Sorbonne Université - MU4IN505

Rapport du projet de Conception et Pratique d'Algorithmes

Julien Fang 28605540

Gabriel Hernandez-Yepes 28707797

3 mai 2024

Abstract

Un clone de Brick Breaker, un jeu vidéo classique en 2D. Ce jeu met en scène une plateforme controlée par le joueur pour rebondir une balle et détruire des briques en haut de l'écran. Celui-ci met en avant les collisions, la physique des objets, la gestion de l'état du jeu ainsi que des éléments spéciaux tels que les briques bonus, offrant ainsi une expérience de jeu immersive et stimulante.

Ce rapport met en avant les étapes réalisations d'un jeu en 2D.

Mots-clés : collision, plateforme mobile, balle ,brique

Table des matières

1 Partie Introductive

1.1 Presentation du projet

Membres de l'quipe :

Gabriel HERNANDEZ-YEPES

Julien FANG

Objectif du projet :

Ce projet a pour but de développer un clone du jeu vidéo **Brick Breaker** de type "old school" en réutilisant le canvas d'un TME.

Technologies Utilisées :

Pour le développement de notre application web, nous avons opté pour une combinaison de technologies modernes et populaires, notamment React, Node.js et TypeScript. Bien que nous n'ayons pas eu d'expérience préalable avec TypeScript, nous avons choisi cette combinaison car nous avons repris un TME qui utilisait ces technologies. En nous appuyant sur cette base déjà établie, nous avons rapidement constaté les avantages de cette stack, largement reconnue dans l'industrie pour sa robustesse, sa flexibilité et sa capacité à faciliter le développement d'applications web évolutives et performantes.

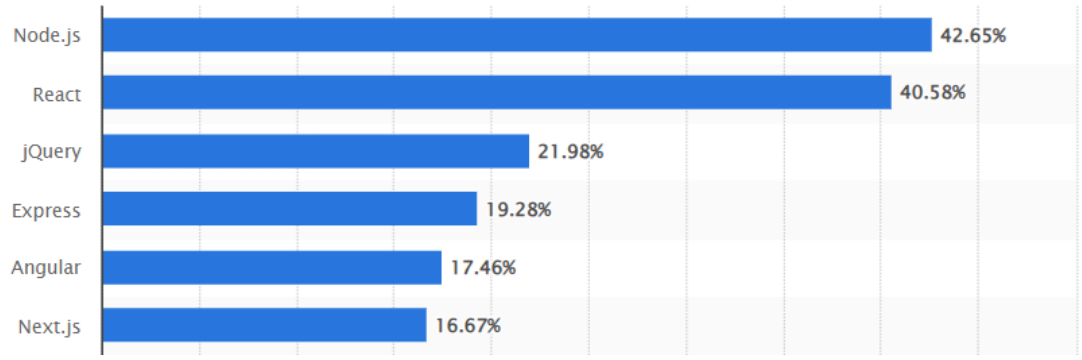

FIGURE 1 – Frameworks web les plus utilisés par les développeurs dans le monde en 2023

	Utilisation en % par tous les sites internet	Utilisation en % par tous les logiciels de programmation
JavaScript	64.96%	68.62%
HTML/CSS	56.07%	55.9%
Python	48.24%	41.53%
SQL	47.08%	50.73%
Java	35.35%	34.51%
Node.js	33.91%	36.19%
TypeScript	30.19%	36.42%
C#	27.86%	29.81%
Bash/Shell	27.13%	27.63%
C++	24.31%	19.94%
PHP	21.98%	22.54%
C	21.01%	16.64%

FIGURE 2 – La place des langages de programmation dans le monde en 2022

Nous pouvons constater que nos technologies utilisées occupent une part importante sur le marché dans le domaine du web, avec Node.js et React occupant respectivement les première et deuxième places. Quant à TypeScript, bien qu'il ne soit pas aussi populaire que ses deux confrères, il a néanmoins occupé une part notable, démontrant ainsi son importance dans le domaine du développement web.

1.2 Organisation

Le développement du jeu s'est déroulé de manière collaborative, avec une approche itérative visant à améliorer progressivement les fonctionnalités principales.

Phase 1 du développement :
Julien : Les briques, la plateforme mobile, les balles (incluant les collisions) .
Gabriel : Le background, la musique, les couleurs et les textures des objets.

Phase 2 du développement :

Julien : Les bonus, les différents niveaux de jeux et un trait pour visualiser la direction de la balle.
Gabriel : La création de la map et les différentes interfaces graphiques.

