

---

Projet individuel innovant

# Génération de données domotique à base d'agents avec NetLogo

Alexis Szmundy

---



---

# Sommaire

<b>Introduction.....</b>	<b>5</b>
<b>Objectifs du projet.....</b>	<b>6</b>
<b>Organisation.....</b>	<b>8</b>
Méthodologie.....	8
Outils.....	9
Ticketing.....	9
Contrôle du code source.....	11
Edition de texte.....	13
Visualisation des données.....	14
<b>Cadrage/Concept.....</b>	<b>16</b>
Obtention du besoin.....	16
Brainstorming.....	17
Développement des idées.....	18
<b>Déroulé du développement.....</b>	<b>19</b>
Environnement.....	19
Appartement.....	19
Température.....	22
Luminosité.....	22
Mobilier.....	22
Capteurs.....	23
Roomba et système de propreté des pièces.....	23
Utilisateur.....	23
Export des données.....	24
Visualisation.....	24
<b>Description du modèle du livrable.....</b>	<b>25</b>
Initialisation.....	26
Monde.....	29
Exécution d'un tick.....	30
Variables globales.....	36
Date.....	36
Température.....	39
Initialisation.....	39
Extérieur simulation.....	41
Intérieur simulation.....	43

---

Luminosité.....	47
Extérieur initial.....	47
Intérieur initial.....	51
Simulation.....	52
Saleté.....	60
Agents.....	61
Patches.....	61
Meubles.....	62
Communs.....	62
Salle de bain.....	63
Chambre.....	64
Salle à manger.....	65
Cuisine.....	65
Objets.....	68
Roomba.....	70
Capteurs.....	71
Luminosité.....	71
Température.....	71
CO/Fumée.....	71
Données.....	72
Déclenchement.....	72
Ouverture.....	72
Mouvement.....	72
Pathfinding.....	73
Links.....	74
Liens agents-capteurs.....	74
Liens contenance.....	74
Liens portance.....	75
Liens tâches.....	75
Utilisateurs.....	76
Routine.....	76
Tâches.....	77
Besoins.....	77
Système de tâches.....	78
Export des données.....	79
Interface du modèle.....	81
<b>Visualisation des données.....</b>	<b>82</b>
<b>Problèmes rencontrés.....</b>	<b>84</b>
Limites de Netlogo.....	84

---

---

Limites de Looker Studio.....	85
Ambition.....	86
<b>Axes d'amélioration du modèle.....</b>	<b>87</b>
Personnalisation de l'environnement.....	87
Configurabilité.....	87
Maintenabilité.....	87
Assumptions et réalisme.....	87
Idées mises de côté faute de temps.....	88
<b>Références et inspirations.....</b>	<b>89</b>
<b>Conclusion.....</b>	<b>90</b>

---

## Introduction

La création d'une représentation virtuelle d'un objet physique pouvant émettre des données dans un contexte d'internet des objets offre de vastes opportunités pour la modélisation et la simulation. Ce projet a pour objectif principal de générer artificiellement des données en modélisant et en simulant le résident d'un logement équipé de capteurs.

Ce rapport s'articule autour de différents aspects du projet. Dans un premier temps nous présenterons l'objectif global du projet ainsi que la façon dont il a été organisé. Ensuite nous détaillerons la phase de conception de ce projet qui a abouti aux différentes fonctionnalités, puis nous exposerons le déroulé du développement de ce modèle, en décrivant les différentes étapes mises en œuvre pour le réaliser. S'en suivra d'une description détaillée du fonctionnement de ce modèle ainsi qu'un exemple d'exploitation des données dans un outil de visualisation. Nous aborderons également les problèmes rencontrés au cours du projet, mettant en évidence les difficultés et les éventuels obstacles auxquels nous avons dû faire face tout du long du développement. Par la suite, nous identifierons les axes d'amélioration potentiels pour le modèle sur différents aspects. Enfin, nous fournirons une liste de références et d'inspirations ayant guidé notre travail, avant de conclure sur les résultats obtenus et les perspectives futures.

---

## Objectifs du projet

L'idée au cœur de ce projet individuel innovant consiste à modéliser trois pièces spécifiques d'un appartement, à savoir la cuisine-salon-chambre, la salle de bain et l'entrée, le tout en y ajoutant un écosystème de capteurs. Lesdits capteurs peuvent être de différents types, tels que des capteurs d'ouverture de porte, de présence, de température, ou encore de déclenchement. Le but du projet est de pouvoir simuler les déplacements d'une personne résidant dans cet appartement et les interactions de celle-ci avec les objets connectés autour d'elle.

Chaque interaction de la personne avec un objet équipé d'un capteur doit déclencher la production d'une donnée correspondante. Pour ce faire, l'ensemble des données générées sera sauvegardé dans un fichier de type CSV, permettant ainsi leur exploitation ultérieure et ce de manière portable. Les capteurs ainsi que la personne seront modélisés par des agents, et la simulation sera réalisée en utilisant le logiciel NetLogo, un outil puissant et flexible pour la modélisation et la simulation d'agents. Les agents sont des entités autonomes et interactives qui peuvent se déplacer, interagir les uns avec les autres et avec leur environnement. La simulation sera mise en œuvre pour représenter de manière dynamique et réaliste les interactions entre ces agents et l'environnement du logement.

---

Pour résumer, les objectifs du projet sont les suivants :

- Modéliser trois pièces spécifiques d'un appartement (cuisine-salon-chambre, salle de bain, entrée) en prenant en compte leurs caractéristiques individuelles et leur mobilier.
- Intégrer un ensemble de capteurs (ouvertures de portes, présence, température, utilisation) dans les pièces modélisées.
- Simuler les déplacements d'une personne dans cet appartement et prendre en compte ses interactions avec les objets connectés présents.
- Déclencher la production de données correspondantes à chaque interaction de la personne avec un capteur spécifique.
- Sauvegarder l'ensemble des données générées dans un fichier texte ou tableau au format CSV/XLS.

---

## Organisation

Pour garantir le bon déroulement de ce projet, nous avons mis en place une organisation rigoureuse tout au long de sa réalisation. Nous avons adopté une méthodologie spécifique et utilisé divers outils pour faciliter notre organisation.

## Méthodologie

Avant toute chose, il était nécessaire de déterminer notre orientation. Pour ce faire, nous avons entamé une phase de cadrage qui a débuté par des réunions de brainstorming, lors desquelles toutes nos idées ont été consignées. En gardant constamment à l'esprit l'objectif du projet, à savoir générer des données cohérentes. Nous avons ensuite développé en détail chacune de ces idées, elles ont été enregistrées sous forme de tickets, prêts à être utilisés lors des futurs développements.



---

## Outils

Pour réaliser notre projet nous avons utilisé différents outils:

### Ticketing

Un outil vital dans l'organisation de nos idées est **Jira**, un outil d'organisation de projets. L'utilisation de Jira s'est avérée cruciale pour structurer nos idées. Grâce à cet outil, nous avons créé des **tickets** que nous pouvons remplir avec des commentaires, des fichiers et des liens avec d'autres tickets. Ces tickets sont ensuite organisés dans un **tableau** personnalisé, qui comprend les états de base tels que "À faire", "En cours" et "Terminé". Afin d'adapter ce tableau à notre phase de cadrage, nous y avons ajouté les états spécifiques à cette étape.

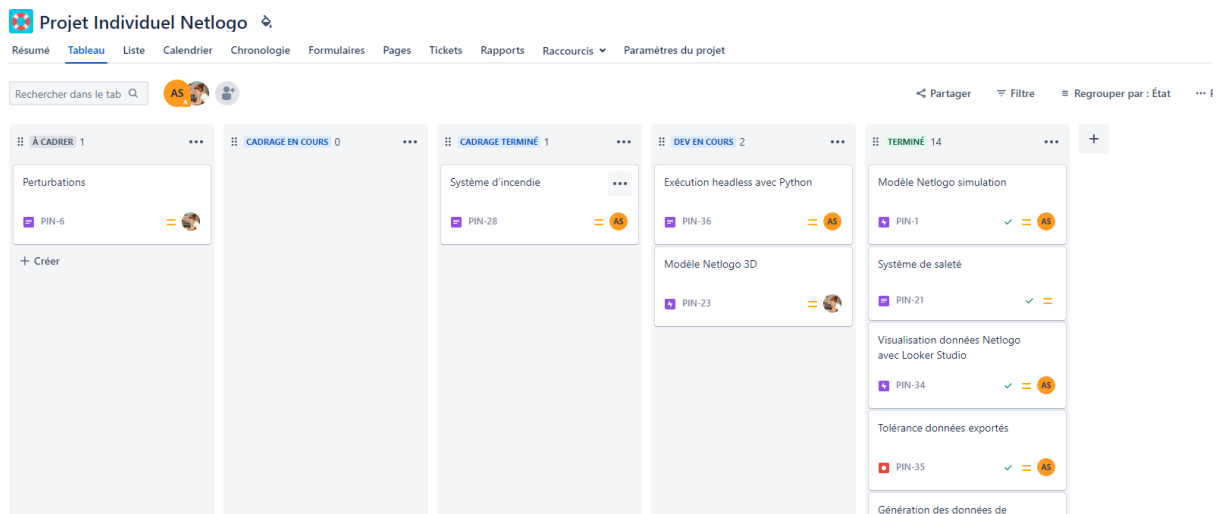


Tableau Jira du projet

PIN-1 / PIN-24

## Comportement des utilisateurs

Joindre Ajouter un ticket enfant Associer un ticket

Description

### Système de tâches

1 Tâche = Link entre Utilisateur et un objet

Utiliser **out-tache-to <cible>** pour générer la tâche

Utiliser **my-taches** pour récupérer toutes les tâches de l'utilisateur

Utiliser **other-end** pour récupérer la cible de la tâche

Attributs du Link tâche :

Attribut	Description
Priorité	Nombre indiquant la priorité de la tâche
Ordre	Ordre de la tâche
Type de tache/Interaction	Action de la tache

PIN-1 / PIN-5

8	0	Va au travail
17	0	Rentre

Flexibilité via le système de tâche

Tickets associés

relates to

PIN-24 Comportement des utilisateurs AS TERMINÉ

Activité

Afficher : Tout Commentaires Historique

Les plus récents d'abord

- AS Ajouter un commentaire...

