Université de Paris 8

Projet Tuteurés 2023-2024

Pavages de Polyominos en JS

Damien GEOFFROY

Viet NGUYEN

Table des matières

Remerciements

Nous remercions Monsieur Nicolas JOUANDEAU pour son accompagnement et ses conseils qui nous ont permis de réaliser ce document et d'appréhender ce projet tuteuré. Nous exprimons également notre gratitude envers Monsieur Farès BELHADJ pour avoir mis à notre disposition son document[1], nous permettant ainsi de fournir un rendu professionnel.

[1] : *Modèle pour projet tuteuré ou rapport de stage*.

I. Introduction

Ce document PDF représente un état de l'art approfondi sur le sujet des pavages de polyominos ainsi qu'un rapport de fin de projet qui s'inscrit dans le cadre de notre projet tuteuré, mené au sein de notre cursus à l'Université Paris 8 Vincennes - Saint-Denis, pendant notre troisième année. Ce projet tuteuré se décline en deux phases distinctes réparties sur les deux semestres. La première phase consiste en une recherche et réflexion approfondie, aboutissant à la rédaction de la partie état de l'art. La seconde partie de notre projet implique la mise en pratique des connaissances acquises, avec la concrétisation du projet dans son ensemble.

Le sujet qui nous a été attribué est particulièrement captivant. Il nous offre l'opportunité d'explorer les domaines des mathématiques à travers l'étude des polyominos. Ce sont des formes géométriques, constituées de l'assemblage de carrés unitaires connectés entre eux, pouvant varier en taille allant d'une taille 1 à une limite indéterminée. Les polyominos ont gagné en popularité au cours du 20e siècle, tant en raison des problèmes mathématiques complexes qu'ils suscitent que grâce à la diffusion mondiale du jeu Tetris. Ce dernier, basé sur des pièces de polyominos tombant sur une grille de dimensions finies, a contribué à populariser ces formes simples. Le jeu consiste à compléter des lignes en largeur pour augmenter le score du joueur.

Ce projet nous offre une perspective unique en nous permettant d'explorer non seulement le domaine des mathématiques, grâce au pavage qui consiste en une disposition régulière d'éléments les uns à-côtés des autres afin d'obtenir une nouvelle forme. Dans le cadre de notre sujet, il s'agit de l'emboîtement des polyominos, mais également celui de l'informatique et de l'algorithmique. Nous mettrons en œuvre nos connaissances, notamment en utilisant JavaScript, un langage de programmation offrant des capacités graphiques avancées tout tout en restant flexible. Sa popularité et sa compatibilité avec les navigateurs web en font un outil idéal pour la visualisation des résultats des algorithmes de pavage. De plus, la syntaxe claire et la facilité de manipulation des objets dans JavaScript faciliteront le processus de conception et de mise en œuvre de notre programme.

Tout au long de ce document, nous allons plonger dans les aspects théoriques des pavages de polyominos, en préparation à la phase pratique de notre projet tuteuré dans laquelle nous exploiteront pleinement les fonctionnalités de JavaScript pour concrétiser nos idées et visualiser les résultats obtenus.

II. Exploration des Types de Polyomino en 2D

Dans ce chapitre, nous explorerons les divers polyominos existants, offrant ainsi une compréhension approfondie de l'enjeu de notre projet tuteuré. Pour ce faire, nous reverrons les fondements même d'un polyomino, cela nous amènera à voir leurs caractéristiques essentielles, tout en élargissant notre analyse pour examiner les nombreuses variantes qui existent pour certains polyominos spécifiques. Cette démarche nous permettra de comprendre les défis et les problèmes liés à ces derniers, nous permettant d'avoir un point de vue global des possibilités et des complexités que nous devrons aborder dans la réalisation de notre projet.

1. Introduction aux Polyomino

Avant de poursuivre notre étude, revenons sur la nature même d'un polyomino.

Le terme "polyomino" a été introduit en 1953 par Solomon W. Golomb[2], un ingénieur et mathématicien célèbre pour ses contributions variées, dont l'invention des "échecodames" [3] et son empreinte laissée grâce à ses travaux sur les polyominos.

L'appellation "polyomino" est dérivée du mot "domino", et le préfixe "poly" peut être substitué par le nombre de parties composant le polyomino en grec. Un polyomino, tel que décrit précédemment, se compose d'un ensemble de carrés connectés entre eux par les bords, le différenciant des autres polyformes tels que ceux formés de répétition pyramides ou des cubes.

Dans le cadre de notre projet tuteuré, nous focalisons notre attention sur les polyominos en raison de leur prédominance et de leur utilisation évidente dans le contexte d'un plan (leurs équivalents tridimensionnels étant les polycubes). En effet, travailler avec des polyominos présente des avantages pratiques, notamment sur le plan 2D. Par rapport aux polyiamondes, qui ont une forme de base triangulaire, les polyominos offrent une plus grande simplicité, réduisant les complications liées aux espaces perdus engendrés par cette forme particulière. D'un autre côté, les formes plus rectangulaires s'avèrent être des équivalents aux polyominos, rendant leur étude moins captivante dans le cadre de notre projet tuteuré.

---

[2] : Pour plus d'informations sur cet ingénieur visitez, cette page : Solomon W. Golomb (2024, 9 juin). Dans Wikipedia : https://fr.wikipedia.org/wiki/Solomon_W._Golomb

[3] : Une fusion entre les échecs et les dames voir les règles ici : Chesskers — the classics with a twist ! (s. d.). https://chesskers.lol/rules

2. Description du pavage

2.1. Règle de classification des Polyominos

Lorsqu'il s'agit de trier les polyominos, une règle de classification en trois catégories est communément utilisée.
Tout d'abord, nous avons les polyominos dits "libres". Il s'agit de l'ensemble des polyominos différents, séparés en rotation, réflexion ou translation. Par conséquent, les tétraminos de type "Z" et "S" et “N” (nommés en fonction de leurs formes similaires aux lettres de l’alphabet) sont distingués en fonction de leur réflexion et rotation respective.

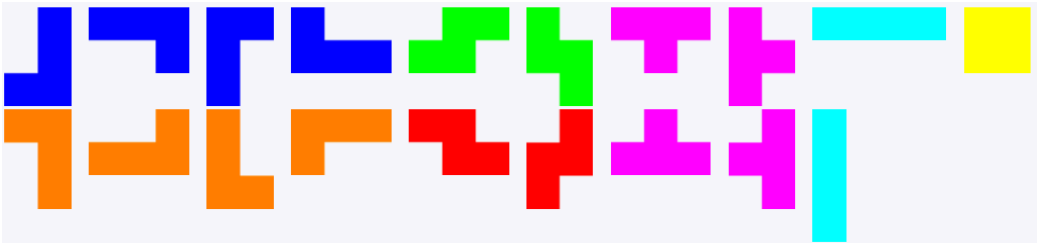


Figure 2.1: montrant les tetrominos de manière libre,
les formes ont été arrangées par nous (Tetromino. (2024, 9 février). Dans Wikipedia. https://en.wikipedia.org/wiki/Tetromino)

La seconde classification, plus restrictive, est appelée "unilatérale". Elle exclut la réflexion, ne conservant que les polyominos en fonction de leur rotation et de leur translation. Cette classification offre une vision plus restreinte mais parfois plus pertinente selon le contexte d'application.

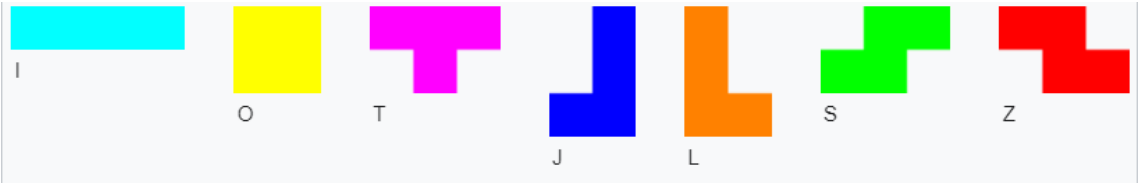


Figure 2.2 :montrant les tetrominos de manière unilatérale , les formes ont été réarrangées par nous (Tetromino. (2024, 9 février). Dans Wikipedia. https://en.wikipedia.org/wiki/Tetromino)

Enfin, la classification dite "fixée" se base uniquement sur la forme des polyominos, sans prendre en compte la réflexion, la rotation ou la translation. Cette approche, moins flexible, est très pratique pour le dénombrement des différentes formes, et permet ainsi une perspective unique sur la variété des polyominos.

