Rapport de Soutenance Final

Belloni Corentin Courtemanche Jessy
Corcione Arnaud GIRARD Ethan
NGOUESSY BOUSSAMBA Daniel

21 mai 2023

Table des matières

1 Introduction

1.1 Présentation du projet

L'objectif de ce projet est de développer un jeu compétitif pour deux joueurs, qui consiste à taper le plus rapidement possible les suites de mots générées par l'application. Ce jeu vise à améliorer la rapidité et la précision de frappe des joueurs tout en offrant une expérience de jeu interactive et amusante.

Pour atteindre cet objectif, nous avons choisi de programmer en langage C et d'utiliser des bibliothèques et des outils appropriés pour créer une interface utilisateur attrayante et une expérience de jeu fluide. Le jeu sera doté d'un générateur de texte aléatoire pour créer des suites de mots pour les joueurs. Les joueurs devront taper les mots correctement et rapidement pour gagner des points.

Nous prévoyons d'implémenter des fonctionnalités pour suivre les statistiques des joueurs, tels que le nombre de victoires et le temps de réponse moyen. Nous nous efforçons de créer une expérience de jeu dynamique et engageante pour les joueurs, avec un niveau de difficulté croissant au fur et à mesure que le jeu progresse.

1.2 Présentation des membres

1.2.1 Arnaud Corcione

Depuis le début de ma jeunesse, grâce à mon père , j'ai été bercé dans le monde de l'informatique, ce qui m'a permis de m'intéresser à ce monde étrange. A mon entrée à Epita, j'ai eu une révélation pour le code, qui maintenant ne quitte pas mon quotidien. J'ai également pu réaliser des projets touchant à plusieurs langages. Ainsi, se creuser la tête pour apprendre et à comprendre une nouvelle technologie est passionnant pour moi.

1.2.2 Corentin Belloni

Je suis quelqu'un qui est passionné par l'informatique. J'aime découvrir de nouvelles technologies et mettre mes connaissances en pratique pour résoudre des problèmes complexes. Je suis également quelqu'un de joyeux, qui aime travailler en équipe et partager mes connaissances avec les autres. Je suis très investi dans mon travail et j'aime relever des défis. Je suis également très travailleur et je m'efforce toujours de donner le meilleur de moi-même.

1.2.3 Ethan Girard

Etant issu d'une terminale scientifique, j'ai été pris de passion par l'infor- matique en classe de première avec la spécialité "Numérique et sciences infor- matiques". Je suis quelqu'un qui aime l'informatique et le sport en général, je suis également passionné par les technologies et l'innovation. Je suis déterminé et engagé, prêt à me donner à fond pour créer un projet de A à Z.

1.2.4 Daniel Ngouessy

J'ai découvert cette passion qu'est l'informatique il y a quelques années et depuis, je m'efforce d'en apprendre toujours plus sur les dernières technologies et tendances dans ce domaine. Je suis également dynamique et motivé, ce qui me permet de m'investir pleinement dans tous les projets auxquels je participe. Je suis convaincu que mon enthousiasme et mon dévouement seront un atout pour réussir ce projet d'école.

1.2.5 Jessy Courtemanche

Depuis jeune, mes frères m'ont inculqué la passion du jeu vidéo, puis de fil en aiguille je me suis intéressé au domaine de l'informatique et chacun de mes centres d'intérêts me stimulaient d'avantage que les anciens. Il y a eu le hard- ware, le gaming que cela soit entre copains ou en mode compétition (l'eSport), mais aussi le software. La programmation m'a toujours particulièrement intéressé car c'était une énigme pour moi, et surtout je savais que tôt ou tard j'allais pouvoir dévelop- per mes compétences techniques dans ce domaine. A mon entrée à EPITA, j'avais seulement vaguement pratiqué le langage C pour utiliser Arduino ou bien des logiciels comme Flowcode permettant de contrôler des microcontrôleurs inclus dans des systèmes plus importants. Au-delà de ce que nous étudions en Sciences de l'Ingénieur, j'ai acheté un kit Arduino qui m'a permis de découvrir les bases du C et de la programmation en général. Mais tout ceci sans vraiment comprendre la logique camouflée derrière de tels outils. Je pense que notre projet va me faire découvrir de nouvelles choses. La programmation est pour moi une source constante d'apprentissage, le domaine étant tellement vaste.

