Thibaut Lesage
Simon Elice
Srinjoy Sanyal
Esteban Bernagou
Corentin Bouffioux

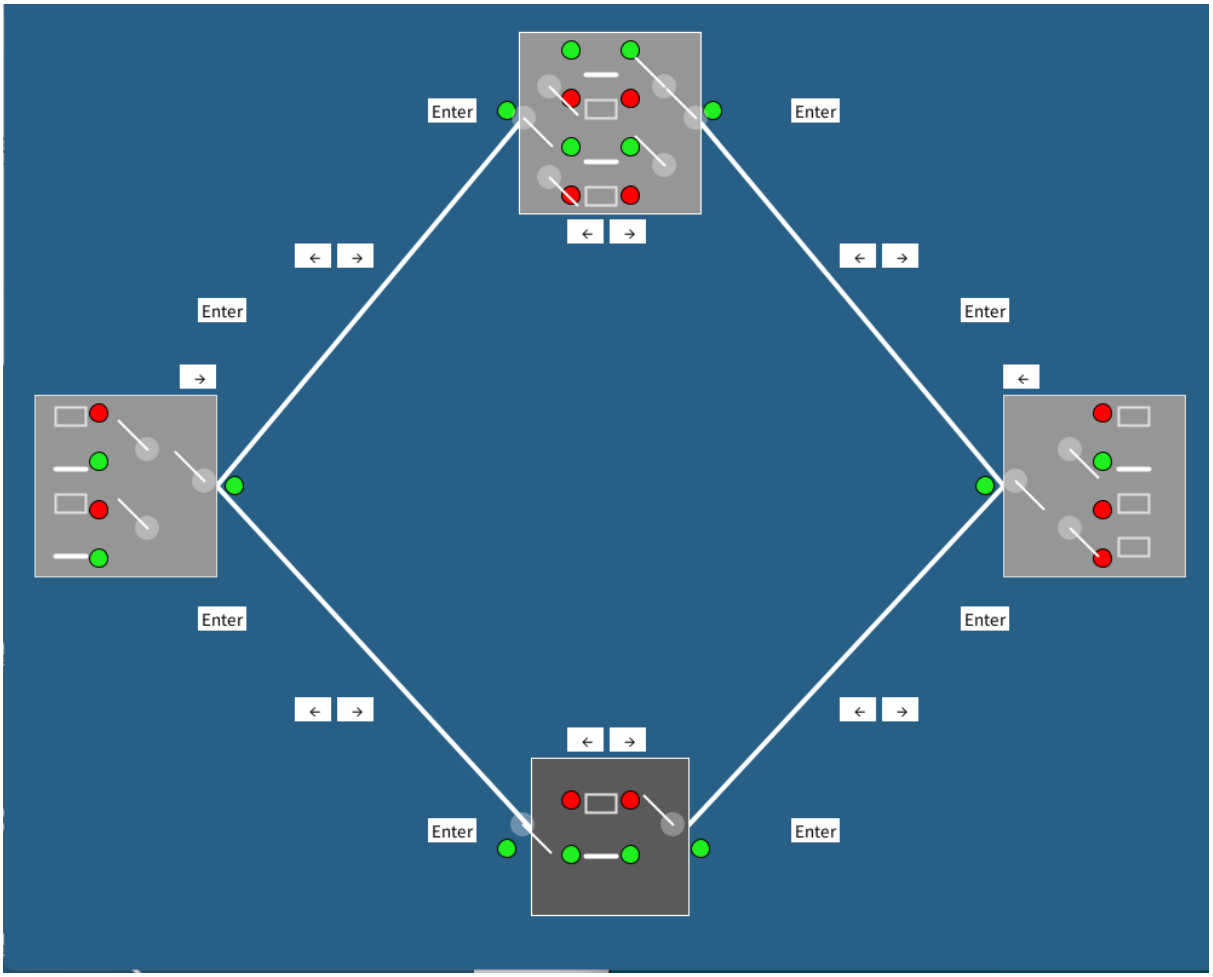
[INFOB236] - Projet de programmation
Animation processing - Projet de groupe