Lorsque des problèmes se sont présentés au cours du développement, une communication régulière a été maintenue entre nous afin d'identifier le(s) problème(s), analyser et trouver une solution adaptée en commun.

1.3 Analyse fonctionnelle

Lorsque que l'utilisateur démarre l'appli, il se trouve sur un écran avec "jouer" et "sélection des niveaux" :

1. Lorsque qu'il appuie sur jouer, il se retrouve sur une "map" quadrillé :
   (a) Sur une des cases se trouve un personnage représentant le joueur que l'utilisateur peut déplacer.
   (b) Entrer dans un niveau : Représentation d'une porte d'entrée d'un niveau demandé. Lorsque le personnage se trouve au même coordonnée que la case du niveau, il entre dans le jeu.
   (c) Lien entre le joueur et la case du niveau : Représentation du plus court chemin entre les deux éléments à l'aide d'un trait rouge.
2. Selection d'un niveau : Le joueur se retrouve sur une autre map mais avec le niveau sélectionné.

Dans notre version de Brick Breaker :

1. Contrôle de la plateforme mobile : Les joueurs pourront utiliser les touches directionnelles pour déplacer la plateforme horizontalement et faire rebondir la balle pour détruire les briques.
2. Intégration de bonus : Des bonus spéciaux aléatoires apparaîtront à intervalles réguliers, tels que l'élargissement de la plateforme, doubler le nombre de balle ou augmenter une vie.
3. Niveaux de difficulté : Nous proposons une sélection de niveaux, allant du niveau 1 à 4.
4. Affichage du score : Indication du nombre de briques détruites par le joueur, gagnant 1 point pour chaque brique détruite.
5. Dynamique du jeu :
   (a) La vitesse de la balle augmentera progressivement à chaque brique détruite, débutant à une vitesse de 1, sachant que chaque brique a un nombre de vies aléatoires allant de 1 à 3.

   (b) Les briques peuvent descendre d'un cran après un certain nombre de briques détruites.
   (c) Lors du lancement initial de la balle, les joueurs auront la possibilité de pivoter la direction de la balle avant de la lancer, afin d'offrir plusieurs opportunités stratégiques.
6. Cas défaite : Si la vie du joueur descend à 0, un écran lui propose soit de recommencer soit d'aller à l'écran principal.
7. Cas victoire : S'il ne reste plus de briques, l'utilisateur peut soit aller au niveau suivant soit aller à l'écran principal.
8. Musique : il y a un bouton qui permet de lancer une musique et un bouton qui permet de l'arreter
9. Briques incassables : les briques de couleur métalisé sont indestructibles

2 Partie Technique

2.1 Architecture générale de l'application

Couche client : Nous avons les fichiers correspondant aux différentes interfaces graphiques. Nous avons un fichier contenant le canvas de Brick Breaker et un autre pour l'affichage de ce jeu. Nous avons de plus un fichier contenant le canvas et les fonctions d'affichage de la carte du jeu.

Couche serveur : Nous avons un fichier contenant les algorithmes nécessaire au jeu Brick Breaker. Mais aussi un fichier similaire pour les algorithmes lié à la map.

2.2 Structures Principales

Dans notre application, nous utilisons les structures de données suivantes pour répresenter les différents éléments du jeu :

- **Coord** : Représente les coordonnées d'un objet. Les propriétés sont **x** et **y** qui correspondent à la position, respectivement, les propriétés **dx** et **dy** indique le deplacement de l'objet.
- **Ball** : Représentation d'une balle dans le jeu, comprenant ses coordonnées **Coord**, son nombre de vie **life**, une propriété **invincible** (objet non destructible) ainsi que la vitesse de la balle **speed**.
- **Size** : Représente la taille d'un objet avec les attributs **height** et **width**.
- **BonusBall** : Représentation d'un objet bonus dans le jeu, contenant les mêmes propriétés qu'une balle mais en rajoutant **type** pour connaître le bonus contenu dans la balle.
- **Pad** : Représentation de la plateforme mobile à l'aide des flèches du clavier, avec les attributs **Coord** pour connaitre sa position, son niveau de vie **life** et une propriété **invincible**.
- **Square** : Représentation d'une brique dans le jeu, avec les attributs **Coord** pour connaitre sa position, son niveau de vie **life**,variable entre 1 à 3 et une propriété **invincible**.
- **State** : Représente l'état global de notre jeu. Les propriétés sont :
  - **pos** : Un tableau contenant toutes les informations des **Ball** présent dans le jeu.
  - **sq** : Un tableau contenant des **Square** présents dans la partie.
  - **pad** : Contient toutes les informations concernant la plateforme **Pad**.
  - **size** : La taille de zone de jeu. Cette propriété est de type Size, contenant

