Introduction

Ce projet a pour but de montrer nos capacités à restituer nos connaissances et nos capacités algorithmique suite à la semaine thématique de C.  
Le sujet se nomme Malloc World. Il consistait à créer un jeu de plateau où l’on devait générer aléatoirement des ressources et des monstres. Notre héros devait récolter les ressources pour crafter des armes et outils afin de vaincre ses monstres.  
Pour réaliser ce projet, nous étions deux étudiants :  
- Erwan DUPREY en classe 3IABD2.  
- Tom CAREGHI en classe 3IABD1.

Analyse de l’application

Liste des structures de données :  
Nous avons une structure Character pour le personnage jouable dans laquelle il y a les variables hp correspondant aux points de vies actuels du Character, hpMax correspondant aux points de vies maximal que peut obtenir un character, exp correspondant à l’expérience actuel du personnage, expMax correspondant aux points d’expériences maximal que peut obtenir un character, lvl correspondant au niveau actuel du personnage, numberOfItem correspondant au nombre d’item que le personnage a dans son inventaire et un tableau d’une structure se nommant Item faisant 10 cases correspondant à son inventaire.  
  
Nous avons une structure Items pour les personnages dans l’inventaire, le coffre et le système de craft. Il y a les variables id permettant de reconnaitre l’item pour les différentes fonctions de l’application, la variable number permettant de savoir le nombre de cet objet dans la structure, le damageArmorHealth permettant mettre une valeur de dégâts pour les armes, d’armure pour l’armure et de point de vie à restaurer pour les potions. La variable durability permettant de mettre la durabilité restante des outils et des armes. La variable name permettant de mettre un nom à cet objet.

Nous avons une structure Chest pour le coffre du PNJ. Il s’agit d’une liste chainée comprenant l’id de l’item, la variable qty pour la quantité d’objet de cet objet dans l’inventaire ainsi que le pointeur de la prochaine liste Chest.

Nous avons la structure Monster pour les monstres. Nous avons la variable hp pour ses points de vies, la variable exp pour le gain d’expérience, le lvl pour l’id du monstre correspondant à son niveau. La variable DPS pour le nombre de dégâts par tour du monstre. La variable name pour le nom de ce monstre.

Nous avons une structure Pile avec une structure du premier Element.  
Nous avons une structure Element qui est une liste chainée d’élément i et j correspondant au x et y de la case sur la carte.

Liste des fonctions principales :  
Dans interface :  
La fonction mainMenu va permettre à l’utilisateur de faire un choix entre faire une nouvelle partie, en charger une nouvelle ou de quitter le jeu.  
La fonction loadGame va permettre de récupérer ou de créer tous les éléments du jeux. Il va aussi lancer les fonctions du jeux et de libérer les allocations mémoires.  
La fonction coreGame qui va nous permettre de générer le menu de choix du joueur, de gérer la fin du jeux et le changement de map.  
La fonction pnj va nous permettre de charger tous les éléments de l’interface du pnj comme la réparation, la création d’objet, mettre dans le coffre les objets et les retirer ainsi que recharger la carte.  
La fonction exitMenu qui permet au joueur de sauvegarder et de quitter le jeu.  
Les fonctions findCharacter, trackMovement et shifting permettent le déplacement du joueur sur la carte et la gestion des événements.  
La fonction fight permet de lancer l’interface de combat du joueur.  
La fonction freeGame permet de libérer les allocations mémoires.

Dans character :  
La fonction createCharacter permet la création du personnage lors d’une nouvelle partie.  
Les fonctions isLevelingUp et levelingUp permettent de gérer les montées de niveau du personnage.  
La fonction printInventory permet de montrer l’inventaire.  
La fonction freeInventory permet de libérer l’allocation mémoire de l’inventaire.  
Les fonctions newItem et takeOffItem permettent d’ajouter et d’enlever les objets de l’inventaire.  
La fonction repair va réparer tous les objets de l’inventaire.  
Les autres fonctions sont de natures utilitaires.

