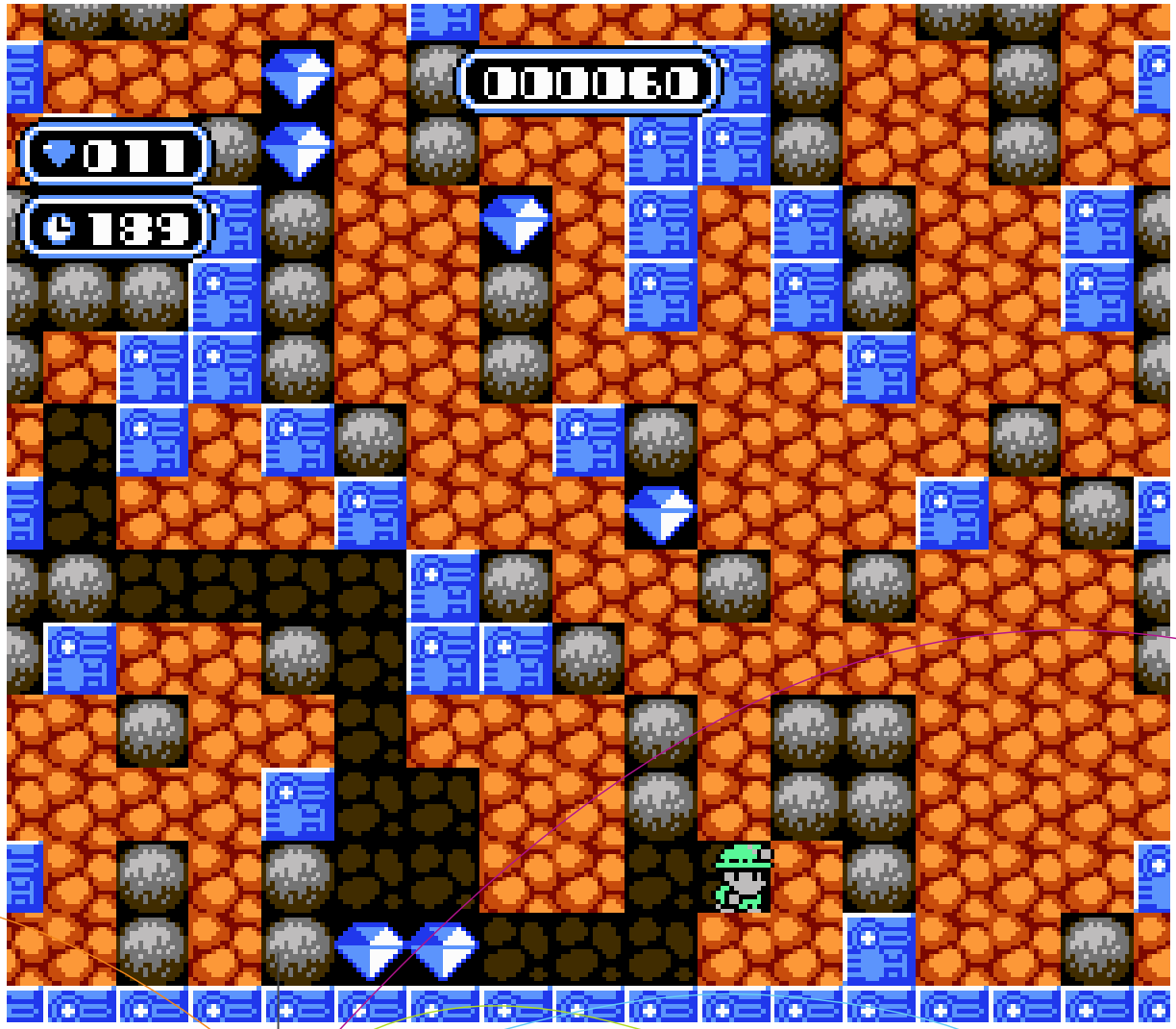
INSA
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
ROUEN NORMANDIE



Maxime Defromerie
Julie Langrand
Victor Léger
Romain Petitalot
Tom Tesniere

Rapport de projet I4-2

Boulder Dash

Introduction

Au cours du semestre 4 de STPI2, nous avons réalisé un projet informatique tout au long du module d'I4-2. Pour ce projet, nous avions le choix entre réaliser soit le jeu Boulder Dash soit le jeu Frogger. Notre choix s'est porté sur le jeu Boulder Dash.