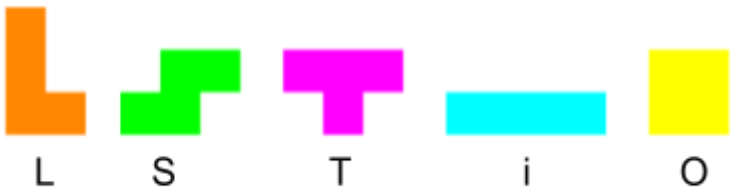


Figure 2.3 montrant les tetrominos de manière fixe, les formes ont été réarrangé par nous (Tetromino. (2024, 9 février). Dans Wikipedia. https://en.wikipedia.org/wiki/Tetromino)

La compréhension de ces différentes classifications est essentielle pour notre projet tuteuré, car elle influence directement la manière dont nous aborderons les différentes formes de polyominos dans notre exploration algorithmique.

2.2. Tableau des pièces de Polyominos

Pour une explication plus fluide et simplifié voici un tableau récapitulant les polyominos en fonction des différentes types de classifications [4]

	forme	total	avec trous	sans trous	unilatérale	fixe
1	monomino	1	0	1	1	1
2	domino	1	0	1	1	2
3	triomino	2	0	2	2	6
4	tetramino / quadriminos	5	0	5	7	19
5	pentamino	12	0	12	18	63
6	hexamino	35	0	35	60	216
7	heptamino	108	1	107	196	760
8	octamino	369	6	363	704	2725
9	nonominos/enneamino	1285	37	1248	2500	9910
10	decamino	4655	195	4460	9189	36 446
11	undecamino	17073	979	16094	33896	135 268
12	dodecamino	63 600	4 663	58 937	126 759	505 861
15	Pentédécominos	3 426 576	424 056	3 002 520		

Ces données riches, présentées dans le tableau, nous fournissent des informations essentielles, mais surtout elles mettent en évidence la présence de certains cas particuliers que nous allons évoquer.

[4] : Les informations viennent de cette page wikipedia (Polyomino. (2023, 8 novembre).Dans Wikipedia. https://fr.wikipedia.org/wiki/Polyomino) ainsi que cette page : (Gerard Villemin, “polyominos,introduction aux dominos, pentominos et autres.” (s. d.). http://villemin.gerard.free.fr/Puzzle/minoPoly.htm) ainsi que (Polyform tiling. (s. d.). https://www.polyomino.org.uk/mathematics/polyform-tiling/)

2.3. Cas particuliers

La présence quasi infinie de polyominos rend impossible une étude exhaustive. Cependant, en examinant certains polyominos déjà analysés par des chercheurs, des cas particuliers se dégagent. Notamment, à partir des heptaminos, nous constatons la présence de pièces comportant des trous. Ces configurations, par définition, empêchent la réalisation d'un pavage parfait et, par extension, intensifient la complexité du pavage.

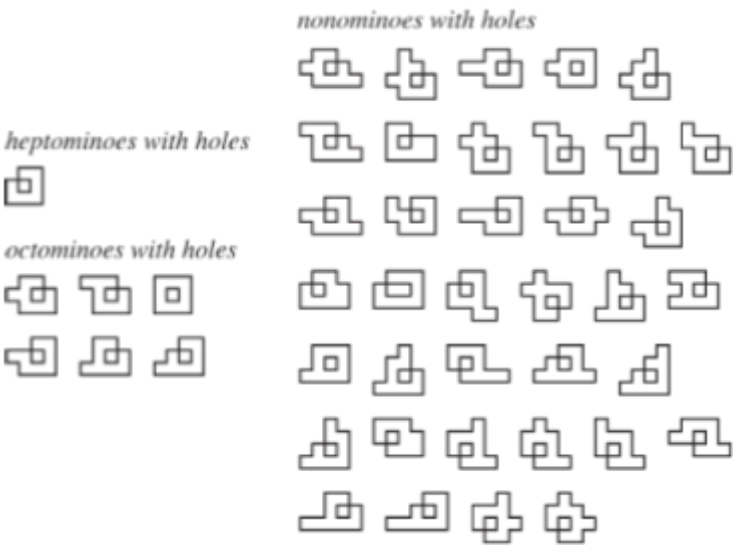


Figure 2.4 montrant les polyominos de 7 à 9 carrés avec des trous (Weisstein, Eric W. "Polyomino." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/Polyomino.html)

Une singularité supplémentaire réside dans le fait que certains types de pavages sont purement et simplement impossibles à remplir et à construire en fonction des formes choisies. Cette démonstration peut être réalisée par des calculs mathématiques (par exemple un jeu de plateau de dame composé de 64 cases ne peut être rempli avec des triominos car 64 n'est pas divisible par 3[5]) ou simplement par une logique claire, comme illustré dans la figure suivante :

[5] : Solomon W. Golomb créateur du terme "polyominos" présente certaines impossibilité lié au plateau du jeu de dame dans son livre : Solomon W. Golomb , Polyominoes. (s. d.). Princeton University Press. https://press.princeton.edu/books/paperback/9780691024448/polyominoes

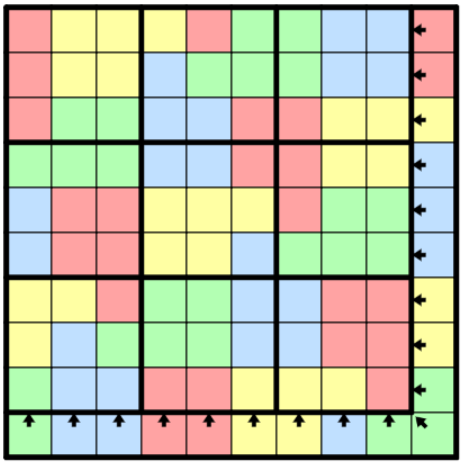


Figure 2.5, montrant l'impossibilité de remplir un rectangle de 9 avec des polyomino différent du mono, trio et nonominos ,(Tiling Pentomino Sandwich Sudoku — Rätselportal — Logic Masters Deutschland. (s. d.). https://logic-masters.de/Raetselportal/Raetsel/zeigen.php?chlang=en&id=00041R)

Par ailleurs, les polyominos de taille importante peuvent poser des défis en raison du nombre limité d'options disponibles pour les placés, complexifiant ainsi la recherche de solutions.

Un autre cas particulier intéressant concerne la présence de motifs dits auto-répliquant, créant ainsi des schémas répétitifs fascinants. Ces motifs offrent des perspectives uniques et ajoutent une dimension esthétique à l'étude des polyominos. Nous pouvons citer par exemple deux dominos coller entre-eux formant un carré de taille 2x2.

Ces cas particuliers enrichissent la compréhension des polyominos pour les pavages, et leur prise en compte peut considérablement influencer la complexité des problèmes et des algorithmes associés. Ils constituent des défis intéressants pour la résolution de problèmes mathématiques et informatiques liés aux polyominos et c'est en gardant ces nuances à l'esprit tout au long de notre analyse que nous serons en mesure de concrétiser le projet lors du second semestre.

III. Les Polycubes

Dans le chapitre précédent, nous avons exploré les polyominos en deux dimensions, découvrant leurs propriétés, classifications et leurs caractéristiques dans les pavages 2D. À présent, voyons leurs équivalents lorsque l'on ajoute une dimension supplémentaire à savoir les polycubes en 3D. Dans ce chapitre, nous verrons la définition des polycubes, leurs classifications, ainsi que leurs caractéristiques. En comparant avec les polyominos, nous mettrons en évidence les différences spécifiques aux pavages en trois dimensions.