Conseil de pro : appuyez sur M pour commenter

AS Alexis Szmundy la semaine dernière  
voir fichier routine.csv et partie génération de tâches par routine dans modèle  
Modifier · Supprimer

AS Alexis Szmundy 3 mars 2023 à 10:04  
<http://ccl.northwestern.edu/netlogo/docs/dict/file-open.html>  
Modifier · Supprimer

1 1 1 1 1

Terminé

Terminé

Actions

Détails

Responsable AS Alexis Szmundy

Étiquettes Aucun

Date d'échéance Aucun

Start date Aucun

Rapporteur AS Alexis Szmundy

Création 28 février 2023 à 14:27

Mis à jour il y a 5 jours

Résolu il y a 5 jours

Configurer

1 1 1 1 1

Terminé

Actions

Détails

Responsable AS Alexis Szmundy

Étiquettes Aucun

Date d'échéance Aucun

Start date Aucun

Rapporteur AS Alexis Szmundy

Création 27 février 2023 à 08:38

Mis à jour la semaine dernière

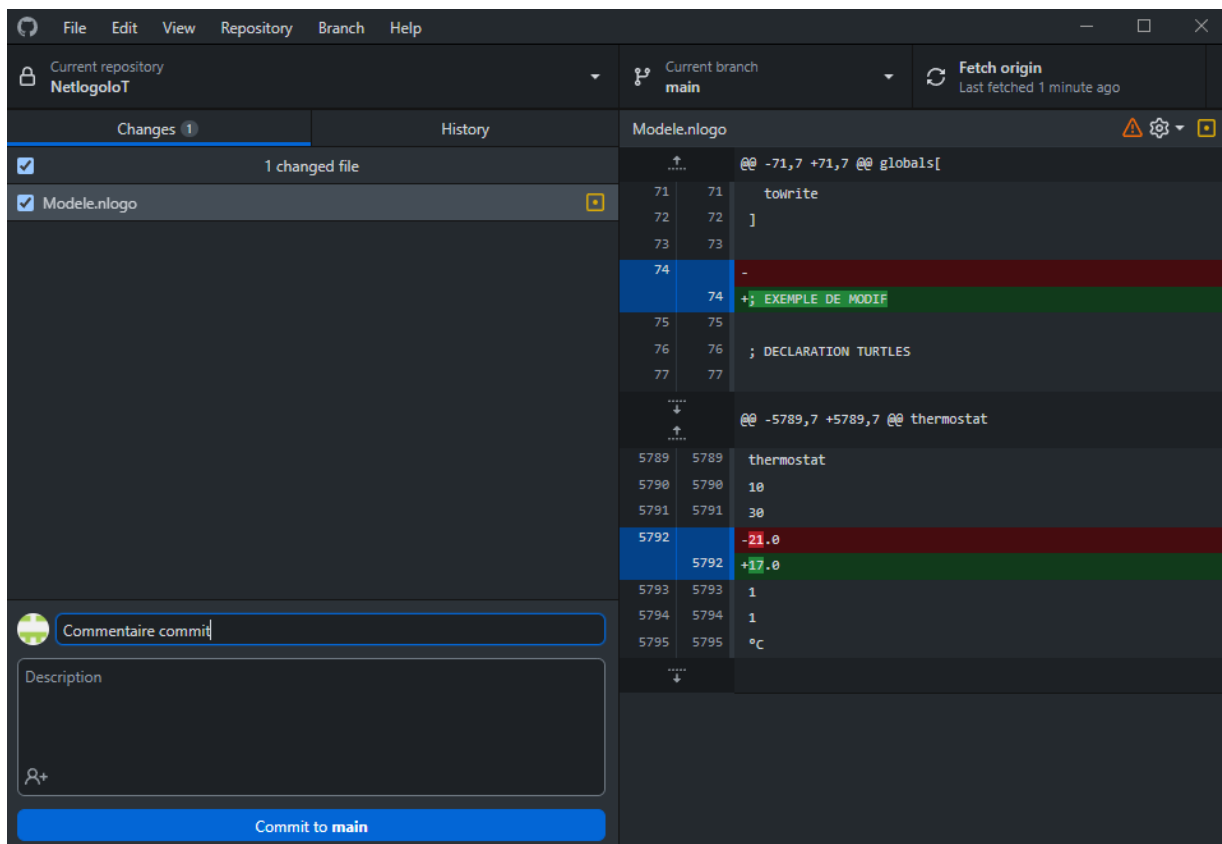
Configurer

Exemples de tickets utilisés lors du développement

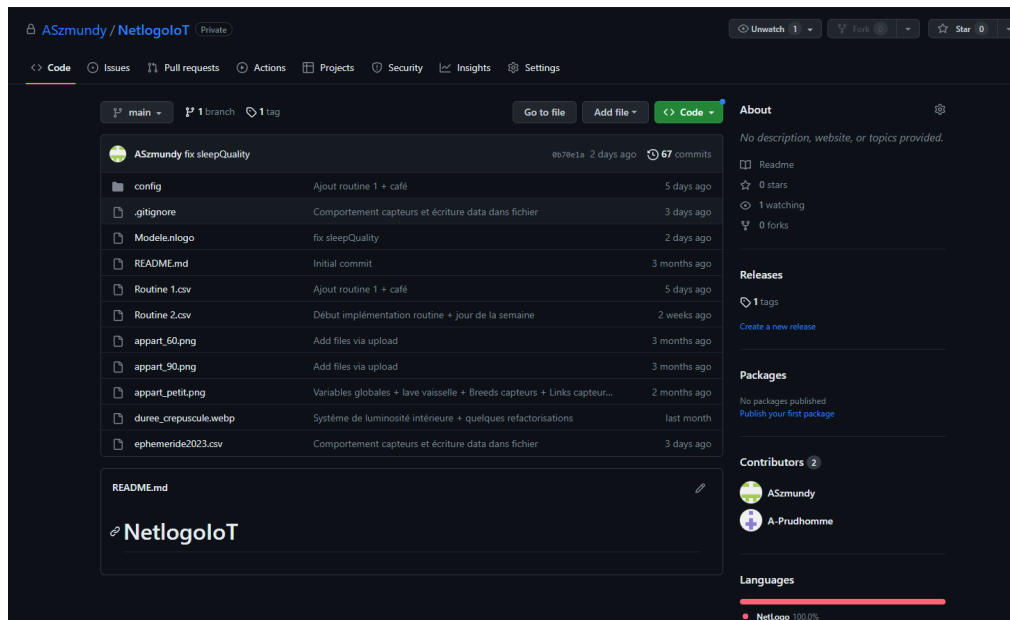
---

## Contrôle du code source

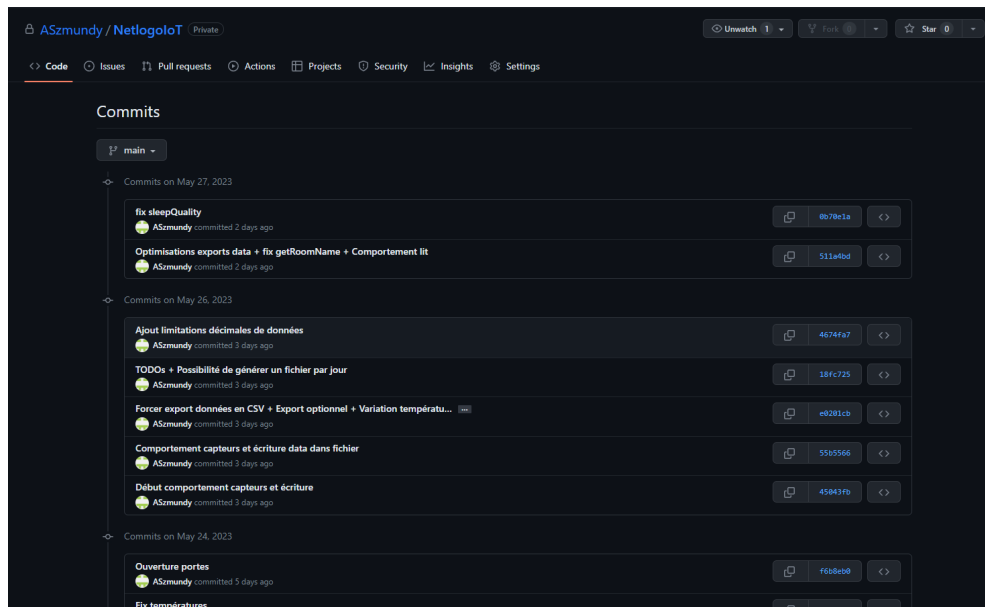
Pour assurer la portabilité du projet lors de la phase de développement, nous avons choisi d'utiliser **GitHub**. Nous avons utilisé **Git Bash** pour exécuter des commandes Git sur Windows, et également **GitHub Desktop**, une interface spécialement conçue pour faciliter l'utilisation de GitHub. Chaque modification était enregistrée via un **commit** avec un commentaire descriptif des changements effectués, puis poussée (**push**) vers un **repository** dédié au projet.