Ce projet mêle différentes compétences et il est basé sur l'organisation pour le travail en groupe. Effectivement, nous verrons que l'utilisation de Gitlab a permis de travailler plus facilement sur le projet et tous en même temps. En outre, développer le jeu Boulder Dash a permis de mieux comprendre comment créer un jeu en utilisant la SDL et de renforcer nos compétences en programmation (ici en pascal). La rédaction de ce rapport nous a conduit à utiliser le LaTeXafin de séparer le fond de la forme, ce qui est parfait pour un rapport de ce type.

1 Analyse du projet

Dans cette première partie, nous allons décrire le projet, présenter les fonctionnalités et conceptions que nous avions initialement réalisées. En outre, nous discuterons des éventuelles modifications apportées pour le rendu final.

1.1 Cahier des charges

1.1.1 Description du projet

Le groupe a décidé de faire un Boulder Dash, car nous y avions déjà tous joué avant ce projet. De plus, les graphismes sont agréables et le jeu est relativement addictif. Ainsi, notre choix s'est rapidement tourné vers ce dernier et les idées de conception du jeu ont été rapides.

Boulder Dash est un jeu où l'on incarne un petit personnage dont le but est de ramasser des diamants en creusant la terre. Ce jeu possède différents niveaux dont la difficulté est progressive. Dans ces niveaux, on retrouve des ennemis et des rochers. Le joueur possède 3 vies, si le joueur se fait écraser par un rocher ou toucher par un ennemi, il perd une vie, au bout de 3 vies perdues la partie du joueur est terminée. Certains niveaux ne possèdent pas de diamants, le joueur doit alors tuer les ennemis du niveau en faisant tomber un rocher sur eux ce qui fera apparaître des diamants. Les diamants peuvent aussi tomber sur d'autres ennemis les tuant à leur tour. Le joueur peut aussi casser des murs pour atteindre des zones bloquées grâce aux rochers. Nous avons aussi la volonté d'ajouter des fonctionnalités facultatives telles que la création de niveau par l'utilisateur ou encore la personnalisation du personnage jouable (ces fonctionnalités sont listées dans la partie suivante).

1.1.2 Liste des fonctionnalités

Le rendu final de notre projet doit comporter les fonctionnalités suivantes :

- Déplacement du joueur
- Chargement des niveaux (mur, terre, bords, rochers) depuis un fichier .txt
- Sauvegarde de la partie
- Différents niveaux
- Menu
- Victoire (débloquer passage pour terminer niveau)
- Détection de mort (écrasé par un rocher, touché par un ennemi)

Nous avons prévu des fonctionnalités optionnelles, afin d'améliorer le jeu, que nous ferions si le temps le permettait :

- Création de niveaux par l'utilisateur
- Musique

— Personnalisation joueur
— Ajout d'ennemi et/ou de PNJ
— Choix mondes
— Chronomètre
— Animation mort personnage (si écrasé par un rocher ou touché par un ennemi)

Versions prévues :
V1 (08/04/2021) :
— Déplacement du joueur
— Chargement de niveaux depuis des fichiers .txt

