Projet : un jeu de placement de briques

UCBL - Département Informatique de Lyon 1 – BDW - 2024

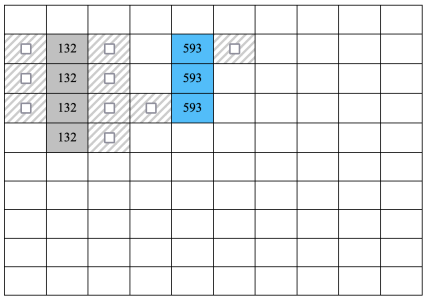
- Ce projet est évalué par plusieurs notes : diagramme E/A, évaluation intermédiaire, soutenance.
- Cette UE est avant tout une introduction aux **bases de données**. La non-utilisation d'une BD relationnelle dans le projet peut se voir attribuer la note 0.
- Le rendu du projet et la soutenance ont lieu lors de la dernière séance de TP. Tout rendu ne respectant pas les consignes peut se voir attribuer la note 0.
- Le projet est réalisé en binôme (pas de trinôme). Un binôme signifie deux étudiant·e·s **du même groupe de TD**.
- Attention à la gestion de votre temps : concentrez-vous en priorité sur l'implémentation des fonctionnalités demandées. Cela ne sert à rien de gagner un point bonus pour quelques améliorations mineures mais de perdre plusieurs points pour des fonctionnalités non développées.

L'objectif du projet est de développer un jeu de placement de briques (type Lego™) afin de compléter une grille.

Voici les règles détaillées : le ou la joueuse possède une grille, sur laquelle on trouve des cases vides et des cases cibles (hachurées sur l'image ci-contre). La pioche contient 4 briques disponibles. Pour gagner, il faut que toutes les cases cibles de la grille soient recouvertes par des briques. À chaque tour, la joueuse sélectionne une brique dans la pioche et peut :

- soit la placer sur sa grille (à un emplacement approprié, i.e., chaque case de la brique recouvre une case cible de la grille encore inoccupée). Par exemple, une brique de 2×3 nécessite 6 cases cibles disponibles pour être placée ;
- soit la défausser (i.e., la retirer de la pioche) afin qu'elle soit remplacée par une nouvelle brique (tirée aléatoirement dans la base de données).

En fin de tour, on vérifie si la joueuse a gagné (i.e., si toutes les cases cibles sont recouvertes). Son score est égal au nombre de tours (l'objectif étant de réaliser le score ou nombre de tours le plus bas possible).



Sur cet exemple de partie, la joueuse a placé une brique grise (#132) de dimension 1×4 (ou 4×1) ainsi qu'une brique bleue (#593) de dimension 1×3. Il lui reste encore 9 cases cibles (hachurées) à recouvrir pour gagner.

1 Des spécifications au script SQL

Les spécifications sont les suivantes. Une partie de jeu se déroule entre une ou plusieurs joueuses. On stocke les dates de début et de fin de la partie, les scores ainsi que la ou les joueuses qui ont gagné (score le plus faible). Concernant les joueuses, on mémorise un prénom, la date d'inscription et un avatar (image de profil). Une partie se divise en plusieurs tours. Les tours sont numérotés à partir de 1 pour chaque nouvelle partie, et sont liés à une joueuse. La brique (ou pièce) est l'élément essentiel de l'application et représente les différents types de pièces disponibles. Chaque brique a un identifiant, un nom, une largeur, une longueur, une hauteur, une forme, une couleur et éventuellement une liste de mots-clés. Un type de brique est fabriqué par une usine (ville, pays) à une date donnée et dans une quantité donnée. Une pièce peut parfois être remplacée par une autre pièce suffisamment similaire : il existe quelques dizaines de critères de substitution (e.g., couleur différente, trou supplémentaire dans la pièce, etc.). Chaque substitution est donc décrite par un critère (identifiant, nom et commentaire). Les constructions sont des assemblages de plusieurs pièces (parfois des milliers), par exemple pour réaliser un repaire de pirates, un château médiéval ou le faucon Millenium. Une construction est caractérisée par son nom, un thème, une description, une année de sortie et des dimensions. La réalisation

