Projet : DSP calculator

1 Enoncé du projet

Les jeux de type *factory builder* sont apparus aux environs de 2015 avec la sortie de *Factorio*, lui-même inspiré d'un mod de *Minecraft* cherchant à automatiser la récupération et la transformation de ressources. L'objectif de ces jeux est de mettre en place des chaînes de production complexes permettant, à partir de ressources primaires, d'effectuer toute une série de transformation afin d'obtenir un produit final spécifique (une fusée par exemple).

Pour y parvenir, il faut organiser ses chaînes de production et veiller à ce qu'aucune rupture d'approvisionnement ne survienne. Il faut donc, à chaque étape, calculer la quantité de ressources nécessaires pour la production d'un produit spécifique, et veiller à ce que l'approvisionnement, en amont, de ces ressources soit assuré. Afin d'aider le joueur dans cette tâche, des programmes de calcul automatiques existent.

Dyson Sphere Program est un jeu de type *factory builder* où l'objectif est de construire une sphère de Dyson autour d'une étoile (il s'agit d'une immense sphère construite autour d'une étoile servant à récupérer l'intégralité de l'énergie produite par cette dernière). Pour cela, le joueur peut aller de planète en planète afin d'y récupérer des ressources et y construire des chaînes de production de plus en plus complexes. Les usines consomment de l'électricité, et il est important de toujours veiller à ce que la production électrique sur ses planètes soit suffisante pour alimenter ses infrastructures.

Vous devrez, dans ce petit projet, utiliser les données contenues dans un fichier XML joint à ce projet. Pour vous aider à visualiser et comprendre les données, vous pouvez utiliser un site web comme `https://jsonformatter.org/XML-viewer` afin de visualiser le fichier XML sous forme d'arbre.

De plus, un code Java vous permettant de lire le fichier XML pour en afficher une partie du contenu est joint aux fichiers de ce projet. Servez-vous de ce code afin de construire la partie de votre programme qui lira le fichier pour en extraire des informations.

2 Composants : les objets "de base" dans le jeu et dans le fichier

Les composants sont la brique de base du jeu. Un composant est n'importe quel objet pouvant être extrait, manipulé ou créé. Toute chose extraite ou produite dans le jeu est un composant. Dans le fichier XML, ces composants sont repérés par le tag **items** avec, comme **category**, la valeur **components**. L'objet *stone-brick* est un exemple de composant simple.

```
<items>
  <category>components</category>
  <id>stone-brick</id>
  <name>Stone brick</name>
  <row>1</row>
  <stack>100</stack>
</items>
```

Un composant possède un nom (tag **name**) et un nom d'identité unique (tag **id**) servant à le référencer, si nécessaire, dans d'autres parties du fichier. Ces deux informations sont les seuls éléments que l'on souhaitera extraire d'un composant "simple", et on ignorera les autres tags qui ne nous serviront pas.

Il existe cependant des composants avec des propriétés supplémentaires.

2.1 Les ressources

Les composants sont en général produits à partir d'autres composants dans une usine, mais certains peuvent également être extraits des ressources naturelles d'une planète à l'aide d'un extracteur. Dans ce dernier cas, on dira que le composant est une ressource. Une ressource est donc un composant, et est

identifiée, dans le fichier XML, par le tag **items** avec, comme **category**, la valeur **resource**. L'objet *water* est un exemple de ressource :

```
<items>
  <category>resource</category>
  <id>water</id>
  <name>Water</name>
  <row>0</row>
  <stack>20</stack>
  <minedby>water-pump</minedby>
</items>
```

En plus du nom et du nom d'identité, on s'intéressera à l'objet ou les objets permettant d'extraire cette ressource. Le (ou les) tag **minedby** permet de connaître le nom d'identité de l'extracteur (nous y reviendrons plus tard) permettant d'extraire la ressource.