Github Desktop



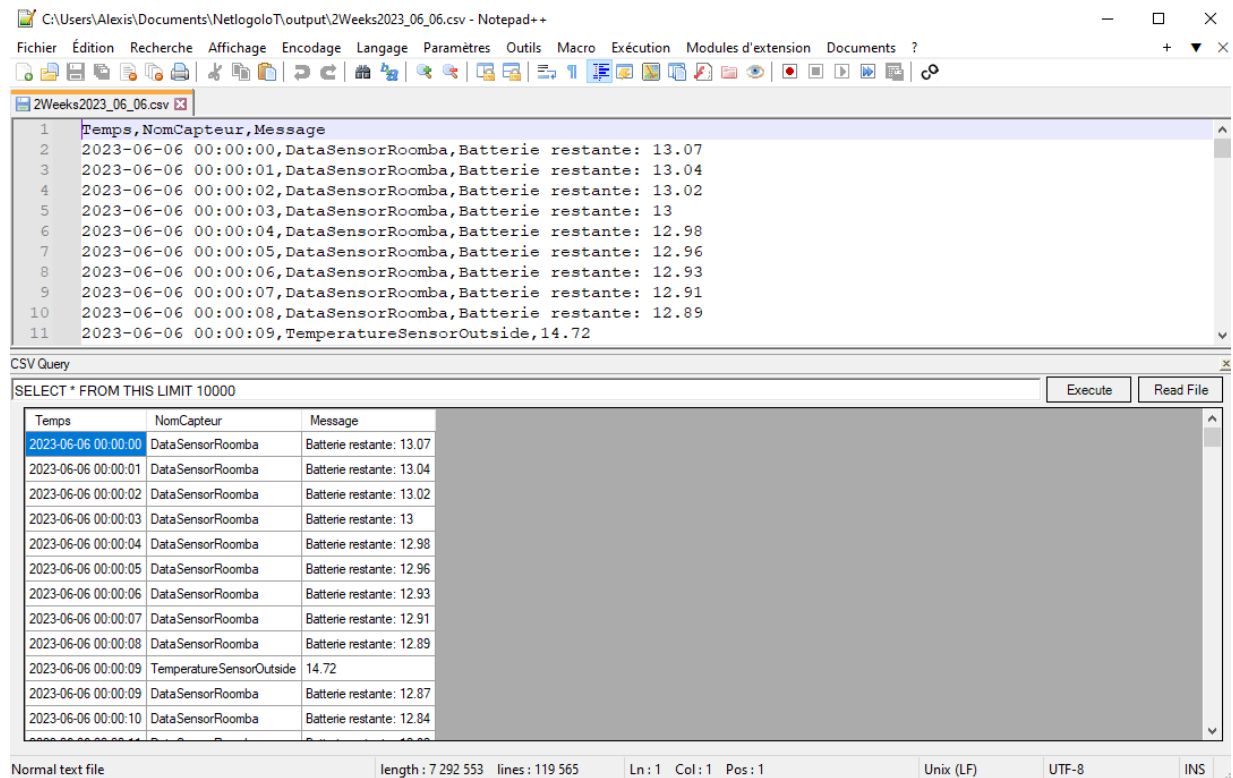
## Repository du projet sur Github



## Commits réalisés dans le cadre du projet

## Edition de texte

Afin de simplifier l'édition des fichiers plats, surtout les fichiers d'exports de données qui peuvent contenir des centaines de milliers de lignes, nous avons utilisé **Notepad++**, un outil puissant dédié à cette tâche. Nous avons également exploité l'extension **CSVQuery**, qui nous a permis de naviguer dans les fichiers CSV comme si nous étions en train de travailler avec une base de données SQL. Cette fonctionnalité s'est avérée particulièrement pratique pour filtrer et visualiser les colonnes du fichier.

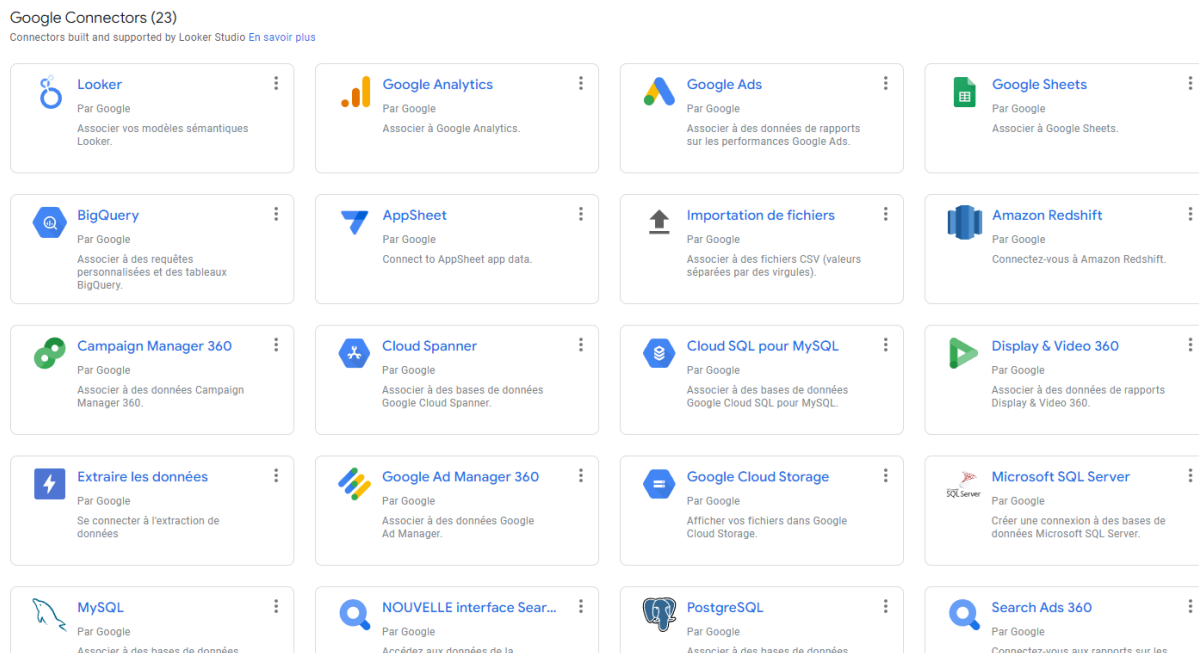


Utilisation de Notepad++ avec CSVQuery sur un fichier de données

---

## Visualisation des données

Enfin, pour visualiser les données générées par le projet, nous avons fait appel à **Looker Studio** (anciennement connu sous le nom de Google Data Studio, à ne pas confondre avec Looker qui est un outil plus avancé intégré à Google Cloud Platform). Ce puissant outil nous a permis de créer des **tableaux de bord** en filtrant les données, à partir de différentes sources telles que des bases de données SQL ou NoSQL, des fichiers plats (ce qui était le cas pour notre projet), ainsi que des outils de campagne marketing, et bien plus encore.



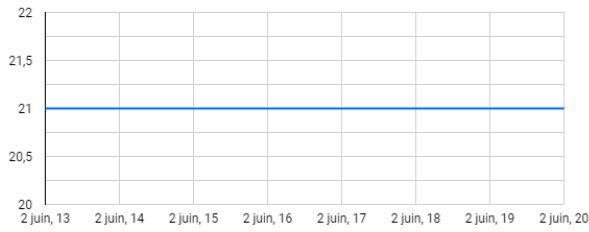
Échantillon de connecteurs possibles avec Looker Studio

Sélectionner la période

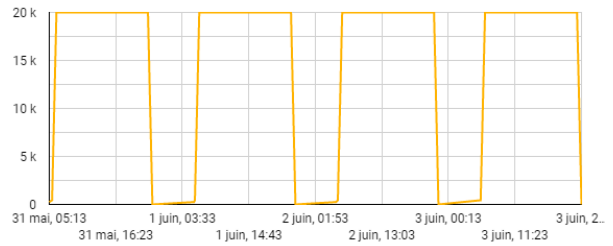
# Entrée



Température



Luminosité



Temps

2 juin 2023, 05:21:22  
1 juin 2023, 05:31:42  
1 juin 2023, 05:30:50  
31 mai 2023, 05:13:19

1 - 4 / 4



Temps

Temps	Porte entre	et
3 juin 2023, 21:36:53	Entrance	DiningRoom
3 juin 2023, 21:36:51	Bathroom	Entrance
3 juin 2023, 19:59:27	Bathroom	Entrance
3 juin 2023, 19:59:25	Entrance	DiningRoom
3 juin 2023, 13:11:54	Entrance	DiningRoom
3 juin 2023, 12:12:01	Entrance	DiningRoom
3 juin 2023, 12:11:34	Entrance	DiningRoom
3 juin 2023, 11:11:40	Entrance	DiningRoom

1 - 48 / 48



Temps

3 juin 2023, 19:59:26  
3 juin 2023, 13:11:53  
3 juin 2023, 13:11:52  
3 juin 2023, 13:11:51  
3 juin 2023, 12:12:05  
3 juin 2023, 12:12:04  
3 juin 2023, 12:12:03  
3 juin 2023, 12:11:33

1 - 34 / 34

Tableau de bord réalisé avec Looker Studio

---

## **Cadrage/Concept**

### **Obtention du besoin**

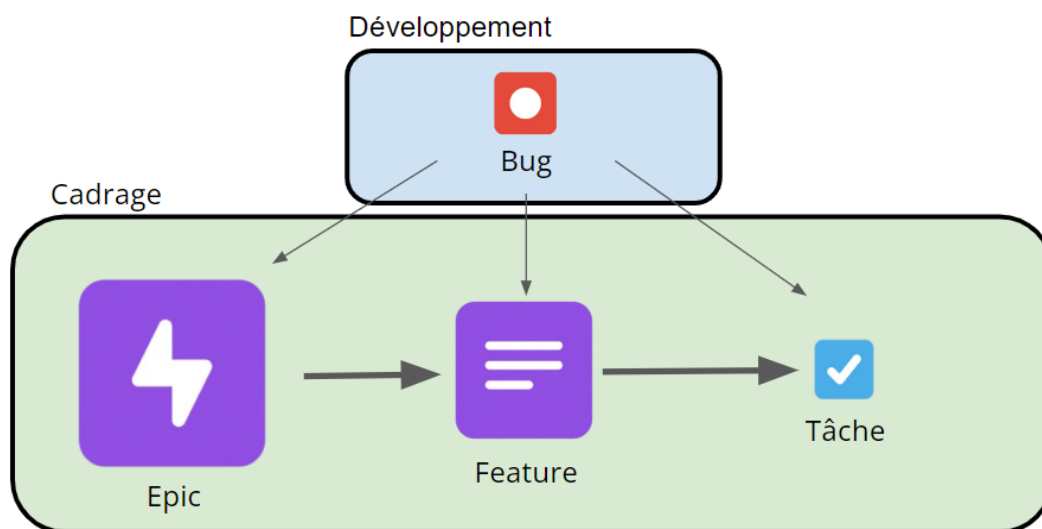
La phase de conception a débuté par des réunions avec notre "client final", Mme. Emmanuelle GRISLIN, l'enseignante encadrante de ce projet individuel. Lors de ces réunions, nous avons discuté des grandes lignes et des objectifs du projet. Ces échanges nous ont permis d'obtenir des détails supplémentaires sur les besoins du projet ainsi que les attentes qui y sont liées.



---

## Brainstorming

À partir des besoins identifiés, nous avons transcrit sur Jira des tickets correspondants aux systèmes que nous souhaitons mettre en place. Nous avons créé un premier "epic", qui représente un élément majeur dans son ensemble, dans notre cas, il s'agit du modèle NetLogo. Cet epic est composé de tickets "features" qui représentent les fonctionnalités à développer. Ces features peuvent elles-mêmes comporter des tickets "tâches", qui sont les composantes spécifiques des fonctionnalités (par exemple, le pathfinding pour l'agent utilisateur). Nous avons donc créé des tickets pour chaque idée qui nous est venue à l'esprit, puis nous les avons détaillés davantage une fois que nous avons épuisé toutes les idées disponibles. Des tickets "bugs" pourront être ajoutés ultérieurement dans la phase de développement si des problèmes complexes sont identifiés, ces tickets peuvent être associés à n'importe quel autre ticket qu'il soit epic feature ou tâche.