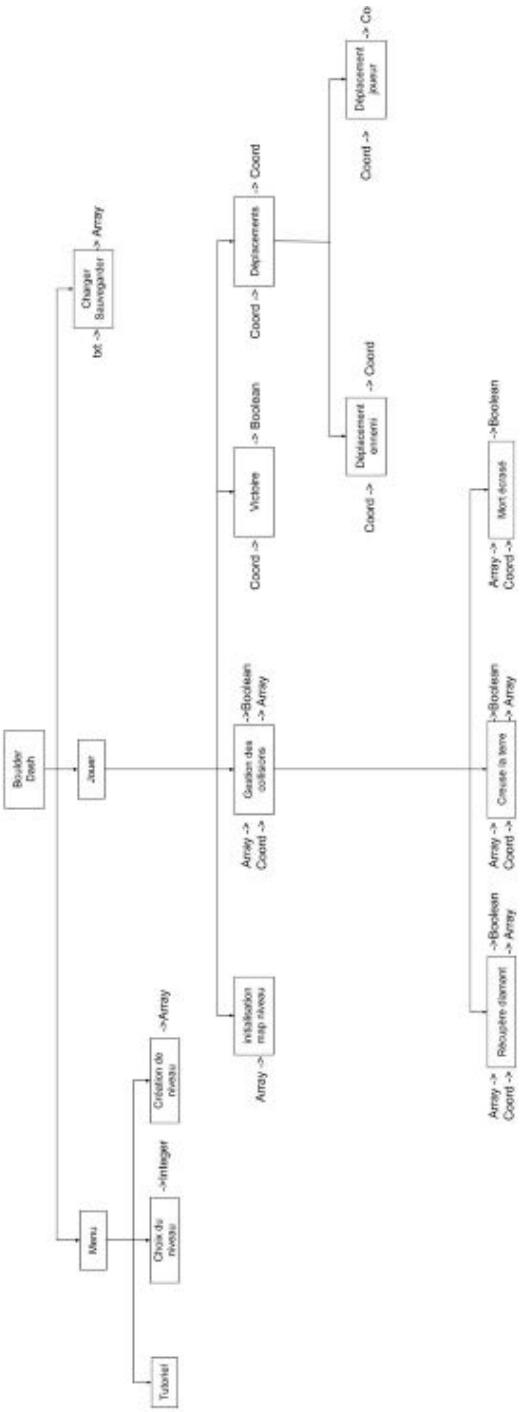
V2(22/04/2021) :
— Mise en place de différents niveaux
— Sauvegarde de la partie
— Implémentation du menu
— Mise en place de la détection de mort

V2(07/06/2021) :
— Finalisation des fonctionnalités présentes
— Implémentation des fonctionnalités facultatives

1.2 Conception globale

Cette section présente la conception globale de notre programme implémentant les fonctionnalités données en section 1.1.2. L'analyse descendante correspondante est donnée à la page suivante.

Voici l'analyse descendante intiale de notre projet :



Cette analyse descendante a été relativement bien suivie. Cependant, nous avons tout de même changé quelque peu cette dernière. Les principales modifications sont les suivantes :

— changement n°1 : Un des changements notable est l'implémentation de différentes unités. Une première unité ("rockfordUtils.pas") a été utilisée pour stocker les différentes structures utilisées tout au long du programme principal. Une seconde unité ("menurockford") a été implémentée permettant de bien séparer le jeu principal du menu et donc de simplifier l'écriture du code pour ces deux parties.

— changement n°2 : On peut noter de plus que certaines entrées et sorties ont été modifiées et de nouvelles ont été rajoutées, n'apparaissant pas dans l'analyse descendante, du fait de l'adaptation de notre programme au langage de programmation SDL.

1.3 Conception préliminaire

Le contenu de cette section présente la signature des fonctions et procédures de notre analyse descendante, en précisant les entrées, sorties et entrées/sorties.

---

procedure JOUER( )

procedure GESTIONCOLLISIONS(E c : Coord, E/S t : Tab, S Coli : booleen)

function MORTECRASE(t : Tab, c : Coord) : Booléen

procedure RECUPEREDIAMANT (E c : Coord, E/S t : Tab, S Dia : Booléen)

procedure CREUSETERRE(E c : Coord, E/S t : Tab, S Terre : Booléen)

function VICTOIRE(c : Coord) : Booléen

procedure DEPLACEMENT(E/S c : Coord)

procedure DEPLACEMENTJOUEUR(E/S cPlayer : Coord)

procedure DEPLACEMENTENNEMI(E/S cEnnemy : Coord)

procedure INITMAPNIVEAU(E t : tab)

procedure CHARGERSAUVEGARDER(E f : txt, S t : Tab)

procedure MENU( )

procedure TUTORIEL( )

procedure CHOIXNIVEAU

procedure CREATIONNIVEAU(t : Tab)