2.2 Les carburants

Comme il avait été écrit précédemment, la gestion de la production électrique est importante dans le jeu. Un moyen de produire de l'électricité est d'utiliser des carburants dans une centrale. Certains composants peuvent être des carburants, comme par exemple l'objet **hydrogen** ou l'objet **graphite** :

```
<items>
  <category>components</category>
  <id>graphite</id>
  <name>Energetic graphite</name>
  <row>1</row>
  <stack>100</stack>
  <fuel>
    <category>chemical</category>
    <value>6750</value>
  </fuel>
</items>
```

```
<items>
  <category>resource</category>
  <id>hydrogen</id>
  <name>Hydrogen</name>
  <row>0</row>
  <stack>20</stack>
  <minedby>oil-extractor</minedby>
  <minedby>orbital-collector</minedby>
  <fuel>
    <category>chemical</category>
    <value>9000</value>
  </fuel>
</items>
```

Le tag **fuel** permet de savoir si un composant est un carburant. Dans ce cas, on s'intéressera, à l'intérieur du tag **fuel**, au tag **value** permettant d'obtenir le rendement énergétique du carburant, en kW/s (nous y reviendrons plus tard) ainsi qu'au tag **category** permettant de connaître la catégorie du carburant. La catégorie d'un carburant peut être **chemical**, **nuclear** ou **antimatter**.

Les carburants sont transformés en électricité dans des centrales, et nous reviendrons plus tard sur ces dernières ainsi que sur les calculs permettant de connaître la quantité d'énergie produite.

3 Les bâtiments

Les bâtiments sont considérés comme des composants, mais pour rendre les explications de ce projet plus claires, nous avons souhaité leur dédier une section à part entière. Les bâtiments sont des constructions permettant d'effectuer des tâches comme produire des composants, extraire des ressources, ou

consommer du carburant pour produire de l'électricité. Dans le fichier XML, les bâtiments sont repérés par le tag **items** avec, comme **category**, la valeur **buildings**. L'objet *tesla-coil* est un exemple de bâtiment simple.

```
<items>
  <category>buildings</category>
  <id>tesla-coil</id>
  <name>Tesla tower</name>
  <row>0</row>
  <stack>100</stack>
</items>
```

Les bâtiments simples dans le jeu permettent d'effectuer des tâches de base, comme transmettre de l'électricité vers un autre bâtiment, transporter des composants, etc. Il existe cependant une classe de bâtiments très importante, et majoritaire, dans le jeu : les usines.

Les usines sont des bâtiments permettant de produire quelque chose, soit par extraction, transformation ou consommation. Nous allons détailler ci-après les différentes usines disponibles dans le jeu.

3.1 Les usines classiques

Une usine classique est un bâtiment permettant de transformer des composants en d'autres composants, tout en consommant de l'électricité. Dans le fichier XML, les usines classiques sont des bâtiments possédant le tag **factory** et, à l'intérieur de ce dernier, le tag **type** vaut **electric**. L'objet *assembler-1* est un exemple d'une telle usine :

```
<items>
  <category>buildings</category>
  <id>assembler-1</id>
  <name>Assembling machine mk.I</name>
  <row>3</row>
  <stack>50</stack>
  <factory>
    <speed>0.75</speed>
    <type>electric</type>
    <usage>270</usage>
    <drain>12</drain>
    <modules>1</modules>
  </factory>
</items>
```

On s'intéressera, dans une usine, à sa vitesse de production ainsi que sa consommation électrique. La vitesse de production est obtenue par le tag **speed** dans le tag **factory**. Si ce tag n'est pas présent, la vitesse vaudra par défaut 1.

3.1.1 Consommation électrique d'une usine

La consommation électrique d'une usine classique est obtenue par le tag **usage** dans le tag **factory**. Si l'usine n'est pas en cours d'utilisation (par exemple, par manque de matériel pour produire ce qu'elle doit produire), elle consomme un petit montant d'électricité quand même, dont la valeur est donnée par le tag **drain** dans le tag **factory**. Toutes ces valeurs sont données en kW/s.