les dimensions de la zone en termes de hauteur et de largeur.
- **endOfGame** : Un indicateur booléen indiquant si la partie est terminée.
- **end** : Un indicateur booléen indiquant si le jeu est terminé.
- **life** : Le niveau de vie actuel du joueur dans la partie.
- **ref** : Une référence React utilisée pour interagir avec un élément du DOM.
- **score** : Le score actuel du joueur dans la partie.
- **start** : Un indicateur booléen indiquant si la balle a été lancé.
- **directionShotX** : Représente la composante X de la direction de tir initial de la balle. Cette valeur sera utile pour afficher un trait visuel indiquant la direction dans laquelle la balle sera lancée lorsque le jeu démarre.
- **directionShotX** : Représente la composante Y de la direction de tir initial de la balle. Cette valeur sera utile pour afficher un trait visuel indiquant la direction dans laquelle la balle sera lancée lorsque le jeu démarre.
- **cptBall** : Un compteur, pour pouvoir envoyer une balle bonus à intervalle régulier.
- **bonus** : Un tableau contenant des **BonusBall** qui sont en pleine chute durant la partie.
- **downSquare** : Un compteur, pour faire descendre d'un cran tous les briques après un certain nombre de briques détruites.
- **win** : Un indicateur booléen indiquant si le joueur a terminé le niveau.

Pour la carte du jeux, nous avons d'autres structures :
- **GridSquare** : Correspond à une case du quadrillage de la map.
- **StateMap** : Représente l'état de la carte, elle possède comme attribut une taille de type **Size** et un attribut chara de type **GridSquare** qui corespond au personnage.

2.3 Fonctions Implémentés

Les fonctions utilisées afin de réaliser **Brick Breaker** :
- **iterate_pad** : Prenant une taille **Size**, une balle **Ball**, la plateforme **Pad** et une vitesse de type **number**. Elle permet la gestion de collision entre la plateforme mobile et les balles de la partie. Lors d'une collision entre les deux éléments, la balle change de direction.
- **iterate_bonus** : Cette méthode permet de verifier la collision entre une balle bonus et la plateforme.
- **collideSq** : Cette fonction prend deux **Coord** et détecte seulement s'il y a

une collision. Elle sera utilisée pour vérifier les collisions entre une brique et une balle, ainsi que entre la plateforme et une balle bonus.

- **collideBoingSq** : Cette fonction permet de gérer la collision entre une balle et une brique. Elle simule le rebond d'une balle sur une brique.
- **doubleBall** : Une balle bonus **BonusBall**, permet de doubler le nombre de balle dans la partie. Chaque balle est doublée avec une direction aléatoire.
- **upgradeSizePad** : Une balle bonus **BonusBall**, permet de rallonger la taille de la plateforme d'une valeur constante.
- **addLife** : Une balle bonus **BonusBall**, permet d'ajouter une vie au joueur dans sa partie.
- **handleKeyMovePad** : Cette fonction gère les mouvements du pad et le tir de la balle en fonction des événements de touche du clavier. Modifier l'angle de tir se fait avec les touches**Q** et **D**. Elle réalise le déplacement du pad horizontalement et l'angle de tir de la balle initial. La plateforme accélère lorsque le joueur maintient enfoncé l'une des deux directions.

Les fonctions utilisées afin de réaliser la map :

- **calculateDistance** : Calcule la distance entre des coordonnés et une case. Elle est utilisée dans **findNearestGridSquare**.
- **findNearestGridSquare** : Cette fonction permet de trouver la case la plus proche d'une autre case. Elle est utilisée pour savoir si la future case du personnage est un obstacle. Si c'est le cas, le personnage ne va pas dans cette case.
- **handleKeyMove** : Permet de bouger le personnage avec les touches Z pour se diriger vers le haut, Q pour aller à gauche, D à droite et S pour se diriger vers le bas.
- **findOptimalPath** : Permet de trouver le chemin le plus court entre le personnage et la case où se trouve le niveau. Dans cette fonction se trouve des sous fonctions :
  - **calculateDistance** : Permet de savoir la distance entre 2 cases du quadrillage
  - **findPath** : Cette fonction est utilisé pour trouver le chemin optimal en suivant l'algorithme A*.
    Voici une explication détaillée de chaque partie de findPath :
    1. Paramètres :
       (a) start : La case de départ dans la grille.
       (b) end : La case d'arrivée dans la grille.
    2. Initialisation des listes :

