V1.0 Projet CY-BER Path

CLASSE préING1 • 2023-2024
AUTEURS Eva ANSERMIN - Romuald GRIGNON
E-MAILS eva.ansermin@cyu.fr - romuald.grignon@cyu.fr

DESCRIPTION DU PROJET

Le but de ce projet est de réaliser un petit jeu sur le terminal où des robots glisseurs doivent atteindre des cibles sur une grille en 2D.
Ce jeu peut se jouer à 2 joueurs ou plus : la limite n'est pas définie.
Le jeu se déroule sur une grille rectangulaire ou carrée, sur laquelle se trouvent des murs par endroits qui empêchent les robots d'avancer librement.
Les robots doivent rejoindre une cible définie aléatoirement en un minimum de mouvements. Un mouvement est un déplacement en ligne droite jusqu'à ce que le robot rencontre un obstacle (un autre robot ou un mur). Les robots « glissent » donc dans une direction donnée à chaque mouvement.
A chaque tour, un robot est choisi aléatoirement, et une cible aléatoire également. Chaque joueur doit réfléchir au nombre de mouvements nécessaires pour que le robot atteigne sa cible, uniquement en regardant la grille de jeu.
Une fois que tous les joueurs ont déterminé dans leur tête le nombre de mouvements, ils doivent saisir ces nombres, et c'est le joueur avec le nombre de mouvements le plus faible qui doit jouer : il doit déplacer son robot en indiquant à chaque fois la direction dans la grille orthogonale (4 directions).
Le programme compte le nombre de mouvements au fur et à mesure, et quand le robot atteint sa cible, si le nombre de mouvements est correct, le joueur marque 1 point, sinon tous les autres joueurs marquent 1 point.
Le gagnant de la partie est celui qui obtient le plus grand score à la fin d'un nombre de manches fixé à l'avance.
Ce projet est donc un projet de type jeu qui va notamment nécessiter une bonne maîtrise :

- de la gestion d'une grille 2D
- de la gestion de l'affichage

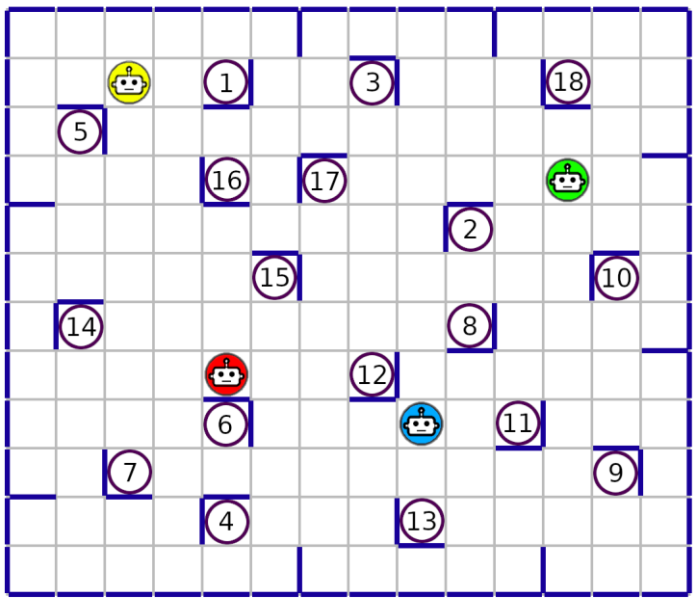
INFORMATIONS GENERALES

Taille de l'équipe
Ce projet est un travail d'équipe. Il est autorisé de se réunir en groupe de 3 au maximum. Si le nombre total d'étudiants n'est pas un multiple de 3 et/ou si des étudiants n'arrivent pas à constituer des groupes, c'est au chargé de TD de statuer sur la formation des groupes. Pensez-donc à anticiper la constitution de vos groupes pour éviter des décisions malheureuses.