3.2 Les extracteurs

Un extracteur est une usine qui produit une ressource en allant directement piocher dans les ressources de la planète. L'objet *oil-extractor* est un exemple d'extracteur :

```
<items>
  <category>buildings</category>
  <id>oil-extractor</id>
  <name>Oil extractor</name>
  <row>2</row>
  <stack>10</stack>
  <factory>
```

```
    <type>electric</type>
    <usage>840</usage>
    <drain>24</drain>
  </factory>
  <mining>
    <speed>120</speed>
  </mining>
</items>
```

Dans le fichier XML, les extracteurs sont des usines possédant un tag **mining** avec un tag **speed** à l'intérieur. Cette valeur de vitesse représente le nombre d'unités d'une certaine ressource pouvant être extraite à chaque minute. Veuillez noter que le tag **speed** n'est, dans ce cas, pas présent dans le tag **factory**. On s'intéressera également à la consommation électrique d'un extracteur, qui repose sur les mêmes principes que la consommation électrique d'une usine classique.

3.2.1 Certains extracteurs ne consomment pas d'électricité

Certains extracteurs ne consomment pas d'électricité et, dans cas cas, ils ne possèdent pas le tag **factory**. C'est le cas de l'objet *orbital-collector* :

```
<items>
  <category>buildings</category>
  <id>orbital-collector</id>
  <name>Orbital collector</name>
  <row>1</row>
  <stack>10</stack>
  <mining>
    <speed>8</speed>
  </mining>
</items>
```

3.3 Les centrales électriques

Les centrales électriques sont des usines produisant de l'électricité pour alimenter les autres bâtiments du jeu. Dans le fichier XML, elles apparaissent avec le tag **factory** et, à l'intérieur, le tag **type** vaut **burner** (centrale à carburant) ou **electric-production** (centrale à énergie renouvelable). Nous décrivons ci-après ces deux types de centrales.

3.3.1 Les centrales à carburant

L'objet *fusion-reactor* est un exemple de centrale à carburant :

```
<items>
  <category>buildings</category>
  <id>fusion-reactor</id>
  <name>Artificial star</name>
  <row>0</row>
  <stack>10</stack>
  <factory>
    <type>burner</type>
    <category>antimatter</category>
    <speed>1</speed>
    <value>72000</value>
  </factory>
</items>
```

Le tag **value** permet de connaître la capacité de production (en kW/s) de la centrale. Les centrales à carburant peut avoir différents catégories, représentées dans le tag **category** : **chemical**, **nuclear** ou **antimatter**. Ces catégories correspondent aux catégories de carburants pouvant être donnés en entrée des centrales pour être consommés. Une centrale à carburant d'une certaine catégorie ne pourra recevoir, en entrée, que des carburants de la même catégorie.

Pour connaître la consommation en carburant par seconde d'une centrale, on regarde le rendement énergétique du carburant et on le divise par la capacité de production de la centrale.

3.3.2 Les centrales à énergie renouvelable

L'objet *wind-turbine* est un exemple de centrale à énergie renouvelable :

```
<items>
  <category>buildings</category>
  <id>wind-turbine</id>
  <name>Wind turbine</name>
  <row>0</row>
  <stack>20</stack>
  <factory>
    <type>electric-production</type>
    <value>300</value>
  </factory>
</items>
```

Ces centrales utilisent les ressources renouvelables d'une planète (chaleur, vent, ...) pour produire de l'énergie. Elles n'ont donc pas de carburant en entrée. Le tag **value** permet de connaître la capacité de production (en kW/s) de la centrale.

4 Les recettes

Les recettes sont caractérisées par le tag **recipes**. Elles expliquent comment produire certaines composants à partir d'autres composants, dans une usine. La plupart des recettes produisent un seul élément en sortie, mais certaines peuvent produire plusieurs éléments.

4.1 Les recettes produisant un seul élément

Voici un exemple de recette ne produisant qu'un seul élément :

```
<recipes>
  <id>oil-extractor</id>
  <name>Oil extractor</name>
  <in>
    <steel-plate>12</steel-plate>
    <stone-brick>12</stone-brick>
    <circuit-board>6</circuit-board>
    <plasma-generator>4</plasma-generator>
  </in>
  <time>8</time>
  <producers>assembler-1</producers>
</recipes>
```

Le type de composant produit par la recette est donné par le tag **id** de la recette. On s'intéressera également aux ingrédients de la recette, repérés par le tag **in** de la recette. Dans ce tag, chaque sous tag représente le nom d'identité d'un composant en entrée de la recette, et sa valeur représente la quantité nécessaire de ce composante pour la recette.