Dans chest :  
La fonction newItemChest permet d’ajouter un objet dans le coffre.  
La fonction freeChest permet de libérer les allocations mémoires du coffre.  
La fonction printChest permet de montrer le coffre.  
Les autres fonctions permettent de sélectionner, ajouter et supprimer des objets du coffre.

Dans craft :  
La fonction crafting permet de donner un objet au héros en fonction de l’id de l’objet donné.  
La fonction searchInInventory permet de chercher un objet et la quantité de cet objet dans l’inventaire et de retourner 1 si tel est le cas.

Dans items :  
La fonction giveItemToHero permet de donner un objet avec sa quantité en retour de la fonction.

Dans map :  
La fonction display affiche la carte.  
La fonction copyMap permet de faire une copie de la carte.  
La fonction randomizedItem permet de mettre tous les objets nécessaires à une carte.  
La fonction freeMap permet de libérer les allocations mémoires de la carte.  
La fonction labyrinthe avec le reste des fonctions permettent de générer un labyrinthe.

Dans monster :  
La fonction creatMonster permet de générer les monstres.

Dans save :  
Les fonctions commençants par save vont permettre de sauvegarder dans le fichier. Les fonctions commençants par retrieve vont permettre de récupérer les données de la sauvegarde dans le fichier.  
Si le nom de la fonction est agrémenté d’un mot comme map, cela signifie que la carte est concerné par la fonction.

Détails des choix d’implémentations :  
Le coffre est une liste chainée car elle a un nombre d’élément illimitée et il fallait pouvoir supprimer des éléments du tableau.

La pile est une liste chainée d’éléments car le labyrinthe se générant aléatoirement, nous ne savons pas combien de fois nous allons boucler avant d’arriver à un labyrinthe finit.

Le reste des autres éléments sont des tableaux de structure car nous avions bien plus besoin d’accéder rapidement à un élément que de l’enlever.

Dossier d’utilisation de l’application

Toute les instructions du déroulement et des choix possibles à chaque étape sont notés dans l’interface. Pour faire un choix, il suffit de taper un chiffre et d’appuyer sur la touche entrée pour valider le choix.  
Si un seul chiffre ou une seule lettre est demandée, seule la dernière tapée sera pris en compte.

ATTENTION : si un chiffre est demandée et qu’une lettre ou tout autre caractère est entrée dans l’interface, le programme va crash et inversement pour les lettres.

Pour les différents chiffres affichés dans l’interface de la carte, veuillez vous référez à l’image ci-dessous :

Une image contenant table

Description générée automatiquement

Une image contenant table

Description générée automatiquement

Il arrivera parfois que le jeux vous bloque dès la première rencontre. Il suffira de relancer une nouvelle partie.

La réapparition des monstres et des ressources se fait uniquement en parlant au PNJ et avec sa montre magique !

Bilan

Points non résolus :  
- La SDL n’est pas faite.  
- La régénération des monstres et des ressources est modifiée (ce n’est plus avec un timer mais en parlant au PNJ).

Bilan général :

Nous n’avons pas rencontrés de difficultés humaines. Nous avons rapidement prit contact et nous nous sommes répartis les tâches équitablement en imposant au minimum un point par semaine.

D’un point de vue technique, nous avons séparer les tâches dès le début du projet. Nous avons donc décider de prendre chacun ce qui nous plaisait et d’en faire un rapport tout du long. Nous étions complémentaires. Seul l’interface s’est faite à deux lors d’une grosse réunion.  
Cependant, nous avons eu des difficultés avec la génération de la carte car plusieurs versions ont été réalisées pour correspondre au mieux au sujet. Les monstres ont aussi posées problèmes car il s’agissait d’une liste chainées et nous nous retrouvons avec un tableau pour optimiser notre temps de création du programme. On peut aussi dire, que faire un jeu équilibré nous as prit du temps. La sauvegarde a posé problème non pas pour son écriture mais la récupération des données avec les tailles de cartes qui peuvent être différentes.  
Malgré tout ses problèmes, nous avons complété chacune des tâches demandées lors de ce projet (hormis la SDL).