d'une construction nécessite de suivre un certain nombre d'étapes, en commençant par la première. En plus de son numéro, chaque étape possède une image et plus rarement des instructions rédigées. Les constructions sont proposées par des fabricants (de briques) ou par des fans. On distingue donc les constructions officielles, qui mentionnent un âge recommandé, et qui sont généralement vendues dans des boîtes (code de référence unique, nom et prix). Pour les constructions amateurs, on ne stocke que le nom de la personne créatrice, et la licence (libre ou non). Pour des raisons pratiques, le système doit connaître le type de construction (officielle ou amateur). Les briques et les constructions s'accompagnent généralement de photos, pour lesquelles on stocke un titre, une description et le chemin vers un fichier. À chaque tour de jeu, la base de données enregistre la brique utilisée ainsi qu'une description de l'action (e.g., placée, défaussée). Enfin, une partie possède une configuration, c'est à dire un ensemble de paramètres représentés par des paires propriété - valeur. Cette configuration est fixée pour la durée de la partie, mais les paramètres peuvent évoluer au fil du temps (e.g., nouvelle propriété).

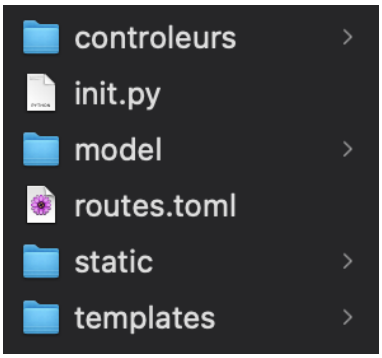
Dans un premier temps, nous allons concevoir la base de données à partir des spécifications. Votre modélisation doit respecter au mieux ces spécifications, mais votre site web n'utilisera pas tous les concepts décrits.

1. Produisez un diagramme entité / association pour ces spécifications (par exemple avec MoCoDo, Looping, AnalyseSI ou JMerise - attention, chaque outil a ses contraintes et limitations). Ce diagramme est à rendre et sera évalué.

Les questions suivantes doivent aussi être réalisées afin de pouvoir programmer votre site web dès la publication de la suite du sujet. Les délivrables produits (schéma relationnel, scripts SQL) ne sont pas à rendre mais peuvent être demandés pendant une soutenance.

2. Produisez le schéma relationnel dérivé de votre diagramme E/A. Si vous générez ce schéma avec un outil de modélisation, il est recommandé de le vérifier et éventuellement de le corriger / compléter.
3. Produisez le script SQL permettant la création de la base de données. Si vous générez ce script avec un outil de modélisation, il sera nécessaire de le vérifier et de le corriger / compléter (ces modifications devraient être stockées dans un autre fichier que celui généré par l'outil de modélisation). Enfin, insérez quelques instances fictives dans les tables les plus importantes.

2 Design du site et pages statiques



Votre site est codé selon l'architecture MVC, et vos fichiers doivent impérativement respecter l'arborescence suivante pour fonctionner avec `bdw-server` : des répertoires `controleurs`, `model`, `static` et `templates` (voir TP4 et TP5).

Vous êtes libres de concevoir vos pages comme bon vous semble. Mais chacune de vos pages doit avoir les zones suivantes :

- Un entête (`<header>`, avec un logo et un nom de site ;
- Un menu (`<nav>`), dont les libellés seront explicites ;
- Le contenu de la page (`<main>`), qui correspond aux fonctionnalités développées dans les sections suivantes.
- Un pied de page (`<footer>`), avec un lien vers la page de BDW et des remerciements (e.g., pour les images).

Il ne faut **pas** implémenter un système d'authentification[1] ! La mise en page et mise en forme se feront évidemment avec des styles CSS. L'esthétique est prise en compte lors de la notation, aussi, soignez votre site. Le projet sera évalué avec le navigateur **Mozilla Firefox**, donc vérifiez le rendu de votre site avec ce navigateur !

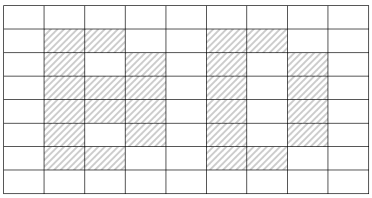
3 Fonctionnalité 1 : accueil et statistiques

Concevez la page d'accueil de votre site. Personnalisez-la comme vous le souhaitez (e.g., objectifs du site, tutoriel).

Cette page d'accueil affiche également des statistiques, notamment :

- Nombre d'instances pour 3 tables de votre choix ;
- Top-5 des couleurs ayant le plus de briques ;
- Pour chaque joueuse, son score minimal et son score maximal ;
- Parties avec le plus petit et plus grand nombre de pièces défaussées, de pièces piochées ;
- Le nombre moyen de tours, pour chaque couple (mois, année) ;
- Top-3 des parties dans lesquelles les plus grandes pièces (longueur × largeur) ont été placées, avec un tri décroissant sur le nombre de pièces utilisées.