## Développement des idées

Après avoir créé les tickets et les avoir mis en statut "À cadrer", nous entamons la dernière phase du cadrage. Nous avons développé chaque ticket en fonction de la fonctionnalité que nous souhaitions mettre en place. Par exemple, pour le ticket "Capteurs (Agents)", nous avons décrit l'ensemble des types de capteurs à mettre en place:

### Capteurs (Agents)

 Joindre

 Ajouter un ticket enfant

 Associer un ticket



Description

### BREEDS CAPTEURS

Les capteurs renverront des données qui seront traitées par l'export de données

### Génériques

Les capteurs génériques peuvent être affectés sur plusieurs types de turtles ou de patches différents.

Nom capteur	Peut être lié à	Donnée produite
CapteurTempérature	Pièce ou ensemble de pièces	Température ambiante en °C
CapteurPorte	Porte	Ouverture
CapteurMouvement	Pièce	Mouvement vivant détecté
CapteurAllumage	Objet connecté (Turtle)	Activation de l'objet Etat de l'objet
CapteurFumée	Pièce ou rayon ou ensemble de	Présence de fumée

---

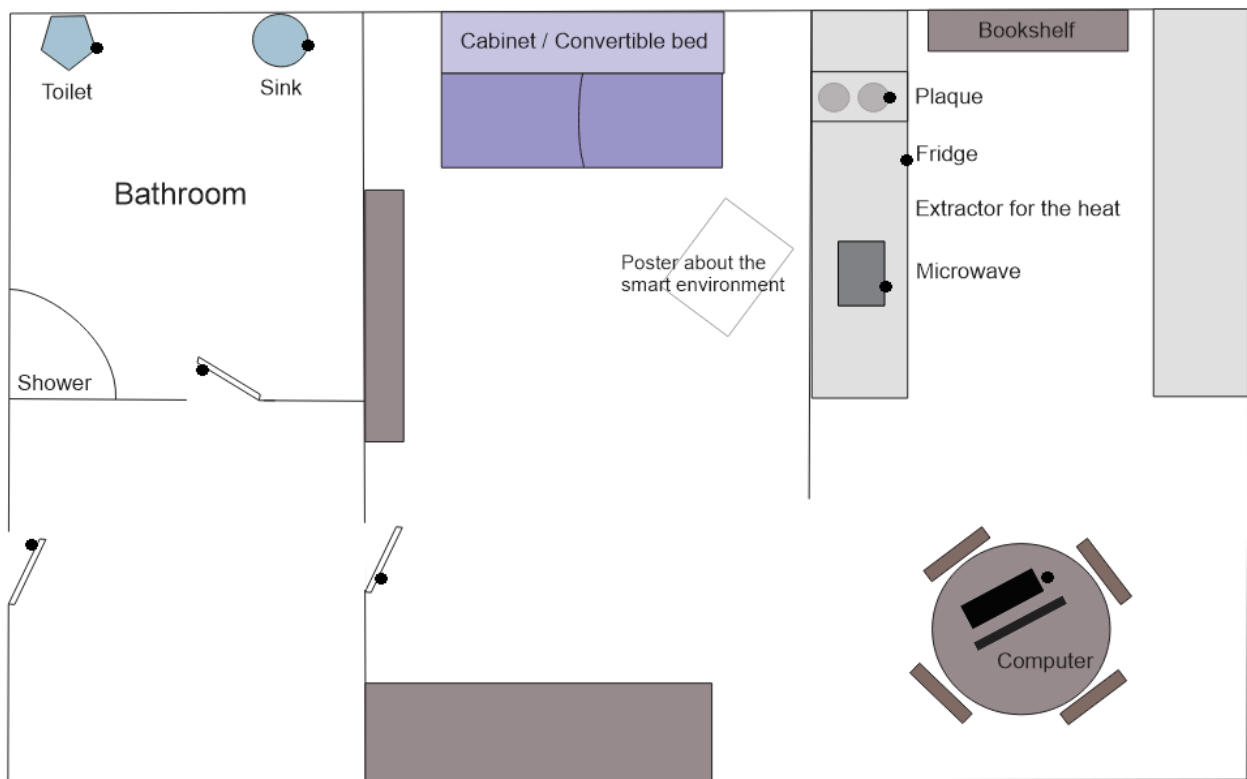
## Déroulé du développement

Une fois la grande plupart des tickets complétés et mis en statut "Cadrage terminé", nous pouvons commencer les développements sur Netlogo. Dans cette partie nous décrivons le déroulement du développement des différentes facettes du projet.

### Environnement

#### Appartement

Il nous a d'abord fallu mettre au point l'environnement du modèle, le "monde" en quelque sorte. Dans un premier lieu on a repris le plan de l'appartement fourni par l'enseignante encadrante:



---

Et nous avons donc retranscrit ce plan pour faire en sorte de respecter le code couleur suivant:

Noir: murs

Gris: portes

Marron: meubles

Vert: entrée

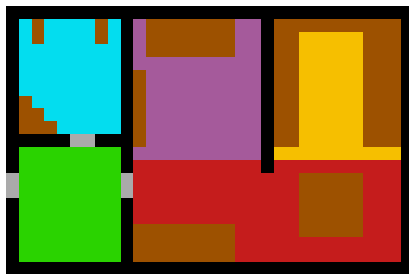
Violet: chambre

Rouge: salle à manger

Jaune: Cuisine

Ces couleurs vont servir à distinguer les pièces ainsi qu'à indiquer à la fonction de pathfinding où est-ce qu'elle peut établir un chemin (Pour ne pas traverser les murs par exemple).

On obtient donc cette image:



---

Entre temps l'image a évolué pour correspondre aux besoins du modèle, la taille a été ajustée, les meubles occupent désormais généralement qu'un seul patch et des [fenêtres](#) ont été ajoutés.

Voici la version finale utilisée par le modèle (appart-petit.png):



Cette image peut être chargée avec NetLogo via la primitive `import-pcolors`. Elle définit les couleurs des patches du modèle. Le monde du modèle doit avoir les mêmes dimensions que l'image pour ne pas avoir de problèmes de couleurs différentes.

---

## Température

Initialement, le plan était de créer un système de température extérieure qui évoluait selon une fonction mathématique sinusoïdale. Cependant, cette idée a été abandonnée au profit d'une approche plus itérative et algorithmique. La mise en place du système de température extérieure s'est déroulée sans incident. Cependant, lors du développement de la température intérieure, il a été nécessaire de revoir certains éléments, notamment en ce qui concerne l'échange de température entre les différentes pièces.

## Luminosité

La mise en place du système de luminosité extérieure repose sur l'utilisation d'un fichier éphéméride.csv. À l'origine, l'idée était d'utiliser une API pour calculer l'éphéméride en se basant sur la longitude et la latitude, mais il a été découvert trop tard que NetLogo ne permet pas d'appeler une API. (cf. [Problèmes rencontrés](#)). Autrement il n'y a pas eu d'écart sur ce qu'il a été cadré initialement.

## Mobilier

Dans la première version du cadrage il était prévu de faire en sorte que les meubles occupent plusieurs patches, mais cette idée a été abandonnée en faveur de l'approche "un agent par meuble". Par conséquent, plusieurs meubles tels que le lit, la douche, les toilettes, etc. ont été réduits à un seul patch, contrairement à l'image initiale. Nous avons défini par la suite l'ensemble des breeds des meubles avec leur attributs. Par la suite les meubles ont été placés sur leur patch respectif via des primitives sprout. Des fenêtres ont été ajoutées à posteriori afin de faire source de lumière naturelle dans l'appartement.

---

## Capteurs

Initialement, le plan était de créer à la fois des capteurs génériques (déclenchement, température) et des capteurs spécifiques d'export de données (un pour chaque objet connecté). Cependant, la conception a évolué pour ne laisser place qu'aux capteurs génériques avec les capteurs spécifiques qui ont été remplacés par un breed de capteur d'export de données (dataSensor) qui aura un comportement différent en fonction de l'objet auquel il est lié avec un link sensorLink.

## Roomba et système de propreté des pièces

Le Roomba a été implémenté tardivement dans le développement. Son rôle est de préparer l'implémentation de l'utilisateur et de son pathfinding. La pathfinding qui sert à retourner à sa station si son niveau de batterie est bas. Il sert également de générer des données relatives à la propreté des pièces.

## Utilisateur

L'implémentation de l'utilisateur et de son comportement a été naturellement la partie la plus complexe du développement, étant donné qu'il est la pièce maîtresse de ce projet. L'ensemble de son comportement a été soigneusement planifié lors du cadrage, il n'a donc pas connu d'écarts majeurs par rapport à ce qui était prévu, la plupart des ajustements étant soit mineurs soit des ajouts complémentaires. L'utilisateur est doté d'un système de pathfinding pour se déplacer d'un point à un autre (le même que le Roomba), d'un système de tâches avec priorités et de besoins à satisfaire. De plus, il peut transporter des objets grâce à un lien NetLogo nommé "carryLink".

---

## Export des données

La partie d'export des données a pu être rapidement mise en place, étant donné que nous avons opté pour une structure de colonnes simple :

Temps	Nom du capteur	Message émis du capteur
-------	----------------	-------------------------

Cependant, nous avons constaté que trop de données étaient exportées, atteignant entre 50 et 60 mégaoctets par jour simulé. Bien que le système ait été conçu pour éliminer les doublons et exporter uniquement les nouvelles données, il était évident que des ajustements étaient nécessaires. Après une optimisation minutieuse, qui comprenait la mise en place d'un arrondissement configurable pour les nombres exportés, nous avons réussi à réduire considérablement la taille moyenne des fichiers exportés à environ 7 mégaoctets.

## Visualisation

Dans le but de démontrer l'exploitabilité des données et de nous familiariser avec un outil de visualisation de données, nous avons créé une première version d'un tableau de bord présentant les données émises par le projet. Bien que nous ayons rencontré quelques difficultés liées à l'outil lui-même, nous avons obtenu un résultat satisfaisant qui démontre que le projet atteint ses objectifs. Ce tableau de bord permet de mettre en évidence les informations pertinentes et offre une vision claire de l'ensemble des données exportées.