1. Présentation des Polycubes

Comme nous l'avions mentionné précédemment, le terme 'polycubes' suit la même logique d'appellation que les polyominos. Ici, le préfixe 'poly' est remplacé par le nombre d'éléments, et le suffixe 'cube' lui y est ajouté. Les polycubes représentent l'équivalent tridimensionnel des polyominos. Contrairement à ces derniers, les polycubes sont formés par des cubes unitaires connectés les uns aux autres au lieu de carrés. Cette simple modification engendre une transformation significative, introduisant un nouvel axe et permettant une diversité bien plus grande de formations. Ainsi, l'ensemble des polycubes s'avère plus vaste que celui des polyominos, ouvrant la porte à une variété accrue dans la construction d'ensembles tridimensionnels.

2. Caractéristiques des Polycubes

Le nouveau plan ajouté permet une complexité bien plus importante, avec des formes pouvant être plus ou moins profondes. De nouvelles constructions, considérées comme impossibles, deviennent envisageables notamment grâce à la manipulation des connexions entre les cubes pour insérer plus de trous. Malgré tout tant qu'une face d'un cube est connectée à une autre face, la formation est considérée comme valide donc aucun cube ne peut être uniquement relié par une arête comme illustré dans ces exemples :

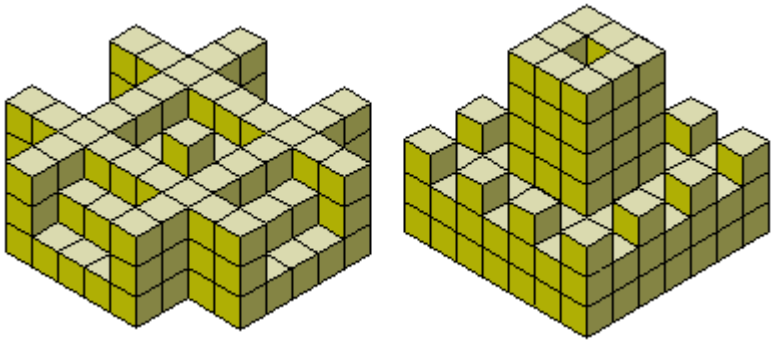


Figure 3.1: montrant des figures complexes de polycube (The Poly pages. (s. d.). http://recmath.org/PolyPages/PPF/index.htm?Polycubes.html)

Nous remarquons également que les règles de classification des polycubes sont identiques à celles des polyominos. Ainsi, nous avons des polycubes "fixes", "libres" et "unilatéraux".

Cependant, leurs points distinctifs sont dans leurs symétries qui sont plus nombreuses et potentiellement plus voyantes. Nous pouvons également voir que certains polycubes ne peuvent pas être retournés ou réfléchis comme les polyominos, et la plupart sont asymétriques. Voici une liste des polycubes avec leurs nombres respectifs[6] :

	forme	**Nombre de n-polycubes unilatéraux (les réflexions sont comptées comme distinctes)**	**Nombre de n-polycubes libres (les réflexions sont comptées ensemble)**
1	monocube	1	1
2	dicube	1	1
3	tricube	2	2
4	tetracube	8	7
5	pentacube	29	23
6	hexacube	166	112
7	heptacube	1023	607
8	octocube	6922	3811

3. Challenges et Complexités liés aux Polycubes

La connexion entre les cubes étant plus complexe que pour les polyominos, cela nécessite la création d'algorithmes capables de comprendre un espace tridimensionnel plus dynamique. Trouver la disposition la plus efficace tout en respectant les contraintes spécifiques des polycubes, telles que les limitations de taille, de forme ou de connectivité dynamique entre les cubes imposées par les polycubes eux-mêmes, ajoute une dose supplémentaire de complexité. Une approche algorithmique sophistiquée est donc requise, notamment pour éviter les collisions entre les formes et maximiser la comblage des trous pour un pavage parfait.

Malgré ces défis, les polycubes offrent des opportunités passionnantes dans la modélisation 3D de notre projet, augmentant la réflexion nécessaire à la résolution des problèmes de pavage.

---

[6] : les informations viennent de cette page : Polycube. (2024b, juin 12). Wikipedia. https://en.wikipedia.org/wiki/Polycube

IV . Utilisation de JavaScript dans la réalisation du pavage de polyominos

Dans ce chapitre nous présenterons Javascript dans le contexte de notre projet. Après une brève introduction à JavaScript, nous allons voir ses capacités graphiques et sa flexibilité algorithmique, soulignant son rôle essentiel dans la visualisation et la manipulation de données pour ce type de problème.Pour cela, nous discutons du rôle de JavaScript dans notre projet tuteuré de pavage de polyominos, en mettant en évidence ses avantages et limitations. Enfin , nous allons voir des exemples concrets de mise en œuvre en 2D et 3D, illustrant comment JavaScript peut être employé pour résoudre des défis spécifiques de pavage.

1. Introduction à JavaScript

1.1. Présentation brève

JavaScript, souvent abrégé en JS (à ne pas confondre avec Java), est un langage de programmation créé en 1995. Fréquemment utilisé avec HTML et CSS, JavaScript joue un rôle essentiel dans la création de pages web dynamiques en permettant la manipulation des données. En tant que langage orienté objet, il utilise des prototypes, offrant une structure flexible de variables et de méthodes réutilisables à travers le code. Cette approche facilite la transmission des données et permet des modifications tant du côté client que du côté serveur.

Du côté client, JavaScript intervient en modifiant le Document Object Model (DOM), la structure de la page HTML, en ajoutant des éléments de manière dynamique. Du côté serveur, il peut être employé pour manipuler des bases de données, par exemple.

1.2. Pertinence de JavaScript pour le projet

Dans le cadre de notre projet tuteuré, l'utilisation du langage JavaScript est imposée, mais nous considérons ce choix comme particulièrement approprié. Cette pertinence réside notamment dans la gestion graphique offerte par JavaScript, que nous détaillerons plus ultérieurement. La facilité d'utilisation de bibliothèques pour obtenir des résultats en 2D et 3D est un atout majeur de JavaScript, une capacité que tous les langages ne possèdent pas, ou qui nécessite des approches plus complexes. De plus, couplé à cela, JavaScript se distingue par sa capacité à gérer des algorithmes complexes, offrant ainsi une solution complète et adaptée à notre projet de pavage de polyominos.

2. Capacités Graphiques et bibliothèques de JavaScript pour la Visualisation

Pour élargir les possibilités de manipulation graphique en JavaScript, l'utilisation de bibliothèques est essentielle. Ces bibliothèques, contiennent des ensembles de fonctions, simplifiant énormément la réalisation de nombreuses opérations car elles permettent d'accomplir des tâches plus facilement. Dans le cadre de notre projet tuteuré, nous devons nous baser sur la librairie Three.js et l'API Canvas pour réaliser les manipulations en 2D et en 3D.

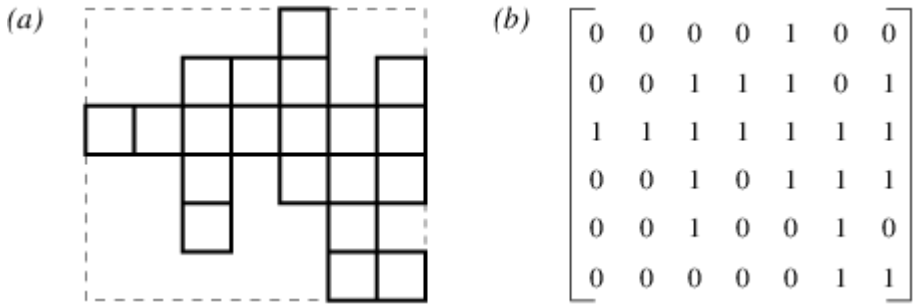


Figure 4.1: montrant une représentation textuelle (ici dans une matrice) des polyominos (Andrea Frosini . Article Research Gate : https://www.researchgate.net/figure/A-polyomino-and-its-representation-as-a-binary-picture-or-matrix_fig1_268167195)

Pour l'instant, concentrons-nous sur l'aspect 3D avec Three.js. Cette bibliothèque facilite la création d'objets graphiques en permettant la mise en place d'une scène. À partir de cette scène, il devient possible d'ajouter une caméra pour obtenir un point de vue défini, ainsi qu'une source lumineuse.