A nouveau, quelques changements ont eu lieu en pratique :

— changement n°1 : La procédure tutoriel n'a pas été réalisé compte tenu de la simplicité du jeu, la procédure CreationNiveau n'a pas été implémenté dans notre code, car ce n'était pas pertinent pour notre jeu et que nous avons préféré nous focaliser sur des aspects plus important du jeu.

— changement n°2 La procédure RecupererDiamant a été directement mise dans celle du déplacement du joueur car c'était plus simple à coder.

— changement n°3 Comme cité précédemment, l'utilisation de la bibliothèque SDL nous a contraint a modifié les entrées et sorties de chacune des procédures et fonctions.

De plus, nous avons utilisé de nombreux types structurés dans notre projet, à commencer par les coordonnées du Menu (coordMenu), les coordonnées du joueur (coordonnees) mais aussi un type block pour définir le tableau de structures Terrain qui était un composé de block pour représenter la grille de jeu.

1.4 Procédures ou fonctions importantes

Les deux procédures que nous avons retenues sont :

- La procédure "chargement"

- La procédure "tombePierre"

Ces deux procédures nous semblaient pertinentes, car elles sont nécessaires au bon fonctionnement de notre programme et leur implémentation s'est avérée quelquefois plus complexes que l'on pensait.

La difficulté principale dans la procédure chargement était de charger toutes les données du niveau notamment la position du personnage, de l'arrivée, et l'ensemble des objets du niveau. De plus, la procédure SauvegarderNiveau devait être complémentaire à cette procédure pour charger des fichiers sauvegardés et se souvenir, par exemple, du nombre de diamant déjà récupéré ou du temps écoulé. Quant à la procédure tombePierre, la difficulté résidait dans le fait de gérer tous les types d'éboulements (par le bas et les côtés) pour les pierres et les diamants et les différents délais d'affichage pour chacun des mouvements et également les collisions avec notre personnage et les PNJ.

```
1  //Procedure chargement du terrain de jeu
2  procedure chargement (name : string; var T : Terrain;var posRF, posFin : coordonnees;var nbDiamant, nbDiamantFin,
3  var fic        : Text;
4          i, j : Integer;
5          str: String;
6  begin
7          assign(fic,name + '.txt');
8          reset(fic);
9          i:=1;
10         read(fic, posRF.x);//On commence par lire les données du niveau
11         readln(fic, posRF.y);
12         read(fic, posFin.x);
13         readln(fic, posFin.y);
14         read(fic, nbDiamant);
15         readln(fic, nbDiamantFin);
16         readln(fic, reserveTemps);
17         while (not eof(fic)) do
18         begin
19                 readln(fic,str);
20                 for j := 1 to 24 do
21                 begin
22                         T[i][j].genre := StrToInt(str[j]);//On lit la map
23                         T[i][j].mouvement := '';
24                 end;
25                 i:=i+1;
26         end;
27         close(fic);
28         T[posRF.y][posRF.x].genre := 0;//Notre personnage part sur une case vide
29         initPapillon(T);//Calcul de la direction dans laquelle les papillons vont devoir partir
30 end;
```

```
1  //Procedure tombePierre permettant la chute de la pierre
2  procedure tombePierre(var window, rockford : Psdl_Surface; var T:Terrain; position : coordonnees;
3  nomObjet:string; var nbDiamant : Integer; var fin : Boolean );
4  var i, j, numeroObj, coordY : Integer;
5  begin
6          if nomObjet = 'Pierre' then
7                  numeroObj := 3
8          else if nomObjet = 'Diamant' then
9                  numeroObj := 4;
10
11         for i := 1 to largueur do
12         begin
13                 for j := 1 to longueur do
14                 begin
15                         if T[i][j].mouvement = nomObjet then
16                         begin
17                                 T[i+1][j].genre := numeroObj;
18                                 T[i][j].mouvement := '';
19                                 afficherfond(window, rockford, T, position, True);
20                                 if (position.y = i+2) and (position.x = j) then
21                                         fin := True;
22                                 if T[i+2][j].genre = 6 then
23                                 begin
24                                         mortPapillon(T, j,i+1, nbDiamant, position);
25                                         mortPapillon(T, j,i+2, nbDiamant, position);
26                                         mortPapillon(T, j,i+3, nbDiamant, position);
27                                 end;
28                         end;