	Démarrage du projet 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. **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. **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. **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 limite 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. **Organisation** Votre projet complet **devrait (dans l'idéal)** être 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, et partager vos progrès de développement facilement avec votre chargé 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 généreusement **commenté**. • Tous les éléments de **votre code** (variables, fonctions, commentaires) seront écrits dans la **même langue**. Langue anglaise conseillée mais pas obligatoire. • 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 de 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 se stoppe de manière imprévue à cause d'une erreur de segmentation ou d'une exception, par exemple, sera un événement très pénalisant. • Votre application devra être **modulée** 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**. Même si vous n'avez pas utilisé ce dépôt régulièrement au cours du projet, le code final sera livré dessus.

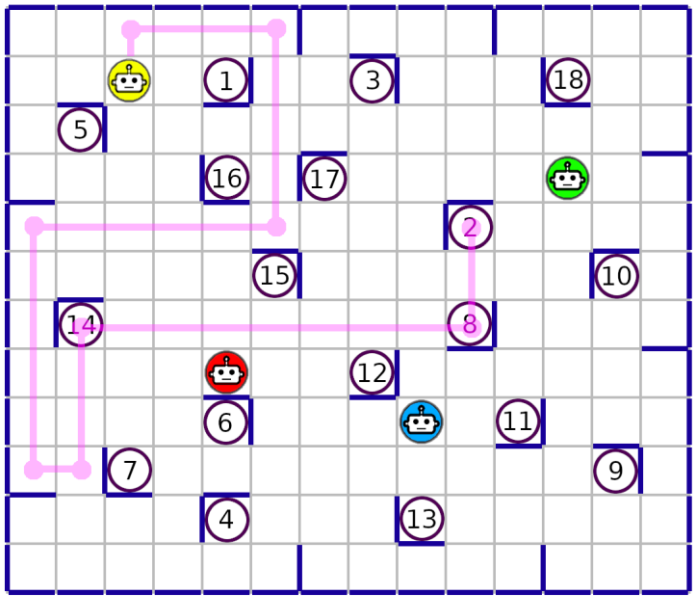
FONCTIONNALITES DU PROJET

- Au lancement, le programme demande le nombre de joueurs.
- Le programme doit créer la grille de jeu, dont les dimensions des côtés doivent être aléatoires dans une plage de valeurs entre 15 et 20 à chaque partie.
- Sur la grille doivent être placées aléatoirement 18 cibles. Ces cibles sont toutes identifiables par des icones/caractères/nombres/couleurs différents. Il ne peut pas y avoir 2 cibles situées côte à côte, même en diagonale. Il doit toujours y avoir au moins une case disponible tout autour de chaque cible.
- Les cibles ne peuvent pas être placées sur les cases de la bordure extérieure de la grille.
- Chaque cible est entourée de 2 murs qui forment un angle droit.
- Sur la grille se trouvent également des murs perpendiculaires aux bords extérieurs de la grille : il y en a 2 sur chaque bord, placés aléatoirement.
- Ensuite la grille doit disposer 4 robots placés aléatoirement sur la grille. Les robots ne doivent pas être placés sur les cibles au départ.
- Ci-dessous un exemple de grille au départ d'une partie (attention ici les dimensions de la grille sont réduites par rapport au cahier des charges pour des raisons de mise en page)