---

## Description du modèle du livrable

Cette section présentera le fonctionnement final du modèle dans les moindres détails. Elle couvrira en profondeur les différentes fonctionnalités établies d'un point de vue technique. Elle comprend une explication détaillée du mode de fonctionnement du modèle, en mettant en évidence les mécanismes clés qui ont été mis en place. Nous aborderons ensuite les spécificités techniques qui permettent d'assurer une performance optimale et une utilisation efficace du modèle. Il est important de noter que 1 tick est égal à une seconde dans la simulation.

---

## Initialisation

La préparation de la simulation est gérée par la procédure "setup":

```
; FONCTION SETUP
to setup
  file-close-all
  clear-all
  reset-ticks
  stop-inspecting-dead-agents

  set toWrite ""
  set dataFilePath nobody
  let dataFilePathInput nobody
  if saveData[
    set dataFilePathInput user-new-file

    if dataFilePathInput != nobody and dataFilePathInput != false[
      set dataFilePath dataFilePathInput
      ifelse (position "." dataFilePath) != false [
        if (substring dataFilePath (position "." dataFilePath) (length dataFilePath)) != ".csv"[
          let dataFilePathWithoutExtension remove (word (substring dataFilePath (position "." dataFilePath) (length dataFilePath))) dataFilePath
          set dataFilePath (word dataFilePathWithoutExtension ".csv")
        ]
      ][
        set dataFilePath (word dataFilePath ".csv")
      ]
    ]
  ]
]
```

Cette procédure débute par le nettoyage de la simulation précédente en exécutant quatre commandes :

- file-close-all : Cette commande ferme tous les fichiers de sortie précédemment ouverts.
- clear-all : Cette commande supprime tous les agents (patches, turtles, etc.) et réinitialise l'environnement à un état vide.
- reset-ticks : Cette commande réinitialise le compteur de pas de temps à zéro.
- stop-inspecting-dead-agents : Cette commande ferme les fenêtres d'inspection des agents morts, soit toutes les fenêtres comme on utilisé clear-all

Ensuite, le code vérifie si l'export des données est activé via saveData qui est un paramètre activable sur l'interface. Si il est activé, l'utilisateur est invité à spécifier un chemin de fichier à l'aide de user-new-file qui ouvre un pop-up de choix de chemin. Ce choix sera le chemin du ou des fichiers de données. Si un chemin de fichier valide est sélectionné (c'est à dire que la fenêtre de choix de chemin n'a pas été fermée et qu'il n'y pas eu d'appui sur annuler), il est assigné à la variable dataFilePath. Une fois le chemin spécifié, le code vérifie si l'extension du fichier sélectionné est .csv (Si il a été spécifié dans le pop-up). Si ce n'est pas le cas, il modifie le chemin de fichier pour inclure l'extension .csv.

---

```

if (saveData and dataFilePathInput != nobody and dataFilePathInput != false) or not saveData[
  ;; Initialisation variables globales
  ;;; Date et saison actuelle
  setupDate

  ;;; Routine
  setupRoutine weekDay hour minute

  ;;; Température
  setupTemperature

  ;;; Gestion de la luminosité extérieure
  setupLightOutside

```

Si l'export des données est activé et que le chemin est valide (ou si l'export est désactivé), les premières procédures d'initialisation du modèle sont lancées: Celles de la [date](#), le chargement du fichier de [routine](#), la [température](#). Ces procédures sont expliquées plus loin.

```

;;; Fichier
if saveData[
  ;;; Insertion date dans nom de fichier
  if oneFilePerDay[
    set dataFilePath insert-item (position "." dataFilePath) dataFilePath time:show currentDateTime "yyyy_MM_dd"
  ]
  if file-exists? dataFilePath[
    file-delete dataFilePath
  ]
  file-open dataFilePath
  file-print fileHeader
]

```

Si l'export est activé, la date de début de la simulation est insérée dans le nom du fichier si l'export d'un fichier par jour est activé via la variable `oneFilePerDay`. Si le fichier d'export existe déjà suite à une ancienne, il est écrasé. Ensuite le fichier est ouvert via `file-open` et le header spécifié dans `fileHeader` est écrit dans le fichier en tant que première ligne.

---

```
;; Chargement des couleurs de patches
import-pcolors "appart_petit.png"

;; Création des meubles (manuel)
setupFurniture

;; Création des objets
setupObjects

;; Création des capteurs
setupSensors

;; Création de l'utilisateur à la porte d'entrée
setupUser

;; Initialisation variable Luminosité intérieur
setupLightInside
]
end
; FIN SETUP
```

L'image source "appart\_petit.png" est ensuite utilisée pour charger les couleurs des patches.

Enfin, les procédures d'initialisation des [meubles](#), des [objets](#), des [capteurs](#), de [l'utilisateur](#) et de la [luminosité](#) sont lancées.

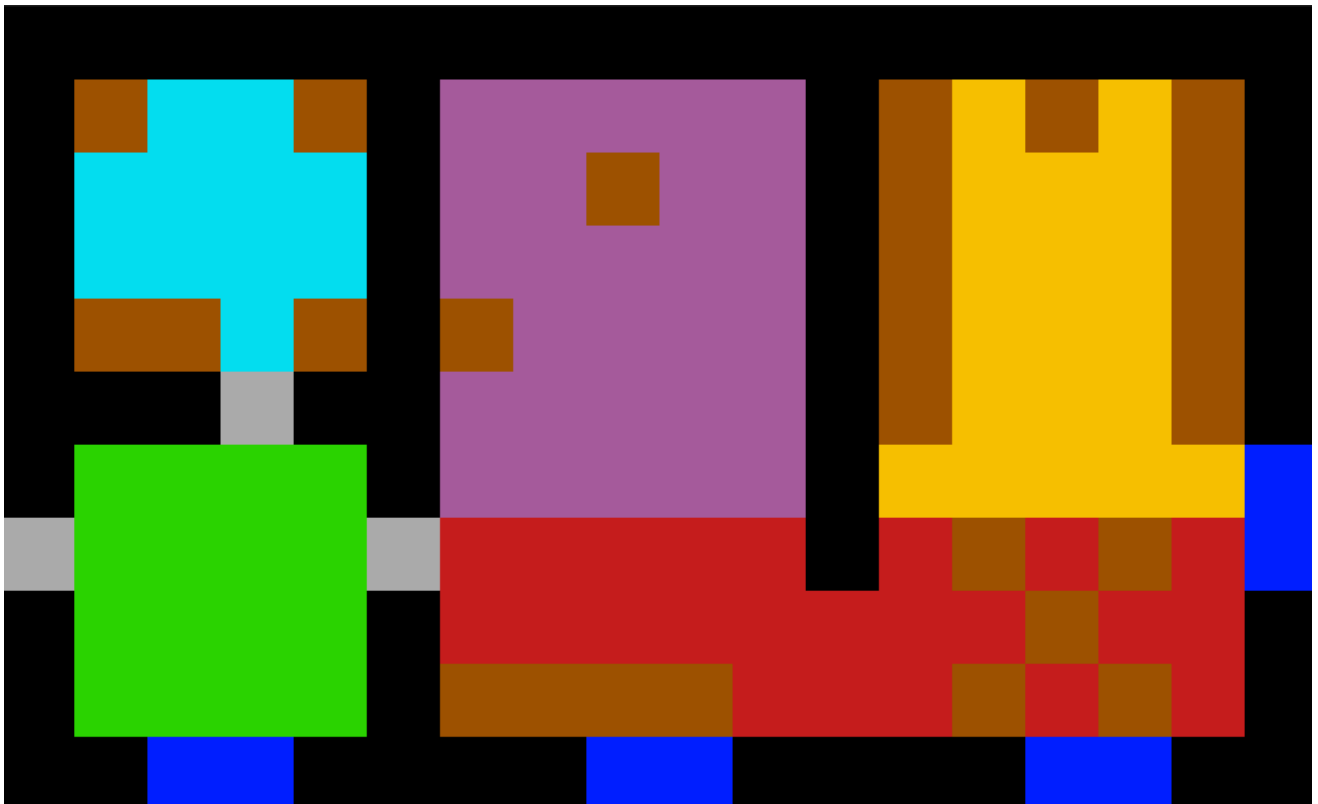
---

## Monde

L'instruction "import-pcolors" dans la fonction "setup" précédemment décrite est utilisée pour importer les couleurs de l'image "appart\_petit.png" dans l'environnement de simulation NetLogo.

```
;; Chargement des couleurs de patches  
import-pcolors "appart_petit.png"
```

L'image "appart\_petit.png" a été préalablement préparée pour représenter les différentes couleurs des patches dans la simulation (cf. [partie précédente](#)). Chaque pixel de l'image correspondra à un patch dans l'environnement NetLogo, et sa couleur sera utilisée pour définir la couleur du patch correspondant. L'ensemble du fonctionnement du modèle est basé sur ces couleurs de patches.



---

## Exécution d'un tick

```
⊞ ; Fonction GO
  to go
    ;; Gestion nouvelle journée
    if hour = 0 and minute = 0 and second = 0 [
      newDay
    ]
```

La gestion d'un tick est faite par la procédure go. La procédure commence par vérifier si l'heure, les minutes et les secondes sont tous égaux à zéro. Si c'est le cas, cela signifie qu'une nouvelle journée commence, donc la fonction newDay est appelée pour effectuer les tâches nécessaires au début d'une nouvelle journée:

```
;; Passage jour suivant
to newDay
  if savedata[
    file-close-all
  ]
  ;;; Définition routine du jour
  readRoutine weekday

  ;;; Définition éphéméride du jour
  readEphemeride month day

  ;;; Définition saison en fonction de la date
  if ((month = 12 and day >= 21) or month = 1 or month = 2 or (month = 3 and day < 20)) and season != "Winter"[
    set season "Winter"
  ]
  if ((month = 3 and day >= 21) or month = 4 or month = 5 or (month = 6 and day < 20)) and season != "Spring"[
    set season "Spring"
  ]
  if ((month = 6 and day >= 21) or month = 7 or month = 8 or (month = 9 and day < 20)) and season != "Summer"[
    set season "Summer"
  ]
  if ((month = 9 and day >= 21) or month = 10 or month = 11 or (month = 12 and day < 20)) and season != "Fall"[
    set season "Fall"
  ]
]
```

---

La procédure commence par vérifier si l'option `saveData` est activée. Si c'est le cas, tous les fichiers ouverts sont fermés à l'aide de la commande `file-close-all`. Cela permet de fermer les fichiers qui ont été utilisés pour enregistrer les données pendant la journée précédente. Cela sert de même à laisser la place aux prochaines procédures pour ouvrir leur fichiers. La [routine](#) du jour est lue à partir d'une source externe en utilisant la fonction `readRoutine`. La variable `weekday` est passée en argument pour déterminer quelle routine doit être chargée pour le jour en cours. [L'éphéméride](#) du jour est lue à partir d'une source externe en utilisant la fonction `readEphemeride`. Les variables `month` (mois) et `day` (jour) sont passées en argument pour déterminer quelle éphéméride doit être chargée pour la date en cours. L'éphéméride contient les informations sur les heures de lever et de coucher du soleil.