29                         if T[i][j].genre = numeroObj then
30                         begin
31                                 if (T[i+1][j].genre = 0) and not((position.y = i+1) and
32                                 (position.x = j)) then
33                                 begin
34                                         T[i][j].genre := 0;
35                                         T[i][j].mouvement := nomObjet;
36                                         afficherfond(window, rockford, T, position, True);
37                                         SDL_Flip(window);
38                                 end
39                                 else if (T[i+1][j].genre = 6)then
40                                 begin
41                                         mortPapillon(T, j,i, nbDiamant, position);
42                                         mortPapillon(T, j,i+1, nbDiamant, position);
43                                         mortPapillon(T, j,i+2, nbDiamant, position);
44                                 end;
45                         end;
46                         if T[i][j].genre = numeroObj then //Les éboulements sur le côté
```

```
47                          begin
48                                  if (T[i+1][j].genre = 3) or (T[i+1][j].genre = 4) then
49                                  begin
50                                          coordY := i+1;
51                                          while (T[coordY][j].genre = 3)or(T[coordY][j].genre = 4) do
52                                          //Pour verif que la pierre en dessous ne tombe pas
53                                          begin
54                                                  coordY := coordY +1;
55                                          end;
56                                          if T[coordY][j].genre <> 0 then
57                                          begin
58                                                  if (T[i+1][j+1].genre=0)and(T[i][j+1].genre=0)and
59                                                  not((position.y=i+1)and(position.x=j+1))and
60                          not((position.y=i)and(position.x=j+1))
61                          and(T[i][j+1].mouvement='')then
62                                                  begin
63                                                          T[i][j+1].genre := numeroObj;
64                                                          T[i][j].genre := 0;
65                                                          afficherfond(window,rockford,T,position,True);
66                                                          SDl_delay(50);
67                                                          T[i][j+1].genre := 0;
68                                                          T[i+1][j+1].genre := numeroObj;
69                                                  end
70                                                  else if(T[i+1][j-1].genre=0)and(T[i][j-1].genre=0)and
71                                                  not((position.y=i+1)and(position.x=j-1))and
72                                              not((position.y=i)and(position.x=j-1))
73                                              and(T[i][j-1].mouvement='')then
74                                                  begin
75                                                          T[i][j-1].genre := numeroObj;
76                                                          T[i][j].genre := 0;
77                                                          afficherfond(window,rockford,T,position,True);
78                                                          SDl_delay(50);
79                                                          T[i][j-1].genre := 0;
80                                                          T[i+1][j-1].genre := numeroObj;
81                                                  end;
82                                          end;
83                                  end;
84                          end;
85                  end;
86          end;
87          afficherfond(window, rockford, T, position, True);
88  end;
```

1.5 Partie développement

1.5.1 Les unités et le programme principal

— Unité rockfordUtils : Cette unité comprend tous les différentes constantes et les différents types utilisés dans notre projet. Pour les constantes, on y retrouve par exemple la largeur et la longueur de la fenêtre de jeu. Pour les différentes unités, nous avons notamment les coordonnées du joueur en x et y, les coordonnées du menu en x et y, mais aussi avec une variable représentant le choix du joueur dans le menu. Puis un tableau Terrain composé du type "block" qui représente les différentes cases de notre jeu.

