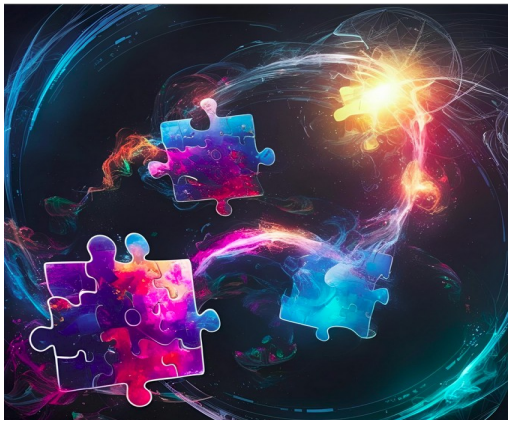


v1.1 Projet CY-Puzzle

FILIERE ING1-GI • 2024-2025
AUTEURS E. ANSERMIN – R. GRIGNON
E-MAILS eva.ansermin@cyu.fr – romuald.grignon@cyu.fr

DESCRIPTION DU PROJET

- Ce projet vous propose de réaliser une application permettant de **résoudre un puzzle**.
  Les éléments à assembler seront des images fournies par l'équipe pédagogique, qui sont obtenus à partir d'une image complète.
- Votre application devra pouvoir choisir le dossier contenant les morceaux du puzzle, lancer l'algorithme de résolution, puis afficher les pièces à l'écran le puzzle terminé.
- Les images fournies sont découpées et s'assemblent au pixel près pour vous simplifier le travail.

INFORMATIONS GENERALES

- **Taille de l'équipe**
  Ce projet est un travail d'équipe. Il est préférable de se réunir en groupe de 5 personnes. Si le nombre d'étudiants n'est pas un multiple de 5 et/ou si des étudiants n'arrivent pas à constituer des groupes, c'est aux chargés de projet de statuer sur la formation des groupes. Pensez donc à anticiper la constitution de vos groupes pour éviter des décisions malheureuses.
- **Démarrage du projet et jalons**
  Vous obtiendrez de plus amples informations quant aux dates précises de rendu, de soutenance, les critères d'évaluation, le contenu du livrable, …, quand le projet démarrera officiellement.
  Quel que soit le planning initial du projet, vous veillerez à planifier plusieurs rendez-vous d'avancement avec votre chargé(e) de projet. C'est à votre groupe de prendre cette initiative. L'idéal est de faire un point 1 ou 2 fois par semaine mais cette fréquence est laissée libre en fonction des besoins identifiés avec le tuteur de projet.
- **Versions de l'application**
  Pour éviter d'avoir un projet non fonctionnel à la fin, il vous est demandé d'avoir une version en ligne de commande fonctionnelle. Ceci vous permettra de tester votre modèle de données indépendamment de l'interface graphique. C'est la version avec l'interface graphique JavaFX qui sera bien entendu évaluée mais dans

le cas où certaines fonctionnalités ne seraient pas visibles avec cette interface, vous devez pouvoir présenter toutes les fonctionnalités de votre code Java en ligne de commande.

- **Dépôt de code**
  Vous devrez déposer la totalité des fichiers de votre projet sur un dépôt central Git. Il en existe plusieurs disponibles gratuitement sur des sites web comme ***github.com*** ou ***gitlab.com***. La fréquence des commits sur ce dépôt doit être au minimum de 1 commit / 2 jours.
- **Rapport du projet**
  Un rapport écrit est requis, contenant une brève description de l'équipe et du sujet. Il décrira les différents problèmes rencontrés, les solutions apportées et les résultats. L'idée principale est de montrer comment l'équipe s'est organisée, et quel était le flux de travail appliqué pour atteindre les objectifs du cahier des charges. Le rapport du projet peut être rédigé en français ou en anglais.
  Ce rapport contiendra en plus les éléments techniques suivants : un document de conception UML (diagramme de classe) de votre application, et un document montrant les cas d'utilisations.
- **Démonstration**
  Le jour de la présentation de votre projet, votre code sera exécuté sur la machine de votre chargé(e) de TD. La version utilisée sera la **dernière** fournie sur le dépôt Git **avant** la date de rendu. Même si vous avez une nouvelle version qui corrige des erreurs ou implémente de nouvelles fonctionnalités le jour de la démonstration, c'est bien la version du rendu qui sera utilisée.
  En parallèle, il vous faudra obligatoirement une deuxième machine avec votre application fonctionnelle car une partie de votre groupe aura des modifications de code à faire pendant que l'autre partie fera la présentation/démonstration de votre projet.
- **Organisation de l'équipe**
  Votre projet sera stocké sur un dépôt git (ou un outil similaire) tout au long du projet pour au moins trois raisons :
  - éviter de perdre du travail tout au long du développement de votre application
  - être capable de travailler en équipe efficacement
  - partager vos progrès de développement facilement avec votre chargé(e) de projet.