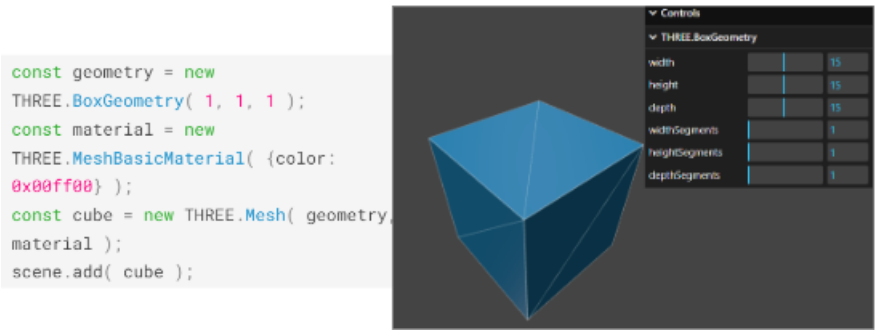


Figure 4.2 montrant la création d'un cube en Three.js (three.js docs. (s. d.). https://threejs.org/docs/#api/en/geometries/BoxGeometry)

Si la création d'un cube est relativement facile, la gestion dynamique de ces objets et leur création dynamique sont bien plus complexes. Three.js offre un ensemble d'outils pour relever ces défis, mais la maîtrise de ces fonctionnalités demande une compréhension approfondie.

Passons à l'aspect 2D, pour cela nous ferons usage de l'API Canvas permettant de dessiner et de manipuler des graphiques en 2D sur une page web, de manière dynamique si nécessaire. L'API Canvas met à disposition un objet de contexte 2D, obtenu grâce à la méthode *getContext('2d')*. Cela permet de dessiner diverses formes telles que des rectangles, des cercles et des lignes, en plus de permettre la modification de leurs couleurs.

Il est donc possible à partir de l'utilisation de ces bibliothèques de nous rapprocher de certains autres projets basés sur les polyominos pour notre projet tuteuré, comme illustré par les images ci-dessous.

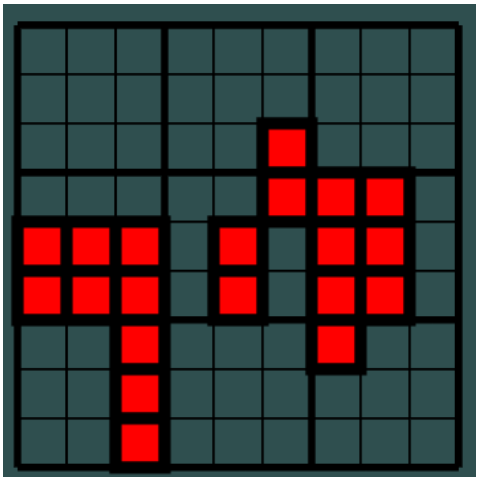


Figure 4.3: montrant un projet basé sur polyomino en Javascript (cependant il n'a pas le même objectif final) (Polyomino blocks. Oliver Merkel. https://omerkel.github.io/PolyominoBlocks/html5/src/index.html)

Comme nous pouvons le voir, ces outils offrent un ensemble diversifié pour aborder notre projet sous différents angles, que ce soit en explorant la complexité des formes en 3D ou en manipulant graphiquement des solutions de pavage en 2D.

V. Preparation au Développement

Les pavages de polyominos suscitent un vif intérêt dans le domaine des mathématiques et des jeux de logique. Trouver des configurations de polyominos pour compléter un pavage sans laisser de vides représente un défi qui requiert une créativité mathématique, notamment en raison de la nature exponentielle des calculs des solveurs qui augmente avec le nombre de pièces.

Il est important de distinguer les pavages de polyominos des solveurs de polyominos. Ces derniers sont des outils ou des logiciels permettant de résoudre divers problèmes liés aux polyominos, y compris des tâches autres que l'agencement sur une grille 2D. Par exemple, un solveur de polyominos peut servir à créer des énigmes logiques ou à effectuer des calculs en rapport avec les polyominos, sans nécessairement impliquer la création d'un pavage.

Pour travailler efficacement et proposer un pavage ainsi que des solveurs, nous avons utilisé GitHub pour partager et modifier les fichiers en ligne. Discord a également été un outil précieux pour communiquer facilement, en plus des moments en classe. Cela nous a permis de comprendre que même avec une préparation préalable, il était nécessaire de continuer à se renseigner et à ajuster notre projet en cours de route. Nos recherches ont inclus des exemples de projets similaires et des documents en ligne pour affiner nos idées. Nous avons ainsi imposé des limites à notre projet, comme la focalisation sur les pièces sans trous et l'utilisation de polyominos prédéfinis plutôt que de créer un générateur de pièces. Notre principale contrainte était l'interactivité, permettant à l'utilisateur de jouer avec les polyominos et d'utiliser des IA pour résoudre les grilles.

Ensuite, nous expliquerons ce que nous avons réalisé pour la partie 2D, puis ce que nous avons fait pour la partie 3D, en nous focalisant sur l'utilisation normale par un utilisateur.

VI. Développement de la Partie 2D

1. Arborescence générale de la partie 2D

L'arborescence du projet est composée des fichiers principaux "index.html", "style.css" et "main.js", ce dernier contenant les fonctions principales utiles pour d'autres fichiers. Un sous-dossier nommé js contient plusieurs fichiers JavaScript, chacun ayant des rôles spécifiques. Un autre sous-dossier, popup, gère les options de la toolbar en utilisant des fonctions exportées.

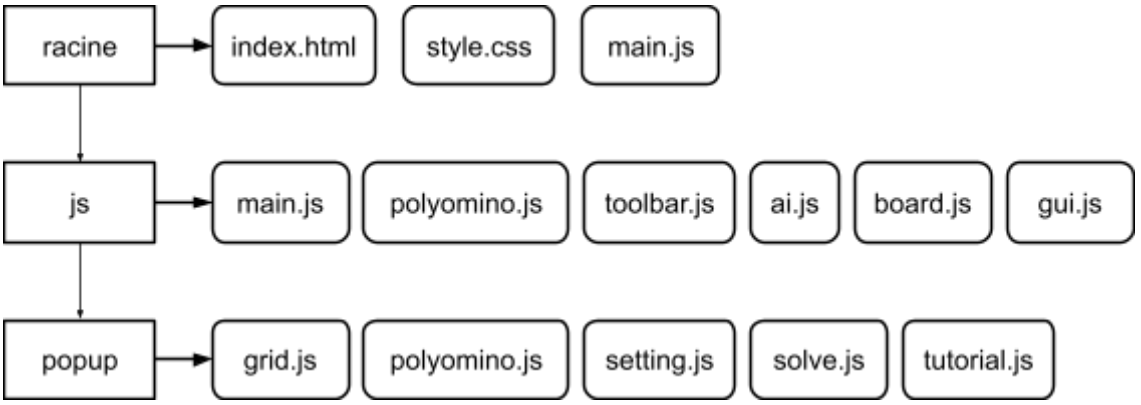


Figure 6.1: montrant l'arborescence de la partie 2D

2. Interface générale

L'interface générale est basée sur le fichier "index.html" qui constitue la structure de base de l'application web.

Il intègre différentes sections de l'interface utilisateur telles que :

- la grille de placement qui est la zone où les polyominos seront placés. Cette grille interactive permettra un placement facile et précis des pièces. En temps réel les modifications apportées par l'utilisateur ainsi que les cellules bloquées seront affichées, offrant une visualisation claire du pavage en cours de création.

- la Barre verticale Gauche (toolbar) qui occupe la longueur totale de l'écran, cette barre verticale sera dédiée aux différents menus, elle permettra à l'utilisateur de modifier la taille de la grille sur laquelle les polyominos seront placés. Des options supplémentaires y sont intégrées, telles que la sélection des polyominos, des différentes intelligences artificielles et d'autres paramètres personnalisables tels que la possibilité de bloquer des cellules tout ceci via des fenêtres pop-up.

C'est donc dans ce fichier HTML que la balise canvas de l'API est défini et où les polyominos seront dessinés en prenant le contexte 2D des fichiers et en utilisant JavaScript et CSS nécessaires eux sont inclus pour assurer une mise en page cohérente et plaisante.