2 Avancement du projet

2.1 Client

Dans ce jeu, les joueurs s'affrontent en tapant des phrases au clavier plus rapidement que leurs adversaires. Je suis chargé de mettre en place la connexion entre le client et le serveur. La première étape pour établir cette connexion est d'ouvrir un socket côté serveur. Nous utilisons la fonction socket() pour créer un socket et nous la connectons à une adresse IP et un port spécifiques. Ensuite, nous entrons dans une boucle infinie pour écouter les connexions entrantes. Lorsqu'un client se connecte, nous créons un nouveau socket côté serveur et acceptons la connexion du client. A ce stade, le serveur est prêt à communiquer avec le client.

2.1.1 Communication

La communication entre le client et le serveur se fait via des messages. Nous avons défini un protocole de communication qui définit la structure de ces messages. Le client envoie des messages au serveur pour informer de son état de jeu et pour synchroniser les statistiques avec le serveur. Le serveur envoie des messages au client pour transmettre des informations sur le jeu, comme le début d'une nouvelle manche ou la fin du jeu.

La communication entre le client et le serveur est l'un des éléments clés de notre jeu. Nous avons donc défini un protocole de communication bien précis pour permettre une transmission efficace des données entre les deux parties.

Tout d'abord, nous avons choisi d'utiliser le protocole TCP/IP pour notre jeu. Ce protocole offre une communication fiable entre les deux parties en s'assurant que tous les messages envoyés sont reçus par le destinataire dans l'ordre dans lequel ils ont été envoyés. Cela est crucial

pour assurer une expérience de jeu fluide et sans interruption.

Ensuite, nous avons défini la structure de nos messages en utilisant des structures de données en C. Chaque message est composé d'un entête contenant des informations sur le type de message, la longueur des données et autres informations de contrôle. Les données elles-mêmes peuvent être de différents types, tels que des chaînes de caractères pour les phrases à taper ou des entiers pour les statistiques de jeu. Lorsqu'un message doit être envoyé, nous appelons une fonction de sérialisation qui prend en entrée la structure de données à convertir et retourne un tableau de bytes qui peut être envoyé au serveur.

La sérialisation de données est une technique qui permet de convertir des données complexes en une forme qui peut être stockée ou transférée plus facilement. Dans le contexte de notre jeu DigitDash, nous utilisons la sérialisation de données pour convertir des structures de données en messages qui peuvent être envoyés au serveur.

Le client envoie des messages au serveur pour informer ce dernier de son état de jeu. Par exemple, lorsqu'un joueur termine une manche, le client envoie un message au serveur contenant les statistiques de jeu, telles que le nombre de phrases tapées, le nombre d'erreurs, etc. Le serveur stocke ensuite ces données dans une base de données pour permettre une analyse ultérieure. Lorsque le message est reçu par le serveur, nous utilisons une fonction de désérialisation pour convertir les données binaires en une structure de données que nous pouvons manipuler dans notre code serveur.

Le serveur envoie également des messages au client pour transmettre des informations sur le jeu, comme le début d'une nouvelle manche ou la fin du jeu. Par exemple, lorsque tous les joueurs sont prêts à commencer une nouvelle manche, le serveur envoie un message à tous les clients pour les informer que la manche commence. De même, lorsque

le temps imparti pour le jeu est écoulé, le serveur envoie un message à tous les clients pour les informer que le jeu est terminé.