  De plus il est recommandé de mettre en place un environnement de travail en équipe en utilisant divers outils pour cela (Slack, Trello, Discord, ...).

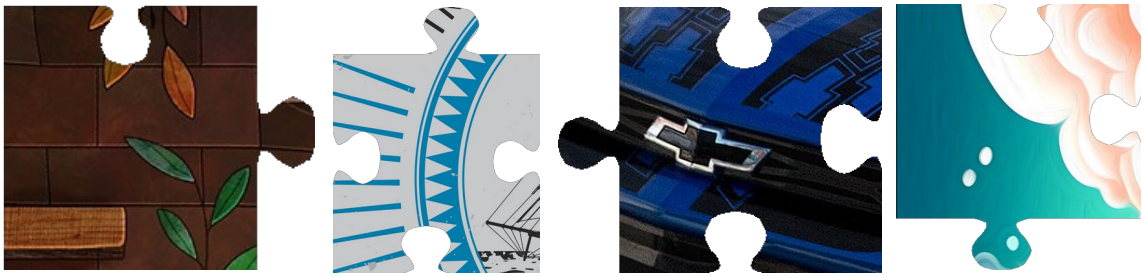
CRITERES GENERAUX

- Le **but principal** du projet est de fournir une **application fonctionnelle** pour l'utilisateur. Le programme doit correspondre à la description en début de document et implémenter toutes les fonctionnalités listées.
- Votre code sera **commenté** de manière adéquate.
- Tous les éléments de **votre code** (variables, fonctions, commentaires) seront écrits **en anglais** obligatoirement.

- Votre code devra être commenté de telle manière que l'on puisse utiliser un outil de génération de documentation automatique de type **JavaDoc**. Un dossier contenant la documentation générée par vos soins sera présent dans votre dépôt de code lors de la livraison.
- Votre projet doit être utilisable au clavier ou à la souris en fonction des fonctionnalités nécessaires.
- Votre application ne doit jamais s'interrompre de manière **intempestive** (crash), ou tourner en boucle indéfiniment, quelle que soit la raison.
  Toutes les erreurs doivent être gérées correctement. Il est préférable d'avoir une application stable avec moins de fonctionnalités plutôt qu'une application contenant toutes les exigences du cahier des charges mais qui plante trop souvent.
  Une application qui s'arrête de manière imprévue à cause d'une exception, par exemple, sera un événement très pénalisant.
- Votre application devra être **organisée en modules** afin de ne pas avoir l'ensemble du code dans un seul et même fichier par exemple. Apportez du soin à la conception de votre projet avant de vous lancer dans le code.
- Le livrable fourni à votre chargé(e) de TD sera simplement l'**URL** de votre **dépôt Git** accessible **publiquement**.

FONCTIONNALITES DU PROJET

- Les images des pièces en entrée de votre programme sont au format PNG. Chaque pièce provient d'une image complète et s'assemble parfaitement avec ses voisines. Les pièces du contour posèdent au minimum un bord rectiligne.
  La taille des pièces est variable. Voici des exemples de pièces :



- Il existe 2 types de pièces fournies :
  - celles qui sont orientées dans le sens de l'image
  - celles qui sont orientées de manière aléatoire par pas de 90°

  Votre programme doit déjà pouvoir effectuer une résolution avec des pièces orientées correctement, puis il faudra l'améliorer pour que votre algorithme prenne en compte des pièces orientées aléatoirement. Cette amélioration doit être transparente du point de vue de l'utilisateur.