Ensuite, la saison est déterminée en fonction de la date. Différentes conditions sont vérifiées en utilisant les valeurs des variables `month` et `day`, et la variable `season` est mise à jour en conséquence. Par exemple, si le mois est décembre et le jour est le 21 ou après, ou si le mois est janvier ou février, ou si le mois est mars et le jour est avant le 20, alors la saison est définie comme "Winter". Des conditions similaires sont vérifiées pour les autres saisons et la variable `season` est mise à jour en conséquence.

```
;;; Gestion anniversaires
ask Users[
  if month = birthMonth and day = birthDay[
    set age age + 1
  ]
]
```

La gestion des anniversaires est effectuée par la suite. Pour chaque agent de type "User", les variables `month` et `day` de l'agent sont comparées avec la date actuelle (`month` et `day`). Si elles correspondent, l'âge de l'agent est augmenté de 1.

---

```
if savedata[
  if oneFilePerDay[
    let yesterday time:plus currentDateTime -1 "days"
    set dataFilePath remove (time:show yesterday "yyyy_MM_dd") dataFilePath
    set dataFilePath insert-item (position "." dataFilePath) dataFilePath time:show currentDateTime "yyyy_MM_dd"
  ]
  file-open dataFilePath
  if oneFilePerDay[
    file-print fileHeader
  ]
]
end
```

Si l'option `saveData` est activée, des actions supplémentaires sont effectuées pour la gestion de l'export des données. Si l'option `oneFilePerDay` est activée, le nom du fichier de données est modifié pour inclure la date actuelle. Le fichier est ensuite ouvert (ou créé) en utilisant la commande `file-open`, et si l'option `oneFilePerDay` est activée, l'en-tête du fichier est écrit en utilisant la commande `file-print` car il a créé un nouveau fichier de données.



---

```
; Fonction GO
to go
;; Gestion nouvelle journée
if hour = 0 and minute = 0 and second = 0 [
  newDay
]

;; Appel gestion de la lumière
lightManagement

;; Appel Gestion de la température extérieure
temperatureOutsideManagement

;; Appel gestion du thermostat
thermostatManagement

;; Appel gestion de la température intérieure
insideTemperatureManagement

;; Gestion de la saleté
dirtManagement

;; Appel comportement Roomba
roombaBehaviour

;; Appel comportement station Roomba
roombaStationBehaviour

;; Comportement Utilisateur
userBehaviour

;; Comportement Objets
objectsBehaviour

;; Comportement Capteurs
sensorBehaviour

ifelse saveData[
  ;; Ecriture des données
  writeData
][
  set toWrite ""
]

;;1 tick = 1 seconde
incrementOneSecond
tick
```

---

---

En retournant à la procédure `go`, après avoir géré la transition vers une nouvelle journée elle va exécuter les différentes fonctions de gestion des agents de la simulation :

- [`lightManagement`](#) : Gère l'éclairage, notamment en ajustant la luminosité intérieure et extérieure en fonction de l'heure et de l'état des volets et des lumières.
- [`temperatureOutsideManagement`](#) : Gère la température extérieure en fonction des pics de température et de la saison.
- `thermostatManagement` : Gère le thermostat en gérant les agents capables de modifier la température, soit les chauffages et les climatiseurs.
- [`insideTemperatureManagement`](#) : Gère la température intérieure, en prenant en compte le chauffage, la climatisation et les sources de chaleur internes.
- [`dirtManagement`](#) : Gère la saleté dans l'environnement, en incrémentant la variable de saleté de chaque pièce.
- [`roombaBehaviour`](#) : Gère le comportement des robots aspirateurs Roomba, tel que le déplacement, le nettoyage, etc...
- [`roombaStationBehaviour`](#) : Gère le comportement de la station de recharge pour Roombas.
- [`userBehaviour`](#) : Gère le comportement de l'utilisateur, qui peut interagir avec le modèle ou prendre des décisions.
- [`objectsBehaviour`](#) : Gère le comportement des autres objets ou agents présents dans le modèle.
- [`sensorBehaviour`](#) : Gère le comportement des capteurs, qui peuvent collecter des données de l'environnement.

Ensuite, le modèle vérifie si l'option `saveData` est activée. Si c'est le cas, la fonction [`writeData`](#) est appelée pour écrire les données des capteurs dans le fichier.

---

```
; FONCTIONS POUR GO
;; Incrémentation currentDateTime de 1 seconde
to incrementOneSecond
  set currentDateTime time:plus currentDateTime 1 "second"
  set year time:get "year" currentDateTime
  set month time:get "month" currentDateTime
  set day time:get "day" currentDateTime
  set weekday time:get "dayofweek" currentDateTime

  set hour time:get "hour" currentDateTime
  set minute time:get "minute" currentDateTime
  set second time:get "second" currentDateTime
end
```

Enfin, la procédure incrementOneSecond est lancée avant le tick pour faire avancer l'heure d'une seconde dans la simulation et actualiser les variables globales associées.

---

## Variables globales

Ce modèle comporte différentes variables globales qui sont gérées de différentes manières.

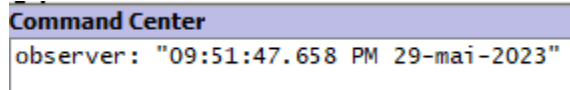
### Date

```
;; Fonction de calcul date actuelle et saison
to setupDate

  ;;; Récupération date
  let datetimeString date-and-time
  ;;; Conversion mois
  if substring datetimeString 19 22 = "jan"[
    set month "01"
  ]
  if substring datetimeString 19 22 = "fev" or substring datetimeString 19 22 = "feb"[
    set month "02"
  ]
  if substring datetimeString 19 22 = "mar"[
    set month "03"
  ]
  if substring datetimeString 19 22 = "avr" or substring datetimeString 19 22 = "apr"[
    set month "04"
  ]
  if substring datetimeString 19 22 = "mai" or substring datetimeString 19 22 = "may"[
    set month "05"
  ]
  if substring datetimeString 19 22 = "jun"[
    set month "06"
  ]
  if substring datetimeString 19 22 = "jui" or substring datetimeString 19 22 = "jul"[
    set month "07"
  ]
  if substring datetimeString 19 22 = "aou" or substring datetimeString 19 22 = "aug"[
    set month "08"
  ]
  if substring datetimeString 19 22 = "sep"[
    set month "09"
  ]
  if substring datetimeString 19 22 = "oct"[
    set month "10"
  ]
  if substring datetimeString 19 22 = "nov"[
    set month "11"
  ]
  if substring datetimeString 19 22 = "dec"[
    set month "12"
  ]
]
```

---

La procédure `setUpDate` est utilisée pour initialiser la date de la simulation. Elle commence par récupérer la date et l'heure actuelles à l'aide de la primitive `date-and-time`. Le résultat est stocké dans une variable `datetimeString`. Voici un exemple de résultat de la primitive `date-and-time` sur le PC utilisé pour développer ce projet :



The screenshot shows a window titled "Command Center" with a single line of text: `observer: "09:51:47.658 PM 29-mai-2023"`. The text is displayed in a monospaced font, with the date and time in French.

On remarque que le mois est exprimé en français, dans l'optique de garantir la portabilité du projet il faudra prendre en compte aussi les versions en anglais.

La suite de la fonction extrait le mois de `datetimeString` en vérifiant différentes conditions pour chaque mois. Par exemple, si les caractères 19 à 22 de `datetimeString` correspondent à "jan", alors le mois est défini sur "01". Cette logique est répétée pour tous les mois de l'année.

```
;;; Création datetime de l'extension date à partir de datetimeString
let datetimeString2 (word (substring datetimeString 23 27) "-" month "-" (substring datetimeString 16 18) " " (substring datetimeString 11 15))
if startAtmorning[
  set datetimeString2 (word (substring datetimeString 23 27) "-" month "-" (substring datetimeString 16 18) " 06:00:00")
]
let tmpDatetime time:create datetimeString2
```

En utilisant les informations extraites du mois et de `datetimeString`, la fonction construit une chaîne de texte `datetimeString2` qui représente la date avec un format spécifique ("YYYY-MM-DD HH:MM:SS"). Si l'option `startAtmorning` est activée sur l'interface, l'heure est fixée à six heures du matin. La chaîne `datetimeString2` est ensuite convertie en un objet `datetime` temporaire en utilisant la primitive `time:create`.

---

```

;;; Affectation variables globales
set year time:get "year" tmpDatetime
set month time:get "month" tmpDatetime
set day time:get "day" tmpDatetime
set weekday time:get "dayofweek" tmpDatetime

;;; Conversion 12h vers 24
if substring datetimeString 13 15 = "PM" and (substring datetimeString 0 2 != "12") and not startAtmorning[
  set tmpDatetime time:plus tmpDatetime 12 "hours"
]
if substring datetimeString 13 15 = "AM" and (substring datetimeString 0 2 = "12") and not startAtmorning[
  set tmpDatetime time:plus tmpDatetime -12 "hours"
]

```

Les variables globales year, month, day, weekday, hour, minute, second et currentDateTime sont définies en extrayant les composants de la date et de l'heure de tmpDatetime. currentDateTime sera l'objet datetime référence dans la simulation. Une conversion de AM/PM vers format 24h est réalisée ensuite.

```

;;; Calcul Saison
if (month = 12 and day >= 21) or month = 1 or month = 2 or (month = 3 and day < 20)[
  set season "Winter"
]
if (month = 3 and day >= 21) or month = 4 or month = 5 or (month = 6 and day < 20)[
  set season "Spring"
]
if (month = 6 and day >= 21) or month = 7 or month = 8 or (month = 9 and day < 20)[
  set season "Summer"
]
if (month = 9 and day >= 21) or month = 10 or month = 11 or (month = 12 and day < 20)[
  set season "Fall"
]
end

```

Enfin, la fonction détermine la saison en vérifiant les conditions sur le mois et le jour. Par exemple, si le mois de décembre et le jour est supérieur ou égal à 21, ou si le mois est janvier ou février, alors la saison est définie sur Winter. Ce processus est répété pour toutes les saisons de l'année en fonction de la date.