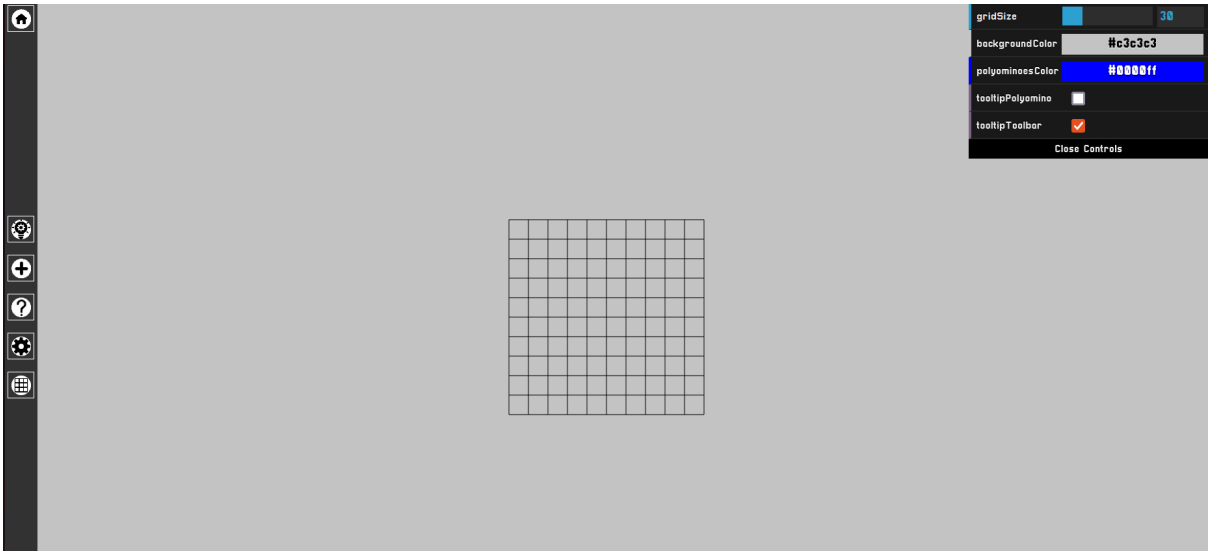
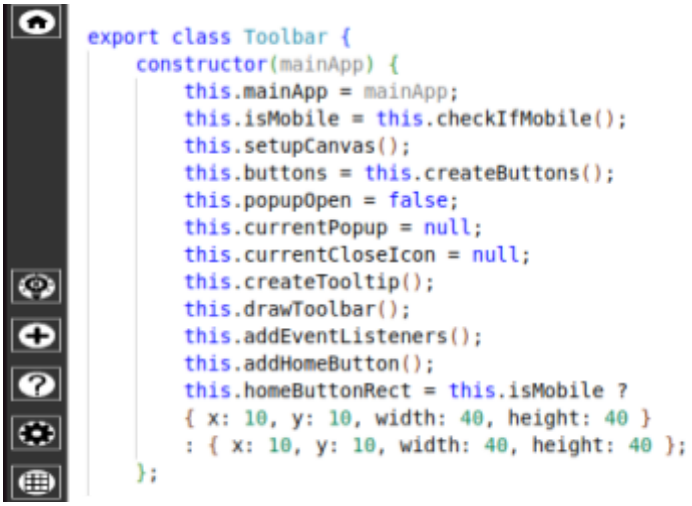


Figure 6.2: Interface globale de l'application

La conception de l'interface utilisateur pour la partie 2D de notre projet se veut intuitive et fonctionnelle, permettant une interaction aisée avec les polyominos.

3. Barre Latérale

La barre latérale (toolbar) offre une expérience utilisateur enrichissante en facilitant l'exploration et la compréhension des concepts de pavages de polyominos. Le fichier "toolbar.js" implémente les boutons de la barre d'outils qui lancent les actions correspondantes dans l'application, elle permet donc d'avoir des raccourcis pour des opérations courantes comme l'ajout de polyominos ou la modification de la grille.
Par exemple, le constructeur de la classe Toolbar initialise la barre, dessine les boutons et gère les clics. Des fonctions comme *createButtons()* démarrent les fichiers correspondants après avoir pressé le bon bouton , d'autre fonctions gère l'apparence de la barre elle même tel *drawToolbarVertical()* avec un *addEventListeners* qui gère l'interaction avec la barre elle-même.



```
export class Toolbar {
    constructor(mainApp) {
        this.mainApp = mainApp;
        this.isMobile = this.checkIfMobile();
        this.setupCanvas();
        this.buttons = this.createButtons();
        this.popupOpen = false;
        this.currentPopup = null;
        this.currentCloseIcon = null;
        this.createTooltip();
        this.drawToolbar();
        this.addEventListeners();
        this.addHomeButton();
        this.homeButtonRect = this.isMobile ?
        { x: 10, y: 10, width: 40, height: 40 }
        : { x: 10, y: 10, width: 40, height: 40 };
    };
```

Figure 6.3: Barre latérale à gauche et constructeur du fichier "toolbar.js" à droite

4. Création des Polyominos

La création des polyominos est gérée par deux fichiers nommés “polyomino.js”. Le premier, situé dans le dossier js, définit la classe Polyomino et ses méthodes associées, permettant de créer et manipuler des polyominos de différentes formes et tailles. Les méthodes incluent *rotate(), rotateLeft(), rotateRight()* pour la rotation, et *flip()* pour l'inversion. Ce fichier permet également de dessiner le polyomino choisi via le contexte 2D et en utilisant les fonctions liées au dessin tel que fillRect() et aussi strokeStyle() en précisant les coordonnées x et y des sous-icônes. Le second fichier, situé dans le sous-dossier “popup”, utilise les informations du premier fichier pour dessiner les polyominos dans la toolbar avec une taille réduite mais aussi en utilisant le dictionnaire SHAPES du “main.js” pour avoir les forme.

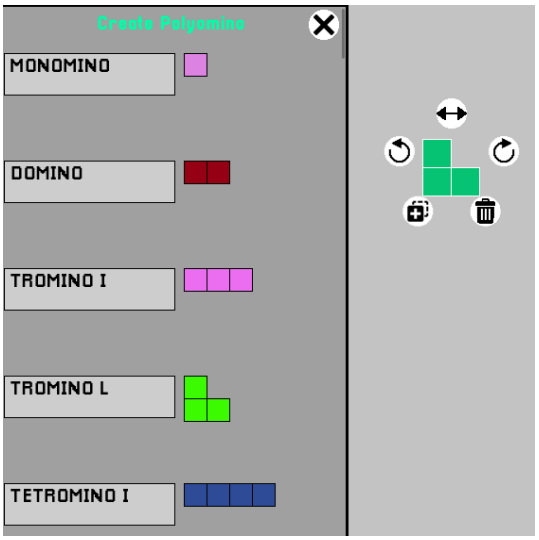


Figure 6.4: Menu création de Polyominos avec un Polyomino accompagné de ses fonctionnalités

5. Gestion de la grille

La gestion de la grille est réalisée par le fichier “board.js”, qui dessine la grille en fonction du nombre de colonnes et de lignes choisi par l'utilisateur de la même manière que les polyominos eux sont dessiner à partir du contexte. Ce fichier peut ajouter un polyomino à la grille après avoir vérifié que le placement est valide et qu’il ne chevauche pas d'autres polyominos ou sort des limites de la grille, et modifie l'état interne de la grille en conséquence. La fonction *clearBoard()* permet d'effacer tous les polyominos de la grille, réinitialisant son état à vide. Le fichier “grid.js”, appelé dans la toolbar, permet d'activer des fonctionnalités sur la grille en appelant des fonctions de “main.js” grâce à des icônes qui sont générées et mises en place comme par exemple créer des cellules noires sur lesquels les polyominos ne peuvent pas être placés ou encore inverser les cellules noires et blanches.