4 Fonctionnalité 2 : afficher une grille (motif fixé) et la pioche



Cette seconde fonctionnalité consiste tout d'abord à dessiner une grille respectant le motif ci-contre : dimension 9x8 et les cases hachurées représentent les cases cibles. **Attention :** il est recommandé de lire la suite du sujet afin de réfléchir à une représentation de la grille qui conviendra également pour les prochaines fonctionnalités. Cette grille à motif fixé ne sera pas utilisée pour jouer une partie (sauf pour les binômes qui n'arriveront pas à développer la fonctionnalité 4 de génération automatique d'une grille). Vous êtes libres du style graphique de la grille.

L'autre objectif de cette section est de programmer la pioche. C'est un formulaire qui permet de sélectionner 1 brique parmi 4 proposées. Quand le formulaire est soumis, on affiche les informations sur la brique sélectionnée.

- Les 4 briques sont choisies aléatoirement dans la BD, mais uniquement parmi les briques qui ont une longueur ou une largeur inférieure ou égale à 2 ;
- La hauteur des briques n'est pas utilisée ;
- Quand une brique est sélectionnée par la joueuse, on complète la pioche avec une nouvelle brique choisie aléatoirement dans la BD (aussi avec une largeur ou longueur <= 2).

[1] Un système d'authentification se base sur des protocoles standardisés et sécurisés comme SSL, Kerberos ou CAS.

5 Fonctionnalité 3 : intégrer les briques

Cette fonctionnalité consiste à migrer des données fournies dans votre base de données. Ces données ne sont malheureusement pas correctement modélisées, et il est donc nécessaire de les transformer avant de les stocker dans votre base. C'est un processus fréquent appelé *intégration de données*. L'utilisation de ces données est **obligatoire**, et votre script d'intégration doit être **exécutable sur de nouveaux jeux de données** lors de la soutenance.

Vous allez donc écrire un script SQL (ou éventuellement Python) qui interroge le jeu de données fourni pour extraire les informations sur des briques, les transformer si nécessaire, puis les insérer dans votre base. En pratique, vous allez tester votre script à plusieurs reprises, donc prévoyez de vérifier si les données n'existent pas déjà avant d'insérer !

On peut décomposer le processus d'intégration en plusieurs étapes :

- Réfléchir aux correspondances entre les données existantes et votre schéma (e.g., à quel attribut de votre schéma correspond tel attribut fourni). Vérifier si les types de données sont cohérents entre attributs correspondants, et quelles sont les transformations nécessaires au niveau des valeurs. N'oubliez pas que le jeu de données fourni n'est pas bien modélisé : si vous modifiez votre schéma de BD pour intégrer plus facilement les données fournies, votre schéma ne respectera plus les spécifications demandées et vous serez pénalisé·e ;
- Créer le jeu de données (schéma `legos`, table `piece`) ;
- Pour chaque tuple résultat d'un `SELECT`, appliquer éventuellement des transformations sur les valeurs (soit directement en SQL, soit en Python), puis écrivez une requête `INSERT` pour peupler vos tables. Si certaines données sont manquantes et ne permettent pas de respecter les contraintes de votre schéma, utilisez un `NULL` ou choisissez une valeur par défaut. Enfin, n'oubliez pas de peupler les tables liées !

Au final, vous devriez avoir 1280 briques dans votre BD.

6 Fonctionnalité 4 : générer une grille aléatoire

L'objectif est de créer une nouvelle page avec une grille générée aléatoirement (cases vides et cases cibles) selon les paramètres suivants :

- La taille de la grille (longueur et hauteur) est configurable par le ou la joueuse ;
- Le nombre de cases cibles est un nombre aléatoire compris entre 10% et 20% du nombre total de cases ;
- Une case cible doit toujours être adjacente (un côté commun) avec au minimum une autre case cible.

Suggestion d'algorithme (ou s'inspirer de maze generation algorithms) :

- Initialiser une grille avec uniquement des cases vides ;
- Sélectionner aléatoirement une première case cible ;
- Pour chaque case cible à ajouter, choisir une direction aléatoire (parmi haut, bas, gauche, droite) et vérifier si la case correspondante (première case cible + direction choisie) est valide (i.e., dans la grille et case vide) : si oui, la transformer en case cible et répéter, sinon choisir une autre direction ;
- Gérer les situations exceptionnelles (e.g., pas suffisamment de cases cibles car le motif est en forme de ”spirale”).

Lors de la soutenance, il est fort probable que les enseignant·e·s testent votre application en saisissant des valeurs absurdes susceptibles de déclencher des erreurs dans votre application. Alors essayez de penser au pire !