1. Introduction

Dans le cadre du cours “Projet de Programmation”, nous vous présentons le rapport de notre projet de groupe sur notre animation processing. Pour ce projet, malgré un problème avec l’utilisation du plugin eventb2java, nous avons donc dû recommencer le code à zéro pour que l’animation soit plus propre et réalisable. Nous vous expliquerons dans les différents pages ci-dessous : le schéma, les différentes classes et le système de mise automatique des feux implantés dans l’animation. Nous espérons, à travers ce rapport, vous montrer le fruit de notre travail et de nos efforts durant ce Q2. Bonne lecture à vous.

2. Sommaire

3. Explication du schéma et des interactions possibles



- Les carrés gris clair sont les gares principales et le carré gris foncé est une gare secondaire.
- Les flèches au-dessus des gares servent à faire sortir les trains.
- Les boutons enter servent à faire rentrer les trains dans les gares.
- Les points verts ou rouges correspondent aux feux d'entrée et de sortie des gares, ils sont gérés automatiquement par le programme.
- il faut cliquer sur le rond au bout des aiguillages pour les faire changer de position.
- Dans les gares lorsqu'il n'y a pas de train la voie est représentée par une ligne et sinon elle est représentée par un rectangle.
- Les feux sont tout de même interactifs même si ils sont gérés par le programme
- Nos 3 gares principales nommées P1, P2 et P3 dans notre programme sont respectivement les gares à gauche, en haut et à droite. Notre gare secondaire nommée S1 dans notre code est la gare en bas .

4. Explication de chaque classe

- **Gare**: cette classe contient des fonctions liées à la gare. La fonction draw dessine les gares primaires, la fonction “Sdraw“ les gares secondaires, et “plateformeDraw” les plateformes dans la gare qui lui est fournie en paramètre. Toutes ces fonctions reçoivent aussi comme paramètre les coordonnées x et y où ils doivent être placés. La fonction “plateformeDraw” dessine des lignes pour les plateformes sans train, et un rectangle sinon.
- **main**: c’est la classe principale où toutes les fonctions des autres classes sont appelées. Cette classe contient aussi plusieurs variables. Parmi ceux-ci se trouvent des listes qui contiennent un entier 0, 1 ou 2 qui représentent respectivement un absence de train, un train allant de gauche à droite et un train allant de droite à gauche. Ces types d’array sont utilisés pour représenter les gares et les rails, et sont présents sous la forme de:
    - “pgare1” pour la gare principale 1
    - “pgare2” pour la gare principale 2
    - “pgare3” pour la gare principale 3
    - sgare pour la gare principale secondaire
    - P1ToP2 pour la voie de la gare principale 1 à la gare principale 2
    - P2ToP3 pour la voie de la gare principale 2 à la gare principale 3
    - P1ToS1 pour la voie de la gare principale 1 à la gare secondaire
    - S1ToP3 pour la voie de la gare secondaire à la gare principale 3