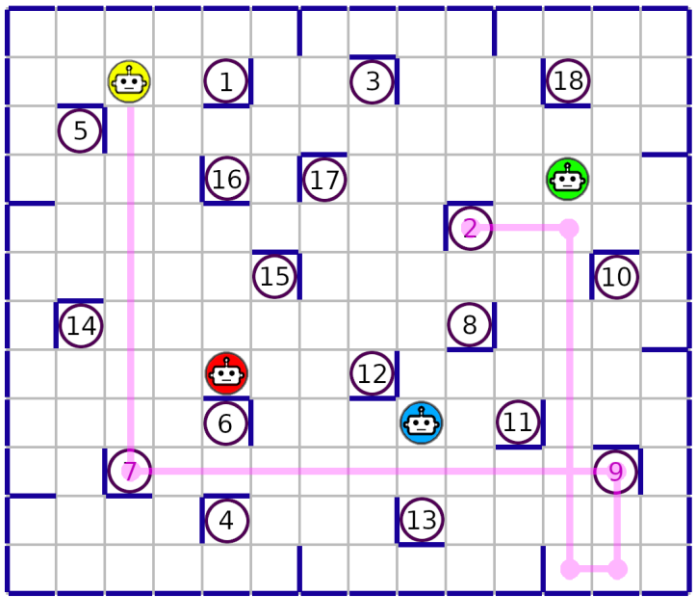


- Ici les cibles sont représentées par les nombres entourés d'un cercle, les 4 robots sont visibles avec leurs couleurs, et les murs sont représentés par des traits épais colorés. On visualise les 2 murs qui forment un angle droit autour de chaque cible (orientation aléatoire), les 2 murs perpendiculaires à chaque bord extérieur de la grille (positions aléatoires), ainsi que les murs tout autour de la zone. Notez les cases vides autour de chaque cible : il ne peut pas y avoir 2 cibles côte à côte, ou des cibles sur les bords extérieurs de la grille.
- Les murs empêchent les robots d'avancer.
- Maintenant que la grille est affichée, le programme va choisir un robot et une cible aléatoires, les indiquer à l'écran, et va chronométrer une durée de plusieurs secondes (à fixer en fonction du niveau de difficulté voulu) pendant laquelle les joueurs vont réfléchir aux déplacements nécessaires pour que le robot atteigne la cible.
- Au bout de la durée fixée, le programme va cacher la grille, et va demander à chaque joueur d'entrer le nombre de mouvements nécessaires.

- Ci-dessous se trouve un schéma de déplacement possible pour que le robot jaune rejoigne la cible #2. On voit que le joueur associé a imaginé un parcours de 9 mouvements. Le robot peut passer par d'autres cibles, cela n'a aucun impact ici.



- Ci-dessous se trouve un autre schéma de parcours imaginé par un autre joueur, et celui-ci ne prend que 6 mouvements. Notez le fait que le robot vert sert d'obstacle : un robot ne peut pas traverser une case occupée par un autre robot.



- Une fois que chaque joueur a indiqué son nombre de mouvements, le programme va demander au joueur qui a indiqué le nombre le plus petit de déplacer le robot : cela signifie de réafficher la grille, demander la direction (parmi 4) au joueur, faire glisser le robot dans la direction demandée (jusqu'à ce qu'il rencontre un obstacle) et incrémenter le nombre de déplacements. Ces étapes vont être réitérées en boucle.
- Si le nombre de déplacements courant dépasse celui indiqué initialement sans que le robot n'atteigne la cible, le programme va ajouter 1 point à tous les autres joueurs. Le joueur courant ne marque donc aucun point.
- Si le robot atteint la cible exactement au bout du nombre de mouvements prévu, le programme attribue 2 points au joueur courant.

	• Si le robot atteint la cible en un nombre de déplacements inférieur à la valeur indiquée initialement, le joueur courant perd 1 point. Certes il a atteint la cible mais il a mal prévu le nombre de mouvements. • Dans tous les cas, le programme se stoppe en l'état et une nouvelle manche démarre avec affichage de la grill, choix d'un robot et d'une cible et lancement du chronomètre. • Le programme ne peut pas choisir une cible si un robot est déjà placé dessus. • Le programme se stoppe au bout d'un nombre de manches fixées. • Le numéro de la manche i/N est toujours affiché à l'écran. • Le nom du gagnant et les scores sont affichés à la fin de la partie.
RESSOURCES UTILES	**Github** • www.github.com • https://docs.github.com/en/get-started/quickstart/hello-world **Modification des couleurs (police et fond) dans le terminal** • http://sdz.tdct.org/sdz/des-couleurs-dans-la-console-linux.html **Affichage d'emoji dans le terminal** • Liste des emojis : https://unicode.org/emoji/charts/full-emoji-list.html