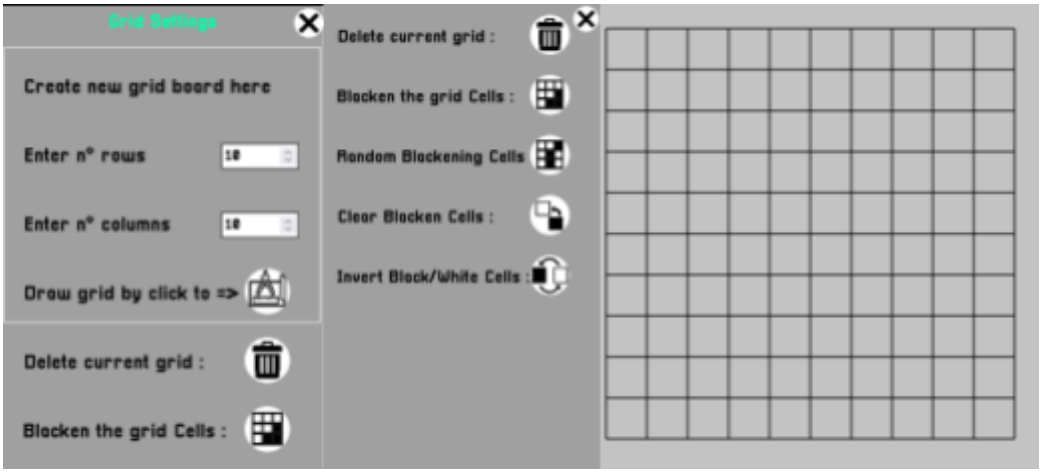


Figure 6.5: Menu réglage de grille avec la grille qui est généré

6. Les algorithmes importants

Pour résoudre notre problématique de pavage le recours à plusieurs algorithmes et méthodes est essentiel pour résoudre les Pavages de Polyominos, notamment :

L'algorithme de recherche exhaustive (Backtracking) : Il explore toutes les possibilités de placement des polyominos et vérifie leur validité, en revenant en arrière en cas d'échec.

La méthode Brute Force: Cette méthode essaie de trouver une solution au pavages en testant toutes les possibilités de placement des polyominoes et leur rotations.

La programmation dynamique : Dans le cas de grilles de taille variables, la programmation dynamique peut être utilisée pour trouver des solutions optimales.

Les méthodes aléatoires: Ces méthodes peuvent être employées pour générer des dispositions aléatoires, elles sont utiles pour fournir un pavage toujours différent et conforme aux exigences .

Les heuristiques et les métaheuristiques : Ces approches permettent de donner des solutions utiles pour montrer la faisabilité d'un problème mais ces solutions ne sont généralement pas optimales .

Ces différents algorithmes sont gérés par le fichier "ai.js", où chaque solveur est défini. Les fonctions exportées sont ensuite appelées dans "main.js," qui transmet les paramètres nécessaires comme la liste des polyominos ou la grille. Le fichier "solve.js" dans le dossier popup permet de créer différents boutons pour le menu et d'appeler les fonctions correspondantes.

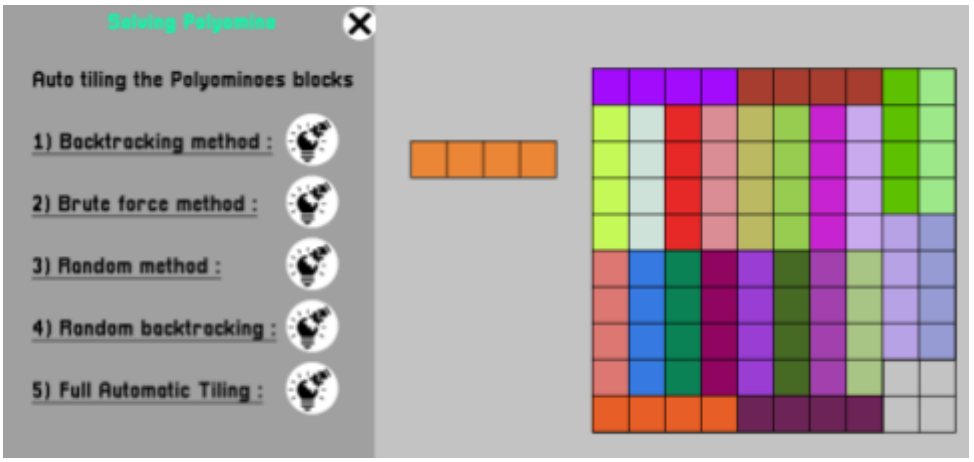


Figure 6.5: Les différents solveur implémenté et une des solutions proposé à partir d'une des pièces disponible

7. Les fonctionnalité supplémentaires

Pour améliorer le confort de l'utilisateur, nous avons créé le menu Settings, utilisant le fichier du même nom. Ce menu permet à l'utilisateur d'interagir avec toutes les pièces en même temps (grâce au fonction de “main.js”) , avec des actions comme remettre les polyominos à leur place d'origine, mélanger leur position, ou les supprimer tous. Cela est utile si l’utilisateur ne veut pas interagir en appuyant sur chacune des pièces via les menu interactif présent autour de chaque pièces.

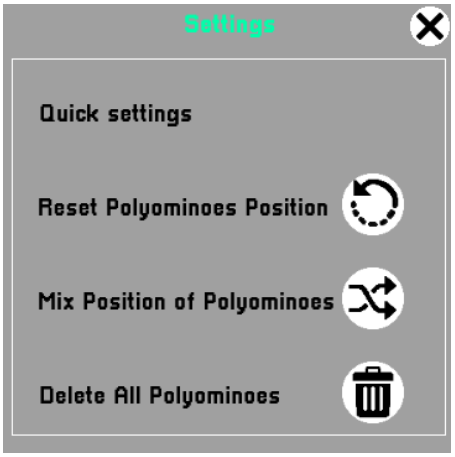


Figure 6.6.1: Menu Settings et ces options

En plus du menu settings, un menu “tutorial” a été créé pour expliquer simplement comment utiliser notre application. Le fichier “tutorial.js”, situé dans le dossier popup, gère le texte explicatif accompagné d'icônes et traite le texte afin d’éviter de déborder et bien dimensionner tous les éléments.:

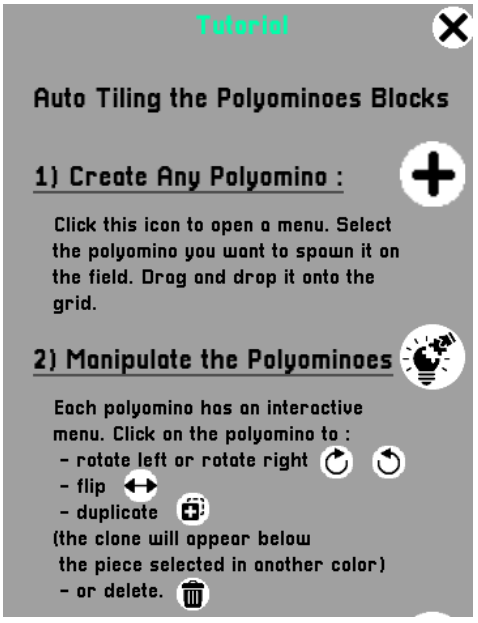


Figure 6.6.2: Menu expliquant le fonctionnement de l’application

8. Le centre du code

Il s'agit du fichier que nous avons mentionné et décrit en partie dans chaque section précédente à savoir le fichier "main.js", ce fichier est véritablement le centre de la partie 2D. Il gère le dessin de la grille, le placement et la rotation des polyominos, la liste des polyominos, le blocage des cellules et bien d'autres fonctionnalités déjà mentionnées. Il importe les fonctions exportées des autres fichiers et permet l'appel des solveurs, il joue un rôle crucial dans le fonctionnement global de l'application car il permet de transmettre les fonctions et contient les informations générales tel que la liste des polyominos et le nombre de colonnes et de lignes.

```
class MainApp {
    constructor() {
        this.canvas = document.getElementById('myCanvas');
        this.gridSize = 30;
        this.rows = this.cols = 10;
        this.polyominoes = [];
        this.selectedPolyomino = null;
        this.icons = {
            flip: new Image(),
            rotateLeft: new Image(),
            rotateRight: new Image(),
            duplicate: new Image(),
            trash: new Image()
        };
        const as = "../assets/";
        this.icons.flip.src = as + 'ic_flip.png';
        this.icons.rotateLeft.src = as + 'ic_rotate_left.png';
        this.icons.rotateRight.src = as + 'ic_rotate_right.png';
        this.icons.duplicate.src = as + 'ic_duplicate.png';
        this.icons.trash.src = as + 'ic_trash.png';
        this.gridBoard = new GridBoard(this.canvas, this.gridSize,
            this.rows, this.cols);
        this.guiController = new GUIController(this);
        this.toolbar = new Toolbar(this);
        this.isBlackening = false;
        this.blackenedCells = new Set();
        this.tooltipPolyominoBlocks();
        this.init();
    };
```