  Un autre array dans cette classe est feux, qui contient tous les feux, représentés par l’objet de classe Feux, qui sera expliqué plus tard. Une autre variable est aiguillage, qui contient des objets de type Aiguillage, expliqués plus tard.
- **Aiguillage**: cette classe prend comme paramètres les coordonnées x et y où il faut le placer, ainsi qu’un booléen qui indique s’il est orienté vers la gauche ou la droite. Elle contient aussi une variable locale booléenne “versDroite” qui indique la direction vers laquelle pointe l’aiguillage. La fonction draw dessine l’aiguillage et si “versDroite” est vrai, alors il va pointer vers la droite avec un angle de 45°. Sinon, il pointera vers la gauche avec un angle de 45°.
- **Button :** La méthode “updateRect” reçoit en argument “getX, getY, getHeight, getWidth” et vérifie grâce au boolean “overRect” si la souris est bien comprise dans un certain rectangle. La méthode “updateCircle” a exactement le même principe mais renvoie vrai si la souris est comprise dans un certain cercle.
  “mouseReleased()” permet, lorsqu’on lâche le clic de la souris, de parcourir une certaine liste d’instance d’objet et d’effectuer la méthode “switch_” de l’objet.
  La méthode “switch_” permettra alors de savoir, en modifiant un boolean par true,si le bouton a été pressé ou non : versDroite pour “Aiguillage”, isPush pour “buttonMouvement”, estVert pour “Feux”. Pour l’aiguillage, il changera alors de direction et, pour les feux, ils changeront alors de couleurs.
- **ButtonMovement :** les “buttonMouvement” reconnaissable par “←”, “→”, “Enter” sont tous assignés à un index différent, de 0 à 23. Ces boutons appelleront les différents events de la classe Event, Enter et Tracks. Ils permettront de changer les trains de positions sur les voies, de rentrer ou sortir un train d’une gare.

- **Entree**: Cette classe contient les fonctions qui changent automatiquement les feux, celles-ci seront expliquées dans le point suivant.
- **Events**: cette classe contient tous les évènements qui gèrent les entrées et les sorties des trains des gares primaires et secondaires. En ce qui concerne les fonctions de sortie des gares primaires 1 et 3, ils vérifieront vers quelle aiguillage une plateforme est-elle pointée, et ensuite feront sortir le train de cette plateforme si le feu de sortie est vert. La fonction “sortirGare1”, qui sort les trains de la gare principale 1, place ces trains sur la voie P1ToP2, représentant la voie de la gare 1 à la gare 2, alors que la fonction sortirGare3, qui sort les trains de la gare principale 3, place ces trains sur la voie S1ToP3, qui est la voie de la gare secondaire à la gare principale 3. Ainsi, ces deux gares ont aussi 2 feux de sortie, une pour la gauche et l’autre pour la droite. De l’autre côté, la gare principale 2 et la gare secondaire ont toutes les deux 2 fonctions d’entrée, l'une faisant entrer les trains venant de la gare principale 1 et l’autre faisant entrer les trains venus de la gare principale 3. Les fonctions d’entrée, eux, regardent la plateforme de leur gare associée qui sera pointée par les aiguillages. Si le feu de cette plateforme est vert, alors seulement elle pourra rentrer. Comme les fonctions d’entrée, la gare secondaire et la gare primaire 2 ont toutes les deux 2 fonctions de sortie. Similairement aux feux d’entrée, chaque plateforme de ces gares ont aussi 2 feux, une pour la gauche et l’autre pour la droite.
- **Feux**: cette classe prend comme paramètres les coordonnées des différents feux de signalisation et contient une variable locale booléenne appelée “estVert” qui signale si le feu est vert ou rouge. Cette variable est d’office initialisée à true.
  “Feux” contient aussi la fonction “draw()” qui dessine le feu aux coordonnées désirées et le remplis en vert si “estVert” est True et en rouge le cas échant.
- **Track:** cette classe contient les fonctions "draw", "moveTrain" et "dessinTrain". La fonction "dessinTrain" prend en paramètre une voie et dessine les trains qu'il y a dessus, s'il y en a. Les trains 1, qui vont de gauche à droite, sont représentés par des rectangles jaunes, et les trains 2, qui vont de droite à gauche, sont des rectangles rouges.
  La fonction "draw" dessine les 4 voies et appelle la fonction "dessinTrain" pour chacune de ces voies. La fonction "moveTrain" sert à déplacer les trains sur la voie qu'elle reçoit en paramètre. Ainsi, si la voie contient un train 2, alors ce train va être déplacé à gauche, mais si elle contient un train 1, alors ce train sera déplacé à droite. Cela s’effectuera en changeant les valeurs de l'array associé à la voie donnée en paramètre à cette fonction.