---

## Température

### Initialisation

```
;; Temperature
to setupTemperature
  ;; Choix de la température de la journée en fonction de la saison
  let temperatureVariation random maxTemperatureVariation
  if season = "Winter" [
    set targetTemperatureOutsideMin minTemperatureWinter + temperatureVariation
    set targetTemperatureOutsideMax maxTemperatureWinter - temperatureVariation
  ]
  if season = "Spring" [
    set targetTemperatureOutsideMin minTemperatureSpring + temperatureVariation
    set targetTemperatureOutsideMax maxTemperatureSpring - temperatureVariation
  ]
  if season = "Summer" [
    set targetTemperatureOutsideMin minTemperatureSummer + temperatureVariation
    set targetTemperatureOutsideMax maxTemperatureSummer - temperatureVariation
  ]
  if season = "Fall" [
    set targetTemperatureOutsideMin minTemperatureFall + temperatureVariation
    set targetTemperatureOutsideMax maxTemperatureFall - temperatureVariation
  ]

  ;; Calcul du temps restant avant le prochain extremum
  ;; Calcul des datetimes extremums
  ifelse hour < maxTemperatureHour
  [
    set dateTimeTemperatureMin time:create (word year "-" month "-" day " " minTemperatureHour)
  ]
  [
    ;; Si setup lancé après heure max, mettre le pic au day suivant
    set dateTimeTemperatureMin time:create (word (time:get "year" currentDateTime) "-" (time:get "month" currentDateTime)
    ]
  ]
  set dateTimeTemperatureMax time:create (word year "-" month "-" day " " maxTemperatureHour)
```

La procédure “setupTemperature” est utilisée pour configurer la température initiale et calculer le temps restant avant le prochain extrême de température. Elle est appelée lors de l’initialisation du modèle avec setup. Tout d’abord, en fonction de la saison spécifiée, les températures minimales et maximales de la journée sont déterminées et ajustées avec une variation aléatoire paramétrable. Ensuite, en fonction de l’heure actuelle de la simulation, les objets datetimes des pics de température (minimum et maximum) sont calculés.

---

```

;;; Calcul du temps restant en secondes
let secondsBeforeExtremum 0
;;; Si après heure de la température max, utiliser l'heure de la température min du jour suivant (déjà calculé)
ifelse time:is-after? currentDateTime dateTimeTemperatureMax [
  set secondsBeforeExtremum time:difference-between currentDateTime dateTimeTemperatureMin "seconds"
]
[
  ;;;; Si avant heure de la température min, utiliser l'heure de la température min
  ifelse time:is-before? currentDateTime dateTimeTemperatureMin [
    set secondsBeforeExtremum time:difference-between currentDateTime dateTimeTemperatureMin "seconds"
  ]
  ;;;; Si entre 2 extremums, utiliser l'heure de la température max
  set secondsBeforeExtremum time:difference-between currentDateTime dateTimeTemperatureMax "seconds"
]
]

;;; Affectation des températures en fonction de la température cible
ifelse time:is-before? dateTimeTemperatureMin dateTimeTemperatureMax [
  ;;;; Si setup lancé avant pic max
  set secondsBetweenExtremums time:difference-between dateTimeTemperatureMin dateTimeTemperatureMax "seconds"
  set temperatureOutside targetTemperatureOutsideMin + (secondsBetweenExtremums - secondsBeforeExtremum) * ((targetTempera
]
[
  ;;;; Si setup lancé après pic max
  set secondsBetweenExtremums time:difference-between dateTimeTemperatureMax dateTimeTemperatureMin "seconds"
  set temperatureOutside targetTemperatureOutsideMax - (secondsBetweenExtremums - secondsBeforeExtremum) * ((targetTempera
]
]
set temperaturePrincipalRooms temperatureOutside
set temperatureEntrance temperatureOutside
set temperatureBathroom temperatureOutside
end

```

---

Ensuite, la différence en secondes entre l'heure actuelle et les pics de température est calculée. Cette différence est utilisée pour ajuster la température extérieure en fonction du temps écoulé depuis le dernier pic. Si l'heure actuelle est avant l'extrême de température minimum, la température extérieure est calculée en utilisant une interpolation linéaire entre les températures minimales et maximales. Si l'heure actuelle est après l'extrême de température maximum, la température extérieure est calculée en utilisant une interpolation linéaire inversée. Enfin, la température extérieure est attribuée aux différentes pièces (temperaturePrincipalRooms, temperatureEntrance, temperatureBathroom) dans la simulation.



---

## Extérieur simulation

```
; Gestion de la température extérieure
to temperatureOutsideManagement
;; Si heure du pic atteint, alors temperatureOutside = temperature du pic et on met le datetime du prochain pic au lendemain
if time:is-equal? currentDateTime dateTimeTemperatureMax[
  set temperatureOutside targetTemperatureOutsideMax
  set dateTimeTemperatureMin time:plus dateTimeTemperatureMin 1 "day"
  ;; Changement température cible
  let temperatureVariation random maxTemperatureVariation
  if season = "Winter" [
    set targetTemperatureOutsideMin minTemperatureWinter + temperatureVariation
  ]
  if season = "Spring" [
    set targetTemperatureOutsideMin minTemperatureSpring + temperatureVariation
  ]
  if season = "Summer" [
    set targetTemperatureOutsideMin minTemperatureSummer + temperatureVariation
  ]
  if season = "Fall" [
    set targetTemperatureOutsideMin minTemperatureFall + temperatureVariation
  ]
  set secondsBetweenExtremums time:difference-between dateTimeTemperatureMax dateTimeTemperatureMin "seconds"
]

if time:is-equal? currentDateTime dateTimeTemperatureMin[
  set temperatureOutside targetTemperatureOutsideMin
  set dateTimeTemperatureMax time:plus dateTimeTemperatureMax 1 "day"

  let temperatureVariation random maxTemperatureVariation
  if season = "Winter" [
    set targetTemperatureOutsideMax maxTemperatureWinter - temperatureVariation
  ]
  if season = "Spring" [
    set targetTemperatureOutsideMax maxTemperatureSpring - temperatureVariation
  ]
  if season = "Summer" [
    set targetTemperatureOutsideMax maxTemperatureSummer - temperatureVariation
  ]
  if season = "Fall" [
    set targetTemperatureOutsideMax maxTemperatureFall - temperatureVariation
  ]
  set secondsBetweenExtremums time:difference-between dateTimeTemperatureMin dateTimeTemperatureMax "seconds"
]
```

La procédure "temperatureOutsideManagement" est utilisée à chaque tick pour gérer le changement de température extérieure en fonction de l'heure de la journée. Si l'heure actuelle correspond à l'heure de l'extrême de température maximum, la température extérieure est réglée sur la température maximale et le datetime de l'extrême de température minimum est mis à jour pour le lendemain. De plus, la température cible est ajustée avec une variation aléatoire en fonction de la saison. Si l'heure actuelle correspond à l'heure de l'extrême de température minimum, la température extérieure est réglée sur la température minimale et le datetime du pic de température maximum est mis à jour pour le lendemain. De plus, la température cible est ajustée en fonction de la saison avec une variation aléatoire.

```

;; Si après heure de la température max ou avant heure de la température min, diminuer la température
ifelse time:is-after? currentDateTime dateTimeTemperatureMax or time:is-before? currentDateTime dateTimeTemperatureMin [
  set temperatureOutside temperatureOutside - ((targetTemperatureOutsideMax - targetTemperatureOutsideMin) / secondsBetweenExtremums)
][ ; ; Sinon (entre 2 extremums), monter la température
  set temperatureOutside temperatureOutside + ((targetTemperatureOutsideMax - targetTemperatureOutsideMin) / secondsBetweenExtremums)
]
end

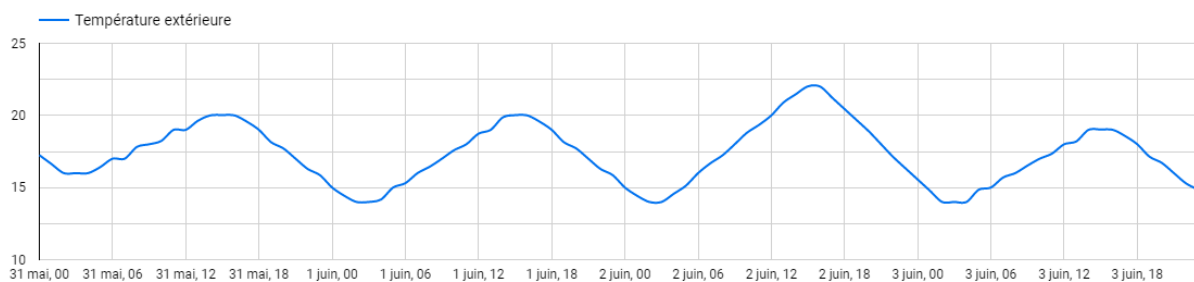
```

Si l'heure actuelle est après l'extrême de température maximum ou avant l'extrême de température minimum, la température extérieure diminue progressivement. Sinon, la température extérieure augmente progressivement. Cela permet de simuler les variations de température au fil du temps.

