour en arrière*
  - *Ajout d'une possibilité de sauvegarder*
  - *Ajout d'un système de score (de bonus/malus)*

- *Ajout d’indices*

- editeur.c
  - *Ajout d’une vérification avant validation*
  - *Changement de décor en fonction de la difficulté*
- *ajout d’un fichier « option »*
  - *permet la configuration des touches*
- *ajout d’un fichier « tutoriel »*

2) Répartition des tâches

Tableau de bord : Yanis

Main, **éditeur** : Julien, Aurélie

Jeu : Paola, Marine, Thiziri

Compte rendu du 08/02

I – Les difficultés rencontrées

- Impossibilité de redimensionner la fenêtre de jeu car les images sont fixes.

II – Idées/Changements

- Le main devient une page d'accueil et le tableau de bord sert à la navigation.
- Lorsque « nouvelle partie » est cliqué, une seconde page s'ouvre et permet de choisir la difficulté de la carte.
- Le score : le temps est compté (et augmente à chaque chaque fois qu'on utilise la touche retour). À la fin, un système de classement en fonction du temps mis. Enregistrement du meilleur score et éventuellement un classement général selon la progression dans le jeu.
- Changement des images afin d'améliorer la résolution.

III – Idée d'arborescence (incomplète)

- Menu / Tableau de bord
  - Nouvelle Partie
    - Nom
    - Niveau de difficulté
  - Options
  - Éditeur de niveau (« Créer ma carte »)
  - Charger Sauvegarde
  - Mode d'emploi/Tutoriel
  - Jeu

IV – Objectifs (pour le mercredi 22 février)

- Commencer la programmation (avoir une maquette jouable).
- Tracer l’arborescence.

Compte rendu du 22/02

Choix d'un modèle hiérarchisé du jeu

I – Main

- Si on clique sur Jouer
  - Nouvelle partie
  - Jouer partie
  - Retour (vers menu)
- Éditeur
- Options
- Ajouter le bouton quitter

II – Tableau de bord

1) Jeu

- Navigation
- Sauvegarder
- Recommencer
- Aide
- Option
- Accueil

2) Éditeur

- Sélection d'objet
- Sauvegarder la carte (dans un niveau personnalisé)
- Option
- Test carte
- Accueil

IV – Critique de la première version de Sokoban

- Un éditeur de carte présentant plusieurs bugs (notamment la possibilité de placer plusieurs personnages et des problèmes de performance (chaque case de l'écran était ré-affichée à chaque fois) : corrigé.
- Lorsque l'on veut jouer : une page avec facile, moyen, difficile et personnalisé permettant d'afficher les cartes du niveau choisi.

V – Hiérarchie de Sokoban

- Main
  - Depuis Jouer : nouvelle partie ou charger partie
    - Retour → Main
  - Vers Éditeur
    - Retour → Main
  - Options
  - Quitter

VI – Objectifs

- Faire une hiérarchisation précise des fichiers
- Planification des tâches (à noter)
- Modification du tableau de bord
- Retravailler l'éditeur de carte (placer les cartes dans un niveau « personnalisé », vérifier que la carte est valide

VII – Travail effectué

Main :

– L'écran d'accueil entièrement refait avec les liens vers jeu, éditeur et options (Julien)

Éditeur :

– Bug des plusieurs personnages, réglé (Aurélie)

– Amélioration du code pour palier aux latences (Aurélie et Julien)

– Ajout d'options (gestion à la souris et au clavier) pour pouvoir ajouter les différents objets de l'éditeur (Julien)

Jeu :

– Fonctions servant à sauvegarder et charger la partie (Marine)

– Ajout d'un chrono, affiché dans une fenêtre à part (provisoire) (Paola)

Tableau de bord :

– Ajout d'un tableau de bord sur jeu et éditeur (Yanis)

Compte rendu du 01/03

I – Objectifs

- Fin mars : version montrable (de secours)
- Deux dernières semaines d'avril (préparation pour la soutenance)
- Un ou deux transparent max par personne pour expliquer le travail effectué
  - Raisons du changement
  - Comment on l'a changé
  - Où se trouve le code dans la hiérarchie (le main, le jeu...)
- Le code se trouvera en annexe du rapport, (possibilité de mettre Doxygen en annexe également)
- Dans le rapport et la soutenance (en conclusion), parler de ce qu'a apporté ce TER ; compréhension de SDL, critique d'un code, …

II – A ajouter

() = à faire à la fin

- Statistiques
- Changement de carte
- Bouton « Help », « Son »
- (Classer les cartes par utilisateur)
- (Tutoriel)
- (Des configurations de touches)
- Quand la partie est finie ou qu'on sauvegarde une carte, demander à saisir le nom

Compte rendu du 15/03

I – Problèmes rencontrés

- 1. La fonction retour ne fonctionne pas encore avec le chrono
- 2. Dans l'éditeur, la croix ne ferme pas le jeu mais retourne au menu
- 3. Jeu trop gourmand en ressources : processeur (dû à la gestion de la souris par SDL) et en RAM

II – Solutions ?