Figure 6.1: Constructeur du fichier main.js

VII. Conception de la Partie 3D

1. Présentation générale

La partie 3D de notre application est conçue de manière similaire à la partie 2D, mais avec un focus spécifique sur la création et la manipulation de polycubes. L'arborescence générale reste en grande partie la même, mais les fichiers sont orientés vers la gestion des objets 3D :

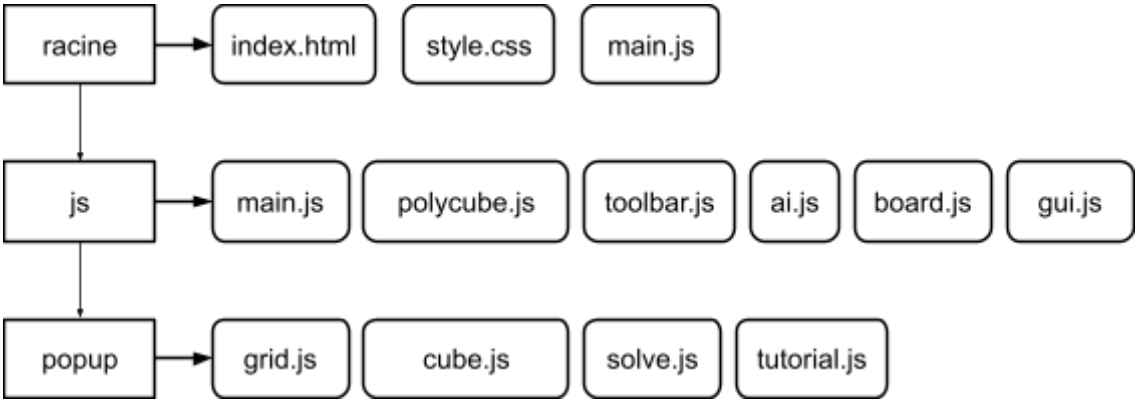


Figure 6.1: Arborescence de la partie 3D

C'est pour cela que nous allons nous concentrer sur les fichiers qui présentent des spécificités notables par rapport à la partie 2D. Par exemple, nous ne détaillerons pas les fichiers liés à la toolbar tels que "toolbar.js" ni ceux du dossier "popups". Nous mettrons plutôt l'accent sur les fichiers qui apportent des fonctionnalités uniques à la partie 3D.

2. Fichier main.js

Ce fichier importe les modules essentiels comme THREE, la bibliothèque principale de Three.js utilisée pour créer et manipuler des scènes 3D, et *OrbitControls*, qui permet d'ajouter des contrôles de navigation pour la caméra, permettant à l'utilisateur de zoomer, tourner et déplacer la vue. La classe MainApp est le cœur de l'application, contenant la logique principale et la gestion des différents éléments de la scène. Le constructeur initialise plusieurs propriétés, notamment *selectedPolycube*, *isDragging*, et *isRightClick*, qui sont utilisées pour gérer l'état des interactions de l'utilisateur avec les polycubes.

La fonction init configure la scène 3D en créant une scène Three.js, une caméra, et un renderer. Elle crée également un plateau de jeu Board, qui, comme en 2D, sera géré dans les fichiers correspondants, ainsi qu'une toolbar. Les événements liés à la souris sont également traités avec *onMouseDown*, *onMouseMove*, et *onMouseUp*. La méthode *onMouseDown* permet de démarrer le déplacement ou la rotation des pièces.

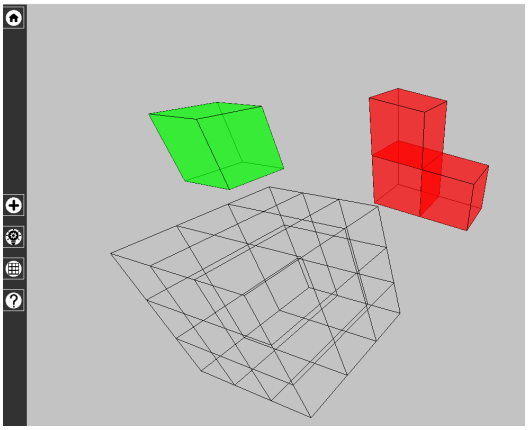


Figure 6.2: fenêtre la partie 3D

2. Création des pièces

Pour créer les polycubes, nous avons modifié le fichier “polyomino.js” en “polycube.js” afin de gérer les cubes. Désormais, il représente un ensemble de cubes connectés dans l'espace 3D. Le constructeur utilise les données de cubeData qui spécifient le nombre de cubes, leurs positions relatives, leur couleur (avec *THREE.MeshBasicMaterial)*, et leur position dans la scène. Grâce à la méthode createPolycube, nous pouvons facilement créer un cube en fonction des informations telles que les coordonnées ou encore la couleur.

Le véritable changement réside dans le fichier “cube.js” du dossier “popup”. Ce fichier est destiné à créer une interface utilisateur interactive pour la manipulation de cubes en 3D.

Le canvas 2D est utilisé pour afficher des informations textuelles à savoir les coordonnées des cubes qui composent les éléments tandis que des boutons permettent à l'utilisateur de spécifier le nombre de cubes et leurs positions. Ceci est fait grâce à des fonctions telles que *createInputField* et *createTextZone* pour les zones descriptives. De même que pour la zone centrale de la fenêtre il est possible de contrôler la caméra avec OrbitControls, permettant de voir les cubes pour une meilleure visualisation.
Ainsi avec le bouton *info* les informations apparaissent tandis que le bouton *create* place le polycube sur la fenêtre.

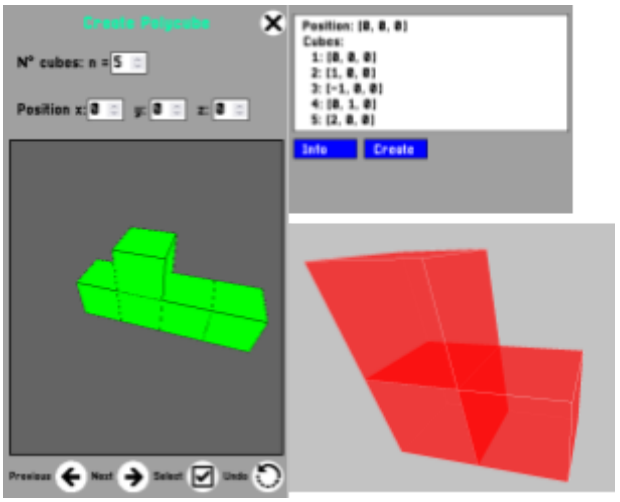


Figure 6.2: Un polycube créer grâce à notre programme et le menu popup

3. La grille en 3D

La classe Board représente le plateau de jeu en 3D. Contrairement à la version 2D qui dessine un ensemble de carrés alignés en colonnes et lignes via une boucle, cette version est plus complexe car elle inclut une dimension supplémentaire, la profondeur.

Comme d'habitude, le constructeur initialise la scène, les dimensions de la grille, et deux groupes Three.js pour les grilles extérieure et intérieure. La fonction *createGridBox* crée les lignes de la grille principale en utilisant des boucles pour couvrir toutes les dimensions spécifiées grâce à *THREE.LineBasicMaterial*. *toggleInnerGrid* permet de montrer ou de cacher la grille intérieure en ajoutant ou en retirant le groupe de la grille intérieure de la scène. *clearGrid* supprime toutes les lignes des groupes de grilles et les retire de la scène, permettant ainsi de réinitialiser la grille.