— Unité menurockford : Cette unité est composé de plusieurs procédures d'affichages des différentes parties du menu, avec le fond mais aussi le curseur (représenté par le personnage rockford), d'une procédure pour la gestion du clavier de l'utilisateur et deux procédures pour gérer le son. Toutes ces procédures sont regroupées dans la procédure menu qui les combinent. La procédure menu est ensuite utilisée dans le code principal du jeu. On peut également citer les deux dernières procédures : ProcessKeyFin et choixFin qui permettent de quitter le jeu en le sauvegardant ou non.

— Programme principal : Le programme principal du jeu qui regroupe toutes les procédures de gestion du jeu, celles liées aux déplacements du joueur, des ennemis ou des pierres mais aussi des procédures d'affichages de tous les blocs et celle pour le temps de jeu. Puis nous avons les procédures pour sauvegarder et charger le jeu à partir de fichier texte qui se situe dans un dossier intitulé "Ressources". Le code principal initialise les variables, reprend le menu puis charge le jeu. Ensuite, le jeu se fait grâce à une itération sur le déplacement du joueur et des ennemis. Enfin, suivant le choix du joueur, si ce dernier meurt ou gagne, le jeu s'arrête. Mais s'il appuie sur échap en pleine partie, il peut choisir de sauvegarder le jeu ou de reprendre la partie.

1.5.2 Utilisation de la SDL

La bibliothèque SDL était essentielle pour notre projet. En effet la gestion du clavier et l'affichage vidéo permettent au joueur d'avoir une meilleure expérience que sur une version avec un terminal. Néanmoins les procédures et fonctions ont toutes été pensées au début sans SDL, ce n'est que pendant le développement et l'écriture du code sur Pascal que nous avons rajouté les éléments liés à la SDL dans notre code.

2 Apport du Projet

2.1 Apprentissage du LaTeX

Afin de réaliser notre rapport, nous avons suivi les directives données par les professeurs, à savoir utiliser le LaTeX.

2.1.1 Découverte de ce langage

Site Overleaf Nous avons fait le choix pour ce projet d'utiliser le site/application Overleaf afin de pouvoir modifier notre rapport simultanément et travailler ensemble. De plus, Overleaf permet de compiler le code directement afin d'avoir un aperçu sur la moitié de l'écran, ce qui est extrêmement pratique lorsque l'on commence à coder en LaTeX car il n'est pas aisé de se représenter le rendu. Enfin, ce site fonctionne par projet, il est donc facile de partager tout le projet à son groupe et de travailler avec les mêmes fichiers indispensables tels que des images.

Avantage du LaTeX Le LaTeX a été quasiment indispensable pour la rédaction de notre cahier des charges et du rapport. Effectivement, nous avons dû insérer du code, qu'il s'agisse de pseudo-code ou de pascal, ce qui aurait été bien plus complexe avec un éditeur comme Word ou Libre office. Le LaTeX permet également des rendus de qualité et il est important pour nous de découvrir ce langage.

Les bases du LaTeX Pour la plupart des membres du groupe, nous n'avions que très peu ou jamais codé en LaTeX. Ainsi, la découverte réelle de ce langage a été réalisée lors de l'élaboration du cahier des charges. Ce dernier étant assez rapide et condensé, il a permis de découvrir doucement les bases du LaTeX.

2.1.2 Approfondissement

Besoins supplémentaires Après avoir acquis les bases indispensables à la création d'un document telles que l'ajout de titres, sections, sous-sections, paragraphes ; nous avons dû apprendre à insérer des images, des algorithmes, des parties de code. L'essentiel de cet apprentissage s'est fait grâce aux explications que l'on peut trouver sur le net. En outre, rédiger un rapport sur un seul document LaTeX n'aurait pas été lisible et compréhensible, d'autant plus que nous l'avons rédigé à cinq. Organiser la rédaction du rapport s'est alors révélé très utile. Nous avons décomposé le rapport en plusieurs sections et avons fait les imports de packages dans un autre document. Cela a permis d'apporter beaucoup de lisibilité et de ne pas modifier involontairement les parties du code déjà réalisées.