Pour la synchronisation des statistiques, nous avons mis en place une structure de données côté client qui stocke les statistiques du joueur, telles que le nombre de victoires, le score, le nombre d'erreurs, etc. Lorsque le joueur termine une manche, nous envoyons ces statistiques au serveur via un message. Le serveur stocke ensuite ces données dans une base de données pour permettre de les consulter ultérieurement.

2.1.2 Synchronisation

Comme je l'ai mentionné précédemment, la synchronisation des statistiques est une étape importante de notre jeu DigitDash. Nous avons donc mis en place une structure de données côté client pour stocker les statistiques de jeu du joueur. Cette structure est mise à jour en temps réel pendant que le joueur joue.

De plus outre la synchronisation Client-Serveur, il a fallu également synchroniser les différents éléments du jeu afin d'optimiser l'expérience de jeu. Nous nous sommes donc chargés de la création d'un simple menu terminal reliant chacune des parties sur lesquelles nous avons travaillées. Nous avons donc créé une interface simple et intuitive qui permet aux joueurs de choisir facilement le type de partie qu'ils souhaitent jouer. Les joueurs peuvent se connecter soit à une partie en ligne, soit à un mode d'entraînement hors-ligne, soit à un chat pour discuter avec d'autres joueurs. Cette fonctionnalité est essentielle pour offrir une expérience de jeu fluide et agréable aux joueurs.

En somme, l'établissement de la connexion entre client et serveur en langage C est essentiel pour assurer une communication efficace entre les joueurs et le serveur. Grâce à notre protocole de communication bien défini et à notre système de synchronisation des statistiques, nous espérons à terme créer un jeu amusant et compétitif.

2.1.3 Chat

Un grand avancement a été fait au niveau du chat désormais le code permet de créer un chat en ligne pour une communication de type "user-to-user" (U2U) entre deux utilisateurs. Il utilise des sockets pour la communication entre les deux utilisateurs, ainsi que la bibliothèque de threads "pthread.h" pour gérer les tâches simultanées.

La fonction principale du programme est appelée "u2u". Cette fonction établit une connexion avec le serveur en utilisant une adresse IP et un port spécifiques, puis crée deux threads pour gérer les messages entrants et sortants.

La première fonction, "send_message", est utilisée pour envoyer des messages entre les utilisateurs. Elle demande d'abord à l'utilisateur de saisir son nom et utilise une boucle while pour lire le message entré par l'utilisateur à partir de la console. Le message est ensuite stocké dans une structure de données appelée "Chat_info" et envoyé au serveur via le socket.

La deuxième fonction, "receive_message", est utilisée pour recevoir les messages envoyés par l'autre utilisateur. Cette fonction utilise une boucle while pour lire les messages envoyés par l'autre utilisateur à partir du socket et les afficher sur la console.

Enfin, la fonction "u2u" crée deux threads pour gérer les messages entrants et sortants. Ces threads sont créés en utilisant la fonction "pthread_create" et sont ensuite joints à la fin de la fonction "u2u" à l'aide de "pthread_join". La fonction "u2u" se termine lorsque les threads sont joints.

2.2 Serialization et deserialization

2.2.1 Serialization

En effet, l'utilisation d'une structure générique est une pratique courante en sérialisation, car cela permet de traiter différents types d'objets ou de structures de données de manière polymorphe. En utilisant un pointeur vers une structure générique, on peut facilement parcourir et extraire les champs de données de n'importe quelle structure. Cela simplifie considérablement le processus de sérialisation, car il n'est pas nécessaire de créer une fonction de sérialisation différente pour chaque type de structure.

Cependant, il est important de noter que la sérialisation présente également des limites et des défis. De plus, la sérialisation peut également affecter les performances du système, en particulier lorsque les objets sérialisés sont volumineux ou doivent être envoyés à travers un réseau lent ou peu fiable. Il est donc important de bien comprendre les avantages et les inconvénients de la sérialisation, ainsi que de prendre en compte les aspects de performance et de sécurité lors de la mise en œuvre de cette méthode. Avec une conception et une implémentation appropriées, la sérialisation peut être un outil puissant pour faciliter l'échange de données dans diverses applications.