   (a) openList : Une liste contenant les cases à explorer, initialement contenant uniquement la case de départ.
   (b) closedList : Une liste contenant les cases déjà explorées, initialement vide.
3. Boucle principale :
   (a) La boucle s'exécute tant que openList n'est pas vide.
   (b) À chaque itération, on sélectionne la case de openList avec le coût total le plus bas (f).
4. Traitement de la case courante :
   (a) La case courante est déplacée de openList à closedList.
   (b) Si la case courante est la case d'arrivée, le chemin est reconstruit et retourné.
5. Calcul des cases voisines : Les cases voisines (haut, bas, gauche, droite) de la case courante sont déterminées.
6. Parcours des cases voisines :
   (a) Pour chaque case voisine, on vérifie si elle existe et n'est pas un obstacle.
   (b) Si la case n'est pas déjà dans closedList, on calcule son coût de déplacement g.
   (c) Si la case n'est pas dans openList, on l'ajoute avec son coût g, son coût heuristique h, le coût total f, et la case parent.
   (d) Si la case est déjà dans openList, mais le nouveau coût g est inférieur à l'ancien, on met à jour les informations de la case dans openList.
7. Retour : Si aucun chemin optimal n'est trouvé, null est retourné.

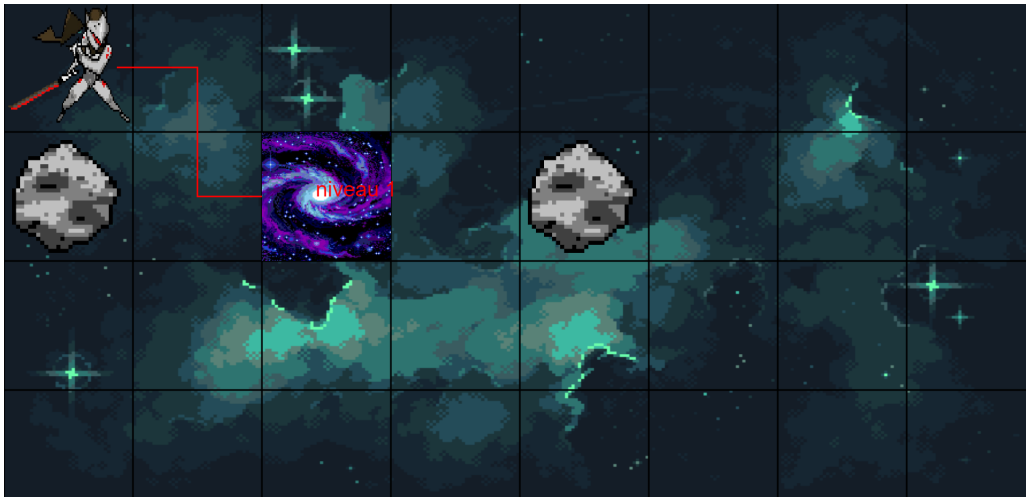

FIGURE 3 – La carte pour entrer dans le niveau 1

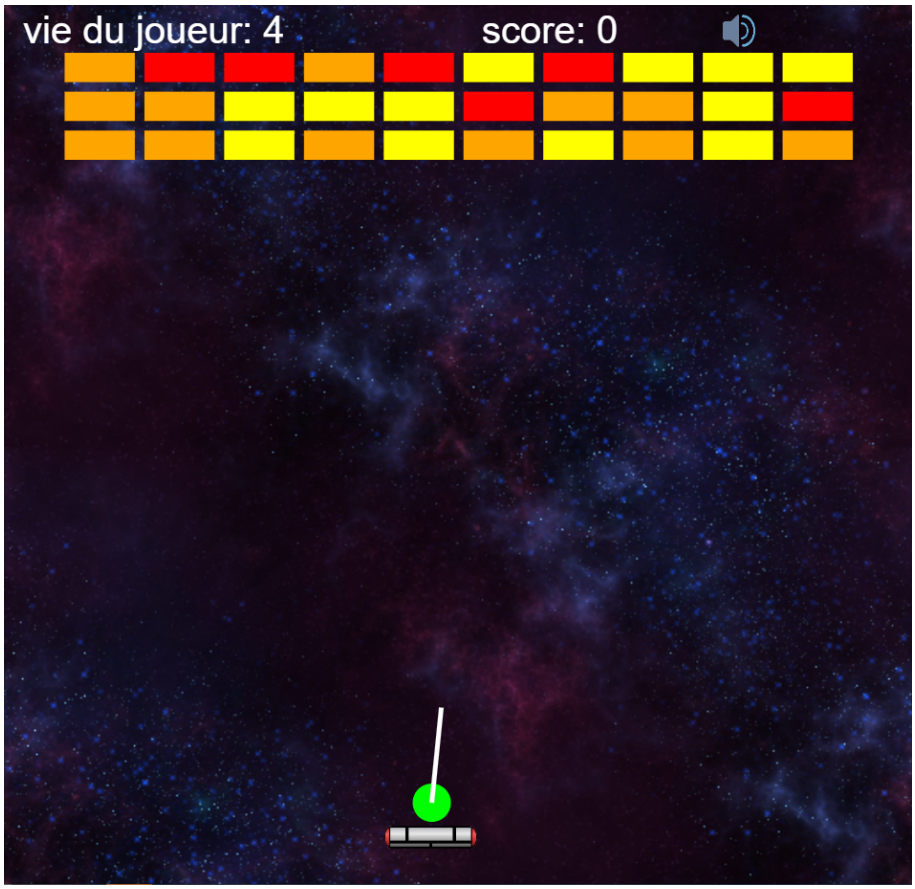

FIGURE 4 – Immersion dans le jeu, au niveau 1

3 Conclusion et Retour d'Expérience

3.1 Retrospective

Ce projet a été une opportunité précieuse pour développer des compétences essentielles pour les programmeurs, en particulier celles liées au travail d'équipe et à l'autonomie. La répartition des tâches et la collaboration pour surmonter les difficultés ont renforcé notre esprit d'équipe, tandis que la responsabilité individuelle sur des aspects spécifiques du développement a renforcé notre autonomie. Travailler sur un projet de jeu vidéo a également été extrêmement gratifiant, ajoutant une dimension ludique à notre expérience de développement.