Difficultés rencontrées Nous avons pu utiliser un modèle de rapport en LaTeX fourni par les professeurs. Celui-ci nous a beaucoup aidés notamment pour la structure du rapport. Cependant, nous avons tout de même rencontré des difficultés comme pour importer des images, notamment l'analyse descendante qui a une taille importante, mais aussi l'insertion de code. Pour réussir à surpasser la difficulté de ce dernier point, certains membres du groupe ont appris à utiliser le package minted.

De plus, nous avons rencontré de nombreuses difficultés lors de l'implémentation de notre jeu, notamment, avec l'utilisation de la SDL. En effet, dans un soucis de réalisme, pour que notre jeu soit le plus agréable possible à jouer, la gestion de l'affichage et des délais des objets de notre jeu était la partie la plus compliquée à implémenter, particulièrement lors des éboulements. De plus, il y avait beaucoup de petits détails à prendre en compte qui ne paraissaient pas très importants individuellement, mais qui, additionnés, transforment l'expérience de jeu, par exemple, le curseur quand on appuie sur échap en pleine partie, ou la gestion du chrono et des pauses quand on est dans le menu.

2.2 Travail en groupe

2.2.1 Découverte et utilisation de Gitlab

L'utilisation de Gitlab nous a été imposé par nos professeurs. Nous n'avions jamais utilisé Gitlab et les débuts sur ce logiciel étaient un peu compliqués, notamment pour comprendre le fonctionnement des différentes commandes. Malgré ces quelques complications, nous avons pu apprendre, à force d'utiliser les commandes, à comprendre leurs fonctions.

2.2.2 Mise en accord sur les fonctionnalités

Pour les différentes fonctionnalités attendues, chacun a proposé ses idées et nous nous sommes mis d'accord pour savoir si c'était réalisable et si cela rentrait dans les fonctionnalités obligatoires ou facultatives.

2.2.3 Répartition du travail en fonction des aptitudes

Concernant la répartition du travail au sein du groupe, nous nous sommes réparti les différentes tâches en fonction des compétences de chaque membre du groupe. Les missions principales de ce projet ont été la rédaction du cahier des charges, de l'analyse descendante et du rapport, mais aussi l'écriture du code dans le langage de programmation demandé.

2.2.4 Limitations et perspectives

Tout d'abord, l'aspect à distance de notre projet a rendu très difficile le travail de groupe et nous avons dû nous organiser en ligne sur Discord et Messenger. Ensuite, le suivi des séances avec le professeur était tout le temps à distance et nous n'avons pas eu de véritable retour sur notre avancée sur le projet. Enfin, Concernant les perspectives, nous avions établi plusieurs fonctionnalités facultatives dans notre cahier des charges, mais que, faute de temps, nous n'avons pas mis dans notre rapport. C'est pourquoi nous voudrions les reprendre par la suite pour les intégrer dans notre Boulder Dash.

3 Conclusion

Pour conclure, ce projet d'I4-2 nous a permis de développer de nombreuses compétences et fut une expérience enrichissante pour l'ensemble du groupe. Néanmoins, nous avons rencontré plusieurs difficultés au cours de notre projet. En effet, le travail à distance empêchait une bonne cohésion d'équipe et cela a été difficile de se coordonner au début du projet. Par la suite, pour le code, certains avaient déjà coder en SDL, c'est pourquoi ils ont aidé les personnes qui avaient plus de mal avec cette bibliothèque. Au cours de ce projet, nous avons eu l'occasion de développer notre expérience dans le travail de groupe qui est une des compétences majeures d'un ingénieur aujourd'hui. De plus, nous avons renforcé nos connaissances et compétences en programmation avec l'utilisation du langage de programmation SDL et Pascal. L'utilisation des bibliothèques SDL fut une première approche pour certains et pour d'autres a permis l'approfondissement de leurs compétences. Enfin l'utilisation de nouveaux outils tels que Gitlab ou encore l'écriture du rapport en LaTeXa permis à l'ensemble du groupe la maîtrise d'outils s'avérant nécessaire pour la suite de notre cursus.

Sources

— Photo de la page de garde : `http://emultest.free.fr/screenshot/nes/boulderdash1.png`