2.2.2 Deserialization

La désérialisation est une étape cruciale dans la communication entre des systèmes informatiques. Elle permet de convertir une représentation linéaire d'un objet ou d'une structure de données, généralement sous forme de chaîne de caractères ou de flux d'octets, en une forme utilisable par un langage de programmation. Contrairement à la sérialisation, qui consiste à transformer une structure de données en une représentation linéaire, la désérialisation nécessite une analyse de cette représentation linéaire pour reconstruire la structure de données initiale.

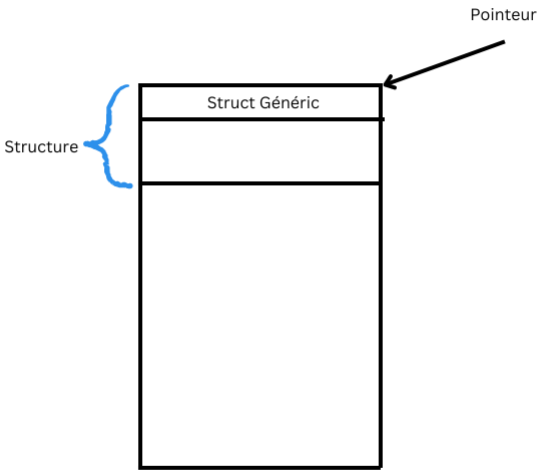


FIGURE 1 – image de la Stack avec représentation d'une structure.

Pour effectuer la désérialisation, on utilise généralement un principe de structure générique. Cela permet de traiter des objets ou des structures de données de manière polymorphe, c'est-à-dire en utilisant une même méthode pour tous les types d'objets possibles. En analysant la chaîne de caractères ou le flux d'octets, on peut identifier les différents champs qui composent l'objet ou la structure de données, et les placer dans une nouvelle structure qui sera utilisée par le langage de programmation.

2.3 Serveur

2.3.1 Serveur

Le serveur actuellement ressemble de loin à ce qu'il était à la premiere soutenance. L'utilisation des processus enfant a été abbandonner. Pour etre remplacer par l'utilisation des threads.

Les threads sont des unités de base de l'exécution parallèle au sein d'un processus. Ils permettent à un programme d'exécuter plusieurs tâches simultanément en partageant la mémoire et les ressources du processus parent. Contrairement aux processus, qui sont des entités indépendantes qui s'exécutent en parallèle avec leur propre espace mémoire, les threads partagent le même espace mémoire et les mêmes ressources avec le processus parent.

L'utilisation de threads dans le serveur présente en effet plusieurs avantages par rapport aux processus enfants. Tout d'abord, les threads sont plus légers que les processus et consomment moins de ressources système. Ils permettent également de partager plus facilement la mémoire et les ressources avec le processus parent, ce qui peut améliorer les performances globales du système.

De plus, les threads permettent une exécution simultanée des tâches au sein du même processus, ce qui est particulièrement utile dans les systèmes multijoueurs. En utilisant des threads pour chaque joueur, le serveur peut traiter plusieurs demandes simultanément, ce qui permet de réduire le temps de latence et d'améliorer la réactivité du système.

Enfin, la communication entre les threads est simplifiée par rapport aux processus enfants. Comme les threads partagent la même mémoire, il est plus facile d'échanger des informations entre eux. Dans le cas du serveur multijoueur, cela signifie que les informations reçues d'un joueur peuvent être directement transmises à l'adversaire corres-

pondant sans nécessiter de processus supplémentaires.
En résumé, l'utilisation de threads dans le serveur multijoueur permet d'améliorer les performances globales du système, de réduire le temps de latence et de simplifier la communication entre les joueurs. Cependant, il est important de mettre en place des mesures de sécurité appropriées pour minimiser les risques de conflits de mémoire ou d'autres problèmes liés à l'exécution simultanée des threads.