Cependant, nous avons également rencontré des défis tout au long du processus. Certains algorithmes et fonctionnalités, tels que l'intégration de la musique, nous ont posé des difficultés. Dans notre cas, la musique fonctionnait partiellement sans que nous réussissions à comprendre exactement notre erreur, même si l'ajout de son devrait être plutôt simple. Nous avons résolu le problème en utilisant deux boutons, l'un pour démarrer la musique et le second pour l'arrêter.
De plus, certaines fonctionnalités, techniquement réalisables, nous ont demandé plus de temps que prévu pour les implémenter, tel que l'ajout d'un trait pour la gestion de trajectoire de la balle. Un des problèmes était dû à une succession de sommes de 0.1. Par exemple, si l'on additionne trois fois 0.1, on obtient 0.30000000000000004 plutôt que 0.3. Par conséquent, des tests d'égalité ne fonctionneront pas comme souhaité. Cela est due à la représentation binaire des nombres à virgule flottante.
Bien que nous ayons réussi à améliorer la fluidité de notre plateforme mobile, nous avons constaté des problèmes persistants de latence entre les entrées clavier et les réponses de la plateforme, ce qui réduit l'immersion du joueur.

3.2 Conclusion

Dans l'ensemble, le projet s'est révélé être une expérience enrichissante et fructueuse pour nous. Non seulement avons-nous réussi à atteindre nos objectifs, Ce projet nous a permis d'approfondir nos connaissances dans le domaine des technologies du web que ce soit en Typescript ou en React. En outre, ce projet nous a offert une opportunité précieuse de perfectionner nos compétences en algorithmie, tels que l'implémentation d'algorithmes comme celui de détection de collisions et l'algorithme A star. De plus, il nous a permis d'explorer notre créativité et de renforcer notre capacité à résoudre des problèmes.

Dans l'ensemble, notre expérience sur ce projet a été très enrichissante. Cependant, nous pensons qu'elle aurait pu être encore plus diversifiée et complète si nous

avions eu l'opportunité de travailler sur un jeu nécessitant une couche data ou d'explorer l'utilisation d'autres frameworks.

Annexe

1. https ://www.lafabriquedunet.fr/blog/barometre-cms-editeur-site-web/

2. https ://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/