Rapport du rendu2

Répartition des responsabilités

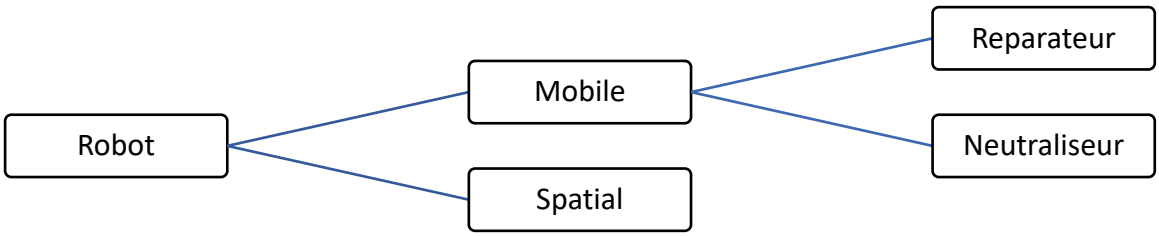
Le module `projet` créée l'instance `Simulation`, qui ne sera jamais supprimée mais uniquement modifiée jusqu'à la fermeture de la fenêtre. Ensuite, projet créée une instance de `Gtk::Window` qu'il ouvre. Le module `gui` contenant le constructeur de `Gtk::Window` construit la fenêtre, gère les interactions avec les boutons, le timer et les actualisations suivantes :

- `gui` ordonne à `Simulation` d'actualiser ses données avec `update`. `Simulation` transmet l'ordre aux modules `robot` et `particule`, qui actualisent le dessin en passant par `shape` et `graphic`.
- `gui` ordonne à `graphic` de dessiner un carré blanc et transmet le pointeur à celui-ci
- `gui` met à jour les valeurs des labels si la simulation change.

Le module `simulation` gère l'instance `Simulation` complète, avec ses données dans des Vecteurs.
Lors d'une détection d'erreur, les données de `Simulation` sont mises à 0, de même pour son attribut `Spatial`.

`graphic` n'effectue finalement que des opérations très simples, telles que dessiner un carré, un rond, une ligne, etc. On ne lui transmet jamais plusieurs commandes (par exemple dessiner un `Neutraliseur`), mais des commandes séparées (dessiner le cercle de `Neutraliseur`, dessiner sa ligne, dessiner son centre).

Hiérarchie des classes des robots



La superclasse `class Robot` ne contient que la méthode virtuelle pure `void draw()` qui est héritée et redéfinie par toutes ses sous-classes.

Structuration des données des autres entités

Simulation possède les attributs suivants.

```
int nbP_;
vector<Neutraliseur> neutraliseurs_;
vector<Reparateur> reparateurs_;
vector<Particule> particules_;
Spatial spatial_;
bool dessiner_;
```

En passant comme attributs de Simulation les différentes listes de tous les objets de la simulation, on peut y avoir accès facilement depuis toutes les méthodes de Simulation (ainsi

que dans Gui, qui est de plus haut niveau), comme pour mettre à jour la Simulation depuis son module ou actualiser les statistiques de la fenêtre depuis Gui.

Particule possède les attributs suivants :

```
Carre forme_;
```

La forme Carre est décrite ci-dessous dans les types de Shape.

Types dans Shape

```
enum Etat { NBP, PARTICULE, SPATIAL, REPARATEUR, NEUTRALISEUR };
```

Cet enum Etat sert à repérer la progression de la lecture de fichier dans Simulation.
Les positions sont désignées par `struct S2d` et les formes carrées (Particules) sont utilisées par `struct Carre` et les formes cercle (Neutraliseurs, Réparateurs et Spatial) par `struct Cercle.` `Carre` et `Cercle` contiennent les deux une position avec un rayon ou une largeur de côté.