2.3.2 À l'avenir

Pour la prochaine soutenance, le serveur sera fini, et opérationel.

2.4 Générateur de texte

2.4.1 Introduction

Dans ce projet, notre objectif était de créer des lignes de code pour que les joueurs puissent les taper. Pour y parvenir, nous avons développé un système qui peut générer un code à partir d'une base de données. L'une des tâches les plus importantes dans le traitement du langage est la détermination du sens d'un mot dans un contexte particulier.

Nous avons donc mit au point un algorithme en langage C qui peut déterminer le sens d'un mot dans un contexte particulier en utilisant un mot précédent. Dans la suite de ce rapport, nous allons expliquer en détail notre approche et comment elle fonctionne, ainsi que les algorithmes que nous avons utilisés pour développer notre solution. Nous allons également discuter des suggestions pour les améliorations futures

2.4.2 Commencement

Pour commencer, nous avons dû nous pencher sur la tâche complexe de créer une liaison entre un mot futur et un mot actuel. Nous

avons exploré différentes approches, notamment les chaînes de Markov. Cependant, nous avons rapidement réalisé que la modélisation des probabilités en C pour obtenir l'état suivant était une tâche ardue. En effet, la modélisation des probabilités conditionnelles peut devenir très complexe, en particulier lorsque le nombre de variables augmente. Nous avons donc reflechi et nous nous sommes dit que pour y arriver nous allons fonction- ner comme cela :

— obtenir une string via un une fichier
— créer une liste chaînée avec tous les mots comme valeur de la liste
— récupérer tous les index d'un mot recherché
— renvoyer le mot suivant grâce à aux indexs trouvés précédemment

2.4.3 Première partie

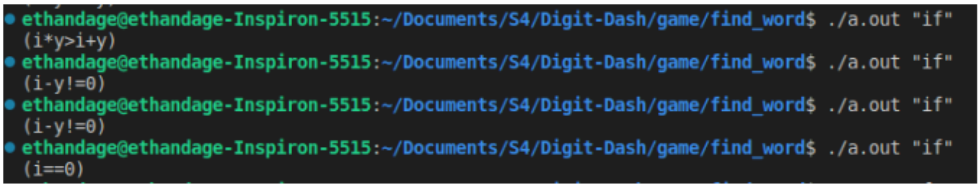
Pour ce qui est du programme en lui même, nous avons du mettre en place plusieurs fonction. La toute premiere qui est utilisé dans le programme est la fonction char *filetochar qui permet d'obtenir une chaîne de charactère via un fichier. Par la suite nous avons donc implémenté la fonction struct list *chartolist qui comme énoncé précedemment sert à créer une list qui va prendre comme valeur tous les mots de la base de donnée. Cette fonction renvoie par exemple ceci :

Ensuite, nous avons mis en place la fonction struct list *chartolist, qui est utilisée pour créer une liste chaînée contenant tous les mots de notre base de données.

Cette liste est ensuite utilisée pour déterminer le mot futur en utilisant la fonction struct inlist *create_index_list. Cette dernière fonction retourne une liste chaînée qui contient les index du mot donné. Cette

liste nous est utile pour déterminer le mot futur en récupérant un de ces index de manière aléa- toire, en lui ajoutant 1, et en récupérant ensuite le mot correspondant dans la liste du début (avec les mots).

Toutes nos fonctions nous donne ce résultat :



2.4.4 Seconde partie

En ce qui concerne l'avancement de cette partie, nous avons réalisé plusieurs actions pour améliorer notre base de données. Tout d'abord, nous avons augmenté la capacité de notre base de données en langage C pour mieux répondre aux besoins de nos utilisateurs. En outre, nous avons mis en place une nouvelle base de données en Python, offrant ainsi la possibilité aux utilisateurs d'améliorer leur dactylographie dans différents langages.