5. Explication des feux automatiques

Les feux associés aux quais de la gare deviennent rouges si un train est dans la gare et vert sinon. Pour ce faire, les fonctions “feuxUpdateGare1”, “feuxUpdateGare2”, ”feuxUpdateGare3” et “feuxUpdateSGare” gèrent les feux dans les gares principales 1, 2 et 3 et la gare secondaire respectivement.

Vu que chaque gare est associée à un array d’entiers 1, 2 ou 0, et si un des éléments de cet array est 0, alors la ou les feux associé(s) à la plateforme correspondante seront mis au vert par les fonctions citées précédemment, et en rouge dans le cas échéant.

Le réglage des feux de sortie des gares se fait un peu différemment. Pour les gares principales 1 et 3, les fonctions “gare1sortie” et “gare3sortie” sont utilisées respectivement. La fonction “gare1sortie” regardera si la voie P1ToP2 est vide (son array correspondant contient des 0 uniquement) et qu’au moins une des plateformes de la gare 2 contient une place vide (au moins un des éléments de “pgare2”, l’array associé à cette gare, contient un élément 0).

La fonction “gare3sortie” fonctionne de la même manière sauf qu'elle vérifie si la voie S1ToP3 est vide et que la gare secondaire a une place vide en examinant l'array sgare.

Pour la gare primaire 2, les fonctions "gare2sortieG" et "gare2sortieD" gèrent respectivement le feu de sortie gauche et le feu de sortie droite. La fonction "gare2sortieG" vérifie que la voie P1ToP2 est vide et que la gare 1 contient une place vide, c’est-à-dire que pgare1 a au moins un élément 0. Pendant ce temps, "gare2sortieD" vérifie si la gare 3 a au moins une place vide, c’est-à-dire que "pgare3" contient au moins un élément 0, et que la voie P2ToP3 est vide.

La gestion des feux de sortie dans la gare secondaire est similaire à celle de la gare principale 2, avec les fonctions "sgaresortieG" et "sgaresortieD" gérant respectivement le feu de sortie de gauche et le feu de sortie de droite. Ces deux fonctions fonctionnent de la même manière que "gare2sortieG" et "gare2sortieD", hormis que "sgaresortieG" vérifie si la voie "P1ToS1" est vide et que la gare 1 a au moins une place vide, tandis que "sgaresortieD" vérifie si au moins une place vide est disponible dans la gare 3, et que la voie "S1ToP3" est vide.

6. Problèmes rencontrés et informations supplémentaires

Le problème majeur rencontré a été lors de l'utilisation du plugin eventb2java. Durant la génération du code java, nous avions des "no_type" qui apparaissait dans notre code et on ne savait pas d'où cela venait. Nous avons essayé de régler le problème mais sans succès. Nous avons donc refait la machine nous-même en java sans utiliser ce plugin ce qui nous a fait perdre du temps dans notre planning. Nous avons quand même essayé de nous rapprocher au plus proche de notre animation event b.
Un autre problème fût la taille de notre groupe, il était parfois difficile de travailler tous en même temps sur le projet, nous nous sommes donc répartis les tâches et attendions que chacun fasse la sienne en lui procurant de l'aide si besoin.
Nous avons utilisé github pour notre projet.

7. Conclusion

En conclusion, dans cette partie processing, nous avons appris le langage java sur Processing et, pour ceux dans le groupe qui en avait déjà la connaissance, nous avons pu étendre les différentes connaissances sur Java et avoir un aperçu différent de ce que l'on peut faire dans ce langage. Ce projet a été enrichissant pour chacun d'entre nous, il a permis également d'améliorer notre maîtrise dans un travail de groupe, à mieux communiquer et coder en fonction des autres, comprendre ce que l'autre fait et savoir s'adapter à chacun. Nous espérons, à travers ce projet, vous avoir transmis nos différents efforts et de vous avoir montrer ce que l'on a pu apprendre durant ce projet pendant ce quadrimestre. Merci d'avoir lu ce rapport.