- La résolution d'un puzzle se fait avec un algorithme qui va maximiser l'association des paires de pièces en fonction de la forme de la jointure. C'est donc un graphe qui représente l'ensemble des liaisons des pièces entre elles jusqu'à ce qu'il n'y ait plus aucune pièce seule.

Si l'algorithme arrive à un point où il n'a pas réussi à connecter toutes les pièces, il doit rebrousser chemin sur la structure créée en mémoire afin de tenter une connexion différente, et ceci jusqu'à réussite du puzzle.

Dans l'hypothèse où votre algorithme n'aboutirait pas, le programme doit tout de même afficher l'état final du puzzle même s'il est incorrect/incomplet. Le but est d'avoir un retour visuel du résultat.

- La taille des pièces est variable, ainsi que leur nombre. Votre programme peut compter le nombres de pièces totales mais il faudra que votre algorithme construise à minima la bordure du puzzle pour savoir combien de pièces il y a en hauteur et en largeur : ces 2 valeurs doivent être affichées à la fin de la résolution.
- Les pièces d'un puzzle sont supposées être des images PNG situées dans un dossier à part. On part du principe que toutes les pièces du puzzle sont présentes dans le dossier (1 pièce = 1 fichier image) et qu'il n'y a aucun autre fichier dans le dossier. C'est donc un dossier que votre programme devra prendre en paramètre pour effectuer la résolution. Si votre programme parvient à créer le contour, vous obtiendrez par calcul le nombre total de pièces, de fait votre programme pourra à ce stade vérifier que le nombre de fichiers image dans le dossier est égal à ce résultat. Si ce n'est pas le cas, votre programme devra afficher un avertissement à l'écran, sans pour autant stopper la résolution du puzzle.
- L'image finale devra être recrée et sauvegardée par votre programme avec la taille réelle sur le disque dur. A vous de voir le nommage précis des différentes tentatives.

  Comme la taille de l'image totale ne sera pas forcément égale (ni en taille, ni en ratio hauteur/largeur) à la fenêtre d'affichage de votre programme, vous devrez pouvoir redimensionner, sans déformer, l'image totale pour qu'elle tienne complètement à l'écran.

  Les images peuvent etre de tailles très petites comme très grandes, avec peu de pièces ou très nombreuses. Un ordre de grandeur de dimension à pouvoir traiter est de l'odre de 15-20 Mpixels, et le nb de pièces peut monter jusqu'à 10000 pièces (ex : 200x50).
- Si vous voulez tester votre programme avec d'autres images que celles fournies, il faudra en faire la demande au plus tôt pour que l'équipe pédagogique les génère et les ajoute sur la page de cours et qu'ainsi tous les groupes puissent en profiter. Chaque image fournie sera générée avec plusieurs découpages différents, et avec les deux modes d'orientation (orientation dans le sens de l'image, orientation aléatoire).
- Pour chaque résolution de puzzle, le temps complet de traitement de votre algorithme devra être affiché. Il faudra distinguer le temps de chargement des données en mémoire, du temps de résolution, et du temps d'affichage du résultat. Ces 3 durées seront affichées à la fin de la génération.

  Au cours du traitement, des informations sur l'état du programme doivent être affichées (chargement en cours, résolution en cours, sauvegarde, affichage, nb de pièces déjà placées, ...).

RESSOURCES UTILES

- **Github**
  https://www.github.com
  https://docs.github.com/en/get-started/quickstart/hello-world
- **Patrons de conception**
  https://fr.wikipedia.org/wiki/Patron_de_conception
- **Sérialisation**
  https://fr.wikipedia.org/wiki/Sérialisation
- **Connexion à une base de données**
  https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html
- **Gestion d'image au niveau pixel en Java**
  https://docs.oracle.com/javase/8/docs/api///?java/awt/image/BufferedImage.html
- **Robot solving puzzles faster than humans ^_^**
  https://www.youtube.com/watch?v=Sqr-PdVYhY4&