De plus, nous avons introduit la création d'un code que l'utilisateur peut coder lui-même. Ce code a été élaboré en utilisant les fonctions décrites précédemment, qui ont été répétées pour renforcer les compétences en dactylographie. Cela permet aux utilisateurs de personnaliser leur expérience de dactylographie en créant leur propre code, adapté à leurs besoins spécifiques.

En résumé, ces mesures ont permis d'améliorer la qualité de notre plateforme de dactylographie et de répondre aux besoins variés des utilisateurs en matière de langages et de personnalisation de leur expérience de dactylographie.

```
ethandage@ethandage-Inspiron-5515:~/Documents/S4/Digit-Dash/game/find_word$ ./markov
int len = (resultat + '0';
 decimal /= 2;
 }
 if (n<2)
 return 1;
}

void count_words(char *text, int len = (resultat + '0';
 decimal /= 2;
 }
 if (n<2)
 return 1;
}
```

FIGURE 2 – Exemple de nos fonctions en action

Il nous reste quand même certain problèmes à résoudre comme le montre la photo suivante. De temps en temps il a une répétition dans le fichier.

```
int chaine_vers_entier(char chaine[])
{
 int i + (binary[i]-'0');
 }
 }
}

int chaine_vers_entier(char chaine[])
{
 int i + (binary[i]-'0');
 }
 }
}
```

2.5 Affichage du jeu

L'affichage est un élément crucial pour tout jeu, y compris ceux joués sur le terminal de commande. Bien que les jeux en ligne de commande ne disposent pas des graphismes et des animations sophistiqués des jeux modernes, ils peuvent offrir une expérience de jeu immersive et engageante grâce à un affichage bien conçu.

Un bon affichage peut rendre le jeu facile à lire, avec des couleurs et des contrastes bien choisis pour une meilleure lisibilité. Il peut également être utilisé pour afficher des informations importantes telles que le score, les niveaux, les vies restantes et les instructions de jeu.

De plus, l'affichage peut aider à créer une ambiance appropriée pour le jeu, en utilisant des couleurs et des motifs adaptés à l'univers du jeu. L'affichage est donc un élément essentiel de tout jeu, qu'il soit joué sur un ordinateur de bureau sophistiqué ou sur un terminal de commande minimaliste.

2.5.1 Code couleur

Lorsque le joueur lance une partie, le jeu commence par afficher sur le terminal le code à écrire. Ce code est affiché en gris clair, presque transparent, pour que le joueur puisse voir ce qu'il doit écrire tout en distinguant clairement le code original. Le code à écrire prend la forme d'un motif, qui peut être une série de lignes de code ou d'instructions. Le niveau approprié est chargé grâce au retour du générateur de texte expliqué précédement, le programme et écrit le code dans la console du joueur, ligne par ligne.

Cependant, l'indentation (la disposition du code en blocs) est ignorée dans cette version de développement, afin de faciliter les tests. Le joueur doit ensuite écrire le code demandé, caractère par caractère, et le programme vérifie chaque caractère saisi pour s'assurer que le joueur

suit correctement le modèle affiché.

Les caractères correctement saisis sont affichés en vert pour indiquer leur validité, tandis que les caractères incorrects sont affichés en rouge. Si aucun caractère n'a été saisi, le programme affiche le caractère correspondant dans le modèle en gris. Le joueur doit ainsi saisir tout le code correctement pour terminer le niveau et passer au suivant.

2.5.2 Ancien fonctionnement

Auparavant, pour afficher le niveau à l'utilisateur, le programme utilisait une méthode simple consistant à lire dans un fichier contenant les différents niveaux et à les afficher ligne par ligne sur le terminal. De plus, pour stocker les entrées de l'utilisateur et afficher les couleurs correspondantes, le programme stockait les entrées dans un fichier temporaire afin de garder en mémoire l'intégralité des inputs. Cette méthode de stockage des entrées n'était pas très efficace et adaptée au fonctionnement du programme, car elle nécessitait une grande quantité de stockage de données et ralentissait l'exécution du jeu.