Ainsi, tout comme la partie 2D, ces fichiers fonctionnent ensemble pour créer une application interactive où l'utilisateur peut manipuler des polycubes dans un environnement 3D, tout en offrant des fonctionnalités de gestion de la scène et des objets 3D.

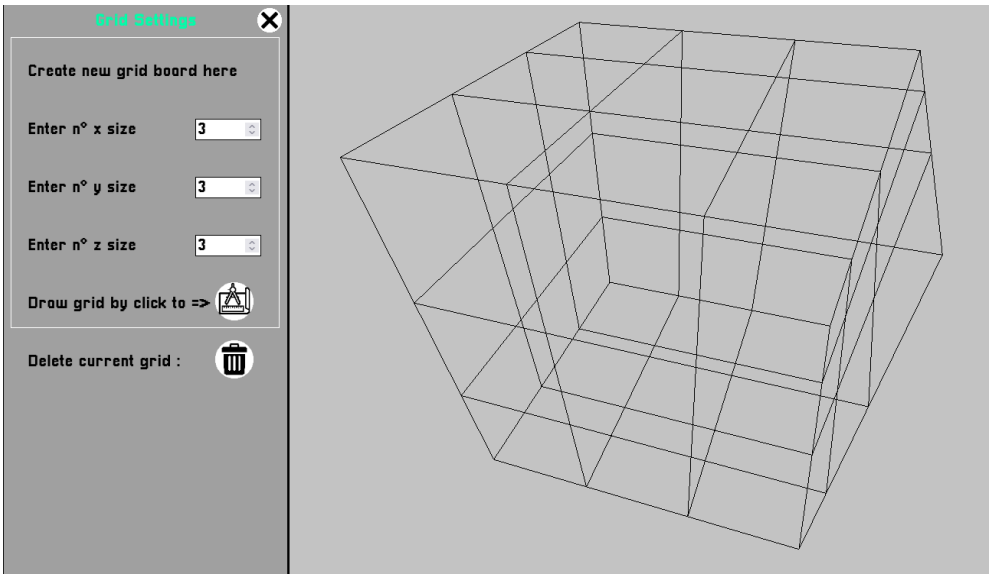


Figure 6.3: la grille et son menu de création

VIII. Contraintes Rencontrées et Pistes d'Amélioration

1. Contraintes

Lors de la mise en place de notre programme, nous avons rencontré plusieurs contraintes. La coordination entre nous deux via des outils comme GitHub et Discord, bien que pratiques, a introduit des défis en termes de gestion de versions et de communication. Nous avons parfois dû réorganiser rapidement des fichiers pour maintenir la cohérence du programme lorsque l'un de nous avait besoin de modifier un fichier utilisé par l'autre.

En termes d'utilisation, la principale contrainte était l'impossibilité, due aux limitations de la page web, de proposer une grande grille ou un nombre très élevé de polyominos. De plus, au niveau des algorithmes de pavage, nous avons constaté que de nombreux algorithmes utilisent des grilles déjà résolues comme une sorte de base de données pour proposer des solutions optimales. Cependant, nous avons décidé de ne pas implémenter cette approche et de proposer des solutions en partant de zéro.

2. Pistes d'Amélioration

Pour améliorer la gestion de la programmation, nous avons pensé à utiliser des fonctionnalités de collaboration en temps réel, ce qui nous permettrait de travailler ensemble via internet. Concernant l'amélioration de la portée du projet, nous avons envisagé d'introduire des modes de jeu tels que des défis chronométrés ou des puzzles avec des contraintes spécifiques, augmentant ainsi l'utilité et la durée de vie de notre programme.

Enfin, pour améliorer l'expérience utilisateur, permettre aux utilisateurs de sauvegarder leurs grilles en cours et de les charger ultérieurement pourrait être très utile pour faciliter la continuité du travail et le partage entre utilisateurs. Ajouter des tutoriels interactifs étape par étape aiderait également les nouveaux utilisateurs à comprendre les fonctionnalités de l'application.

En prenant note de ces contraintes et pistes d'amélioration, nous pourrons enrichir et optimiser nos futurs travaux.

Conclusion

À travers ce projet, nous avons acquis de nombreuses connaissances intéressantes sur les polyominos et les polycubes en explorant leurs propriétés, classifications et implications dans le domaine du pavage. Nous avons constaté comment, malgré leur simplicité, les polyominos en deux dimensions et trois dimensions ont pu engendrer de nombreux défis.

La programmation de notre pavage de polyominos a constitué une expérience très enrichissante. Nous avons dû relever différentes contraintes, ce qui nous a poussé à une réflexion approfondie et nous a permis de mettre en pratique les connaissances acquises pendant nos années de licence en informatique. Cela nous a permis d'obtenir un résultat fonctionnel qui nous satisfait. De plus, nous nous sommes développés personnellement en apprenant l'importance de la persévérance et de l'adaptabilité.

Malgré les difficultés rencontrées, nous avons su les surmonter grâce à une compréhension approfondie de notre code et une préparation minutieuse. Nous sommes donc fiers de voir le projet aboutir et de proposer un programme complet qui répond aux exigences de la consigne. Nous espérons que cela vous plaira.

Bibliographie

Sources utilisé pour les images et les notes

- [Figure 2.1 , 2.2 , 2. 3] Tetromino. (2024, 9 février). Dans *Wikipedia*. https://en.wikipedia.org/wiki/Tetromino
- [Figure 2.4] : Weisstein, Eric W. "Polyomino." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/Polyomino.html
- [Figure 2.5] : Tiling Pentomino Sandwich Sudoku — Rätselportal — Logic Masters Deutschland. (s. d.). https://logic-masters.de/Raetselportal/Raetsel/zeigen.php?chlang=en&id=00041R
- [Figure 3.1] : *The Poly pages. (s. d.). http://recmath.org/PolyPages/PPF/index.htm?Polycubes.html*
- [Figure 4.1] : Andrea Frosini . Article Research Gate : A-polyomino-and-its-representation-as-a-binary-picture-or-matrix_fig1_268167195
- [Figure 4.2] : *three.js docs. (s. d.). https://threejs.org/docs/#api/en/geometries/BoxGeometry*
- [Figure 4.3]: Polyomino blocks. Oliver Merkel. https://omerkel.github.io/PolyominoBlocks/html5/src/index.html

Sources utilisé pour la réalisation de notre document

- [Note 1] *Modèle pour projet tuteuré ou rapport de stage*.
- [Note 2] Solomon W. Golomb (2024, 9 juin). Dans Wikipedia: https://fr.wikipedia.org/wiki/Solomon_W._Golomb
- [Note 3] Page d'explication de règles des echecodame, (Chesskers — the classics with a twist ! (s. d.). https://chesskers.lol/rules)
- [Note 4] Polyomino. (2023, 8 novembre).Dans Wikipedia. https://fr.wikipedia.org/wiki/Polyomino ,
    - Gerard Villemin, "polyominos,introduction aux dominos, pentominos et autres." (s. d.). http://villemin.gerard.free.fr/Puzzle/minoPoly.htm ,
    - Polyform tiling. (s. d.). *https://www.polyomino.org.uk/mathematics/polyform-tiling/*
- [Note 5] Solomon W. Golomb , Polyominoes. (s. d.). Princeton University Press. https://press.princeton.edu/books/paperback/9780691024448/polyominoes
- [Note 6] Polycube. (2024b, juin 12). Wikipedia. https://en.wikipedia.org/wiki/Polycube
- Kevin Gong, ParallelPoly.html. (s. d.). http://kevingong.com/Polyominoes/ParallelPoly.html
- Joseph Myers. *Polyform tiling. (s. d.-c). https://www.polyomino.org.uk/mathematics/polyform-tiling/*
- Alexandre Blondin Massé , Amadou Makhtar Tall, Hugo Tremblay. *On the Arithmetics of Discrete Figures*.
- Olivier Boudini. *Z-tiling of polyominoes and standard basis*.
- *Polyomino. (2024b, mai 12). Wikipedia. https://en.wikipedia.org/wiki/Polyomino*
- *JavaScript. (2024a, mai 23). Wikipedia. https://fr.wikipedia.org/wiki/JavaScript*
- *Three.js – JavaScript 3D library. (s. d.). https://threejs.org/*