- 1. N'afficher le score qu'à la fin
- 2. Réduire le main

III – Encore à faire

  - Urgent (pour avoir une version montrable)
- Corriger les bugs
  - Détails supplémentaires
- Le menu option
- Si le joueur passe sur le menu en jeu, il faut gérer le chrono pour ne pas que ce temps compte
- Une option « recommencer/recharger la carte »
- L'aide

IV – Pour le compte-rendu

Parler des limites rencontrées avec SDL et le C++, car on partait d'un jeu existant codé en C. Expliquer pourquoi le choix des outils de base est donc très important.

Compte rendu du 29/03

I – À faire

- Makefile
- Score, classement et entrer le nom une fois le score affiché
- Enlever les options
- (éditeur) dire que la carte a été sauvegardée

(La version Window n'a pas de son)

II – Pour le rapport

- Au final, les options n'étaient pas nécessaires (elle devaient permettre de changer les touches, mais avec la gestion de la souris, ce n'était plus utile)
- Gestion dans le jeu de la souris est catastrophique (trop gourmand en processeur : 25% à chaque déplacement de la souris)
- Mettre le « Read me » en annexe du rapport

III – Questions probables

- Par rapport au cahier des charges, que manque-t-il ? Qu'est-ce qui n'a pas été fait et pourquoi ?
  - Les options
  - La gestion de la souris dans le jeu
- Expliquer le système de classement

IV – Présentation

- Bien donner le contexte ; on a repris un travail existant, parler des difficultés (temps, gestion souris sur le jeu, limites SDL), dire qui a fait quoi (l'organisation) et dans quelles conditions on a travaillé
- Contrainte : pour pouvoir jouer au jeu ; sur Linux il faut installer SDL (sur Windows on fournit un .exe)
- À la fin de la présentation, conclure par « pour toutes les questions techniques, nous sommes à votre disposition »

- Montrer une comparaison entre la version d'Open Classroom et la nôtre (interface ajoutée, possibilité d'éditer plusieurs cartes, gestion d'un système de niveau, système de score, fonction retour)

Conclusion

Notre projet consistait à améliorer une version préexistante d'un jeu de réflexion : « mario sokoban » et ainsi à compléter un travail déjà existant (en langage C que nous avons adapté en langage C++ par la suite).

Dans un premier temps, ce projet nous a amené à maîtriser l'utilisation de la bibliothèque « SDL » qui nous était inconnue. Cette bibliothèque nous a permis de nous initier à la programmation multimédia en 2D, nous a appris à gérer les événements ainsi que l'affichage et la gestion des images, du son, de la souris.

Néanmoins nous nous sommes vite rendu compte que cette bibliothèque n'offrait qu'un champ de travail limité, en effet nous avons rencontré beaucoup de problèmes, dont deux principaux. Le premier concernait le processus ; dès lors que nous avons intégré la gestion de la souris, le jeu devenait trop gourmand en ressources. Le deuxième était au niveau de l'assemblage de la fonction retour et du chrono qui se sont révélés difficilement compatibles. Cela nous amène à dire que le choix des outils qui seront utilisés tout le long du projet est une étape primordiale.

De plus, il s'agissait ici d'un projet de création d'un jeu en 2D. Si nous sommes un jour amené à créer un autre jeu, nous choisirons certainement un autre environnement.

Dans un deuxième temps, ce projet nous a permis d'approfondir et d'enrichir nos connaissances dans le langage C/C++.

Nous estimons qu'à la fin de ce projet, le travail obtenu est satisfaisant dans la mesure où cette version améliorée respecte le cahier des charges qui était fixé, dans la mesure des limites imposées par les conditions de départ (reprendre un travail en C, qui utilisait SDL).

Bien entendu, les possibilités d'amélioration ne se limitent à notre travail. Il serait possible de le rendre encore meilleur : en le mettant en ligne, ou à disposition au téléchargement, en ajoutant un multi-joueurs, en programmant des cartes aléatoires qui seraient générées à l'infini...

Pour conclure, nous tenions à remercier notre encadrant, Violaine Prince, qui nous a guidés et aidés tout au long du projet.

Bibliographie

[1] Mathieu Nebra. **Mario Sokoban**. *https ://openclassrooms.com/courses/apprenez-a-programmer-en-c/tp-mario-sokoban*.

[2] Sam Lantinga. **Simple DirectMedia Layer** (SDL 1.2), 1998.

[3] Université Montpellier. **GitLab**. *https ://gitlab.info-ufr.univ-montp2.fr/users/sign_in*.