De plus, elle rendait la gestion des entrées plus complexe, car le programme devait parcourir le fichier temporaire à chaque nouvel input pour vérifier la validité des caractères saisis et mettre à jour les couleurs d'affichage. Par conséquent, cette méthode a été remplacée pour simplifier la gestion des entrées et améliorer les performances du jeu.

2.5.3 Nouveaux fonctionnement

Dans la nouvelle version du jeu, le générateur de texte renvoie maintenant un tableau de chaînes de caractères. Cette amélioration a grandement simplifié l'affichage du jeu, car il suffit de parcourir le tableau et d'afficher une chaîne de caractères par ligne pour afficher le code à écrire.

En outre, le stockage des entrées de l'utilisateur a également été amélioré. Le même fonctionnement que pour le générateur de texte est utilisé, à savoir un tableau de chaînes de caractères alloué en mémoire de la même taille que celui renvoyé par le générateur de texte. Chaque chaîne de caractères à l'intérieur du tableau est initialisée avec un caractère nul. Ainsi, à chaque caractère entré par l'utilisateur, le caractère nul est décalé d'une position pour laisser la place au nouveau caractère saisi. Cette méthode permet de stocker efficacement les entrées de l'utilisateur tout en minimisant l'utilisation de la mémoire et en améliorant les performances du jeu.

2.5.4 Création des statistiques du joueur

Les statistiques occupent une place importante dans le jeu, car elles permettent au joueur de mesurer sa performance et de suivre sa progression au fil du temps. Ces statistiques sont affichées pendant et après la partie. Le but du jeu est de réaliser le meilleur score possible, que ce soit en comparaison avec un autre joueur ou même par rapport à son propre meilleur score.

Pour cela, plusieurs statistiques sont disponibles, telles que le nombre de victoires, la fréquence de frappe ou même le taux d'entrées correctes. Ces statistiques fournissent des informations clés pour aider les joueurs à comprendre leur niveau de compétence et à identifier les domaines où ils peuvent améliorer leur performance. De plus, elles peuvent également être utilisées pour suivre les progrès des joueurs et leur permettre de fixer des objectifs pour atteindre des scores plus élevés.

2.5.5 Initiation au TUI

Pour créer une interface utilisateur avancée pour notre jeu sur terminal, nous allons utiliser la bibliothèque open source ncurses. Cette bibliothèque fournit des fonctionnalités pour contrôler et manipuler des éléments graphiques tels que des fenêtres, des curseurs, des couleurs et

des zones de texte dans une console de terminal.

En utilisant ncurses, nous serons en mesure de créer des interfaces utilisateur plus avancées pour notre jeu sur terminal. Nous pourrons utiliser des fonctionnalités telles que des boutons, des listes déroulantes, des barres de progression et des menus déroulants pour améliorer l'expérience utilisateur.

La bibliothèque nous permettra également de mettre en évidence des éléments importants et de rendre le jeu plus attrayant en utilisant des couleurs.

Finalement, la bibliothèque ncurses est un outil précieux pour nous afin de créer des applications de terminal qui cherchent à améliorer l'interface utilisateur de leurs applications. Grâce à ses fonctionnalités avancées, nous pourrons créer une interface utilisateur plus intuitive pour notre jeu sur terminal et améliorer l'expérience de l'utilisateur.

Nous avons commencé à utiliser la bibliothèque ncurses pour cette soutenance, mais l'interface n'est qu'à ses débuts. Nous prévoyons de la compléter pour la prochaine soutenance.