Le tag **time** représente le temps de production, en seconde, de la recette et le tag **producers** représente le nom de l'usine permettant de produire cette recette.

4.2 Les recettes produisant plusieurs éléments

Voici un exemple de recette produisant plusieurs éléments en sortie :

```
<recipes>
  <id>plasma-refining</id>
  <name>Plasma refining</name>
  <in>
    <oil>2</oil>
```

```
  </in>
  <out>
    <hydrogen>1</hydrogen>
    <refined-oil>2</refined-oil>
  </out>
  <time>4</time>
  <producers>oil-refinery</producers>
</recipes>
```

Le tag **out** représente la liste des éléments produits par la recette, dans le même format que pour la liste des éléments en entrée de la recette. Le tag **id** représente le nom d'identité de la recette, qui ne correspond plus au nom d'identité d'un composant.

5 Ce qui est à faire

Vous devrez proposer un programme Java permettant de lire le fichier XML, et construire une structure de classes bien pensée afin de proposer les fonctionnalités suivantes :

- Générer une exception :
  - dans le cas où un bâtiment électrique consommerait plus "au repos" qu'il ne le fait "au travail".
  - dans le cas où une centrale à carburant ou un carburant n'aurait pas l'une des catégories attendues.
  - dans le cas où une centrale à carburant serait associée à un carburant d'un autre type que la centrale.
- Afficher la liste des composants non bâtiment par ordre alphabétique et permettre, pour un composant spécifique, d'obtenir toutes ses informations.
- Afficher la liste des bâtiments par ordre alphabétique et permettre, pour un bâtiment spécifique, d'obtenir toutes ses informations.
- Afficher la liste des recettes par ordre alphabétique et permettre, pour une recette spécifique, d'obtenir toutes ses informations.
- Afficher, pour une usine donnée, toutes les recettes qui lui sont associées (vous devez afficher les informations complètes de chaque recette).
- Afficher, pour un extracteur donné, toutes les ressources qu'il peut extraire (vous devez afficher les informations complètes de chaque ressource).
- Afficher, pour une centrale à carburant donnée, tous les carburants qu'elle peut prendre en entrée (vous devez afficher les informations complètes de chaque carburant).
- Afficher, pour une recette spécifique, l'ensemble des composants de type ressource qui sont nécessaires pour l'élaboration de la recette. Il faudra, pour chaque composant de la recette qui n'est pas une ressource, rechercher de façon récursive comment produire ce composant jusqu'à obtenir une décomposition en ressources. Si des composants peuvent être produits par plusieurs recettes, on choisira arbitrairement une des recettes.

Si vous avez le temps, voici des fonctionnalités supplémentaires que vous pourriez proposer :

- Afficher, pour une recette donnée, la consommation électrique totale requise par cette recette à partir des ressources de base.
- Afficher, pour une recette donnée, le temps de production minimum d'un composant produit par la recette à partir uniquement des ressources de base. Le temps de production d'un composant est égal au temps de production de la recette divisé par la vitesse de l'usine produisant la recette. Comme on souhaite avoir le temps de production minimum, on imaginera avoir autant d'usine et d'extracteurs que nécessaire afin de paralléliser un maximum la production.
- Permettre à l'utilisateur de spécifier au programme ses centrales électriques (et les carburants associés quand c'est nécessaire), et lui afficher sa production électrique par seconde, ainsi que la consommation de carburant (par seconde) qui en résulte.
- Proposer une interface graphique à votre logiciel. Un fichier icône est disponible dans les fichiers joints à ce projet.

Vous devrez également **rédiger** un court rapport expliquant votre diagramme de classe, vos choix d'implémentation, les extraits de code vous paraissant intéressants, et tout autre détail vous semblant important. N'oubliez de consulter le guide de rédaction des rapports.