7 Fonctionnalité 5 : programmer une partie

Cette dernière fonctionnalité consiste à programmer une partie (1 joueur humain, qui essaie de réaliser le score le plus faible). Vous avez normalement déjà programmé la génération de la grille (aléatoire) et la pioche. La difficulté principale concerne la vérification de la conformité d'un emplacement pour une brique sélectionnée.

Paramètres de la partie :

- Dimensions de la grille ;
- Optionnel : un nombre de tours maximum (au bout de laquelle la partie s'arrête) ;
- Mode : facile (les briques sélectionnées ont obligatoirement une longueur ou une largeur inférieure ou égale à 2) ou difficile (n'importe quelle brique peut être sélectionnée).

Précisions sur la gestion de la partie :

- Les briques placées sur la grille doivent être identifiées et colorées (voir la figure illustrant un exemple de partie, mais vous êtes libres d'améliorer le style graphique) ;
- En cas d'erreur (e.g., mauvais placement sur la grille, oubli de sélectionner une case cible), le nombre de tour s'incrémente quand même (pénalité) ;
- Si une partie n'est pas terminée (abandon ou nombre de tours maximum atteint), le score vaut 999.

Les informations sur la partie seront progressivement stockées en base de données. Pensez à gérer les erreurs (messages pertinents). Libres à vous d'ajouter de nouvelles fonctionnalités (e.g., page des meilleurs scores, système de niveau pour une difficulté croissante, des briques bonus).

8 Préparation des livrables

Deux livrables sont à rendre **le jour de soutenance de votre projet, avant 23h59, sur Tomuss** :

- Une archive de votre site web, en **zip** ou **rar** (colonne *archive_projet*), qui contient a-minima :
  - le répertoire de votre site (code complet, commenté et indenté, respectant l'arborescence de la section 2 et incluant votre fichier de configuration `config.toml`) ;
  - un fichier `.txt` ou `.sql` avec le script SQL **"exécutable"** de création de votre base de données. Votre BD ainsi que ce script SQL doivent posséder des instances en nombre suffisant pour simplifier les tests.
- Une affiche en **pdf**, de 1 page maximum (colonne *affiche_projet*). Ce document n'est pas évalué directement, mais offre à vos enseignant·e·s un aperçu de votre site (sans l'installer), notamment pour harmoniser les notes après la soutenance. Sur cette affiche (au style graphique libre), vous mettrez :
  - Les noms et prénoms du binôme ;
  - Un résumé des fonctionnalités implémentées (e.g., sous forme de liste ou tableau) ;
  - Des captures d'écran annotées de votre site.

9 Soutenance

La soutenance dure 20 minutes par binôme et se décompose en deux parties : une démonstration du site (5 minutes) et une séance de questions (15 minutes). Les conditions suivantes s'appliquent :

- Une bonne démonstration doit être préparée : réfléchissez à un scénario (e.g., ce que vous allez montrer, dans quel ordre, quelles valeurs seront saisies dans les formulaires) et à la manière de le présenter (e.g., transitions, répartition du temps de parole, interaction ou pas avec l'enseignant·e). Concentrez-vous sur les points importants (e.g., si vous avez implémenté les dernières fonctionnalités, inutile de montrer les liens dans le footer...). Enfin, respectez le temps (vous serez interrompus au bout des 5 minutes, et pénalisés) ;
- Les deux membres du binôme doivent parler lors de la démonstration. Entrainez-vous ;
- Vous présentez soit sur votre machine, soit sur une machine de la fac. Prévoyez d'arriver quelques minutes en avance devant la salle, et quand on vous fait entrer, préparez-vous pour la présentation et les questions (i.e., lancement du site dans le navigateur, ouverture des fichiers source) ;
- Vous devez répondre à différentes questions pendant les 15 minutes. Donc donnez des **réponses claires et concises**. Si vous ne savez pas répondre à une question, dites-le pour ne pas perdre de temps ;
- Si vous arrivez en retard à votre soutenance, vous passez en fin de séance... ou pas du tout selon les disponibilités de votre jury.

Concours (optionnel) : Pour vous détendre (en cas de blocage sur un bug, d'engueulade avec votre binôme, etc.), deux concours sont organisés, celui du **meilleur nom de site** et celui du **plus beau logo**. Les deux binômes gagnants remporteront un paquet de chamallows. Indiquez si vous participez au concours lors de la soutenance. Soyez créatifs/ves !

Calvin et Hobbes (Bill Watterson)