2.6 Site web

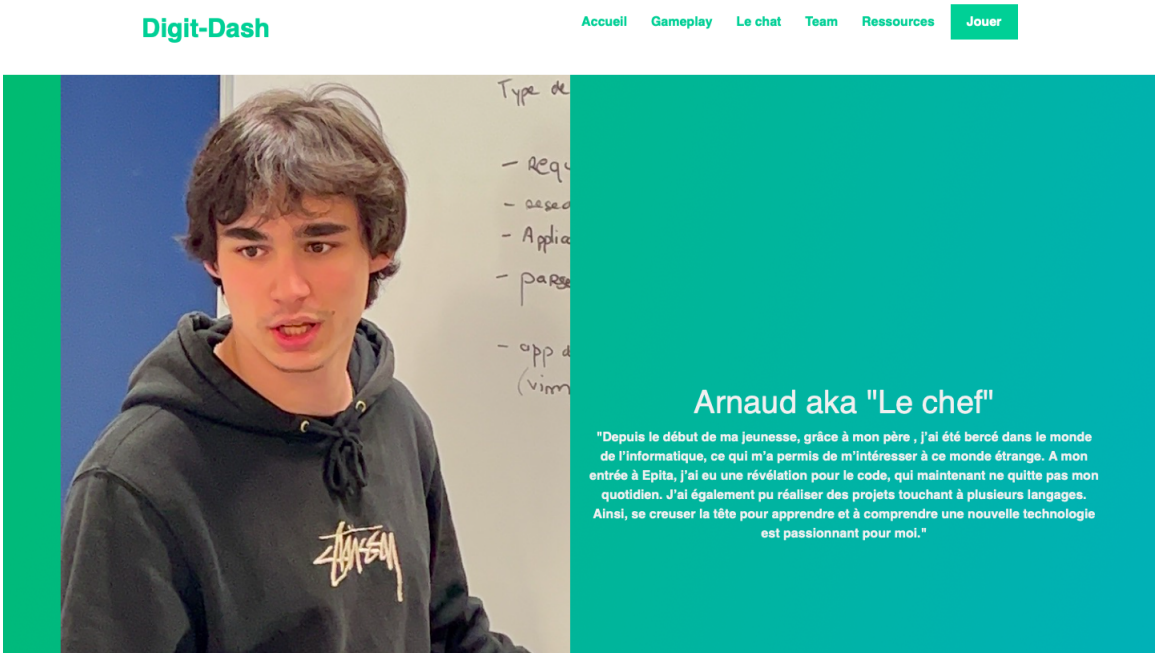
Pour cette présentation, nous avons apporté quelques améliorations graphiques mineures au site web. Nous avons notamment changé la couleur sur conseils des ASM en utilisant un dégradé entre le vert clair et le bleu turquoise. Cette modification de couleur ajoute un peu plus de dynamisme et d'attrait visuel à la page, ce qui peut aider à attirer l'attention des visiteurs.

Une autre amélioration légère a été apportée en augmentant l'espacement entre les éléments de la première page. Cela a permis de rendre

la page plus aérée, ce qui rend la lecture plus agréable et moins oppressante.

En outre, nous avons ajouté une nouvelle page "Team" en utilisant le framework Bootstrap. Cette page offre une interface conviviale qui présente à plusieurs reprises la photo d'un membre du groupe, son prénom et sa description. Cette page est très utile pour présenter l'équipe et pour donner aux visiteurs une idée de qui se cache derrière le site web.

En somme, les améliorations apportées au site web permettent d'améliorer l'expérience utilisateur en ajoutant un peu plus d'attrait visuel et en offrant une présentation plus conviviale de l'équipe. Ces changements mineurs peuvent aider à renforcer l'engagement des visiteurs envers le site web.



2.7 Conclusion

En conclusion, notre jeu avance correctement, l'ensemble des éléments atteint pour cette soutenance le sont ! Pour la prochaine soutenance, l'objectif final sera de finir correctement du jeu.