On donc doit retrouver la température sous forme d'une pseudo-sinusoïdale:



Sélectionner la période



---

## Intérieur simulation

```
; Gestion de la température intérieure
to insideTemperatureManagement
;; Echange passif avec extérieur
;; Si au moins 1 fenêtre ouverte dans les pièces principales, isolation avec l'extérieur divisé par 100
ifelse any?(windows with [isOpen = true and (any?(neighbors with [pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3]) or any?(neighbors with [pcolor = 23.3 and any?(n
[
  set temperaturePrincipalRooms temperaturePrincipalRooms - ((temperaturePrincipalRooms - temperatureOutside) / (isolation / 100))
][
  set temperaturePrincipalRooms temperaturePrincipalRooms - ((temperaturePrincipalRooms - temperatureOutside) / isolation)
]
;; Si au moins 1 fenêtre ouverte dans l'entrée
ifelse any?(windows with [isOpen = true and (any?(neighbors with [pcolor = 64.7]) or any?(neighbors with [pcolor = 23.3 and any?(neighbors with [pcolor = 64.7]))]))
[
  set temperatureEntrance temperatureEntrance - ((temperatureEntrance - temperatureOutside) / (isolation / 100))
][
  set temperatureEntrance temperatureEntrance - ((temperatureEntrance - temperatureOutside) / isolation)
]
;; Gestion température SdB
set temperatureBathroom temperatureBathroom - ((temperatureBathroom - temperatureOutside) / isolation)
```

La procédure `insideTemperatureManagement` est responsable de la gestion de la température intérieure, elle est exécutée à chaque tick. La première partie du code concerne les échanges passifs de température avec l'extérieur. Si au moins une fenêtre est ouverte dans les pièces principales, l'isolation est divisée par 100. La température des pièces principales est ajustée en fonction de la différence entre la température intérieure et la température extérieure, divisée par l'isolation. La même logique s'applique à l'entrée et à la salle de bain.

---

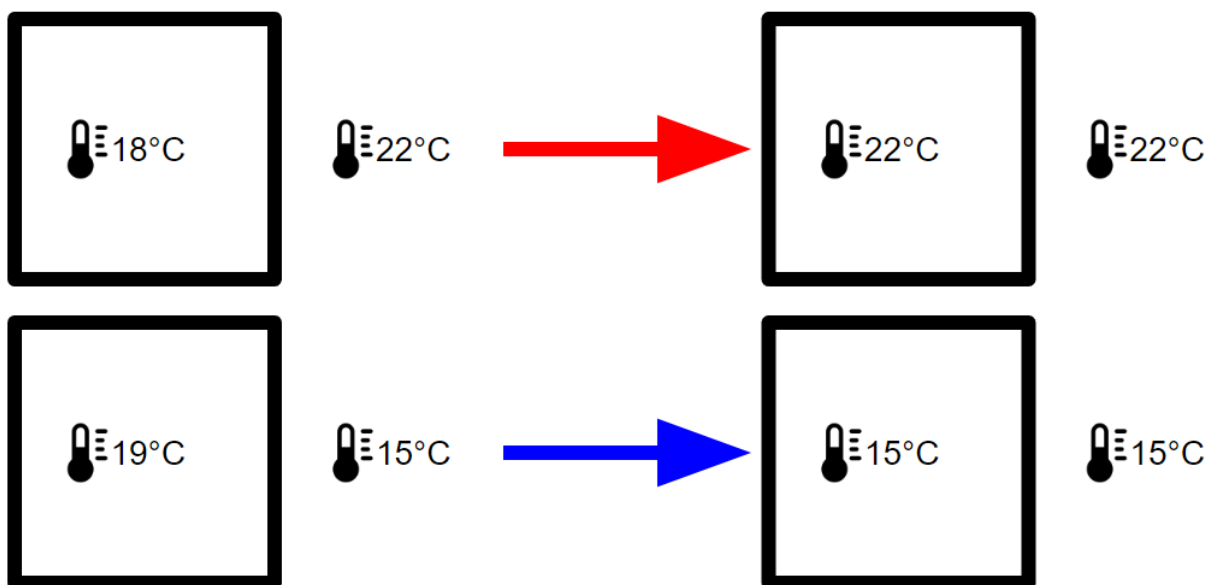
```

;; Equilibre température entre les pièces
let avgTemperature 0
;;; Entrée-Salle de bain
set avgTemperature (temperatureEntrance + temperatureBathroom) / 2
set temperatureEntrance temperatureEntrance - ((temperatureEntrance - avgTemperature) / (isolation / 10))
set temperatureBathroom temperatureBathroom - ((temperatureBathroom - avgTemperature) / (isolation / 10))

;;; Entrée-PiècesPrincipales
set avgTemperature (temperatureEntrance + temperaturePrincipalRooms) / 2
set temperatureEntrance temperatureEntrance - ((temperatureEntrance - avgTemperature) / (isolation / 10))
set temperaturePrincipalRooms temperaturePrincipalRooms - ((temperaturePrincipalRooms - avgTemperature) / (isolation / 10))

```

Ensuite, l'équilibre de température entre les différentes pièces est pris en compte. La température moyenne entre l'entrée et la salle de bain est calculée, et les températures de ces deux pièces sont ajustées en fonction de la différence avec la température moyenne, divisée par l'isolation (divisée par 10 pour une réduction de l'effet d'isolation, étant donné que les murs intérieurs sont moins bien isolés). De même, la température moyenne entre l'entrée et les pièces principales est calculée, et les températures de ces deux zones sont ajustées en fonction de la différence avec la température moyenne, divisée par l'isolation (divisée par 10).



---

```

;; Chauffage
;; On assume qu'il faut 70W pour 1m3
ask heaters with [isActive = true][
  ;; Entrée
  if pcolor = 64.7[
    set temperatureEntrance temperatureEntrance + ( (power / (25 * 70)) / 3600 ) ;;;; On assume que la pièce fait 25m3 (9m², 2.8m de hauteur)
  ]
  ;; Pièces principales
  if pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3[
    set temperaturePrincipalRooms temperaturePrincipalRooms + ( (power / (84 * 70)) / 3600 ) ;;;; On assume que la pièce fait 84m3 (30m², 2.8m de hauteur)
  ]
  ;; Salle de bain
  if pcolor = 84.9[
    set temperatureBathroom temperatureBathroom + ( (power / (25 * 70)) / 3600 )
  ]
]

```

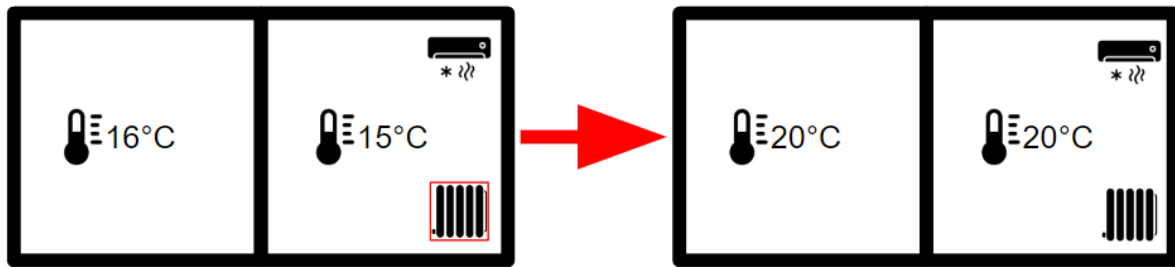
Juste après, la procédure gère dans un premier temps le chauffage. Si un radiateur est actif dans une pièce spécifique, la température de cette pièce est augmentée en fonction de la puissance du radiateur, en supposant qu'il faut 70W pour chauffer 1m<sup>3</sup> d'espace. Les calculs sont effectués en fonction du volume assumé de chaque pièce spécifique.



```

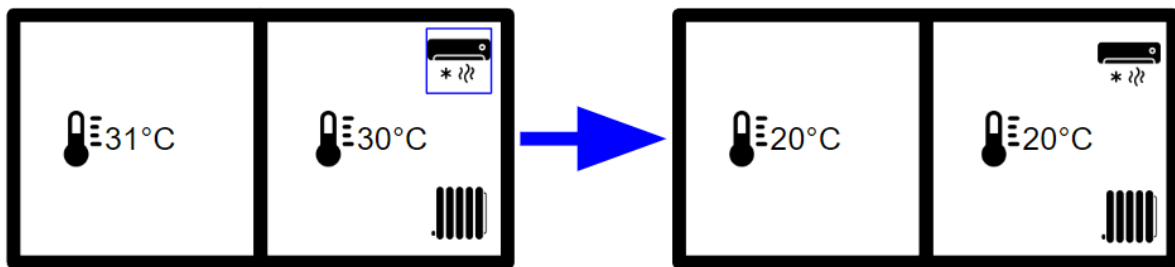
;; Refroidissement par clim
;; On assume qu'il faut 100W pour 1m3
ask ACs with [isActive = true][
  ;; Entrée
  if pcolor = 64.7[
    set temperatureEntrance temperatureEntrance - ( (power / (25 * 100)) / 3600 ) ;;;; On assume que la pièce fait 25m3
  ]
  ;; Pièces principales
  if pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3[
    set temperaturePrincipalRooms temperaturePrincipalRooms - ( (power / (84 * 100)) / 3600 ) ;;;; On assume que la pièce fait 84m3
  ]
  ;; Salle de bain
  if pcolor = 84.9[
    set temperatureBathroom temperatureBathroom - ( (power / (25 * 100)) / 3600 )
  ]
]
end



```

Enfin, la procédure gère le refroidissement par climatisation. De la même manière que les chauffages, si un climatiseur est actif dans une pièce spécifique, la température de cette pièce est réduite en fonction de la puissance du climatiseur, en supposant qu'il faut 100W pour refroidir 1m<sup>3</sup> d'espace. Les calculs sont effectués en fonction du volume de chaque pièce spécifique.



 20°C       15°C



 20°C       30°C

## Luminosité

### Extérieur initial

Afin de déterminer les heures de lever et de coucher du soleil, on utilise un fichier CSV d'éphéméride calculé depuis le site de la marine américaine : [https://aa.usno.navy.mil/data/RS\\_OneYear](https://aa.usno.navy.mil/data/RS_OneYear)

Pour le livrable nous avons utilisé la latitude et la longitude de Valenciennes (50.357761, 3.523180).

C:\Users\Alexis\Documents\NetlogoloT\config\ephemeride.csv - Notepad++

Fichier Édition Recherche Affichage Encodage Langage Paramètres Outils Macro Exécution Modules d'extension Documents ?

2Weeks2023\_06\_06.csv ephemeride.csv

```
1 DAY; Jan.SR; Jan.SS; Feb.SR; Feb.SS; Mar.SS; Mar.SR; Apr.SS; Apr.SR; MaySS; MaySR; JuneSS; JuneSR; JulySS; JulySR; Aug.SS; Aug.
2 01;0946;1753;0921;1839;0830;1927;0723;2018;0622;2105;0540;2148;0539;2200;0613;2130;0700;2031;0745;1925;0836;182
3 02;0946;1754;0919;1840;0828;1929;0721;2019;0620;2107;0540;2149;0540;2200;0615;2129;0701;2029;0747;1923;0838;182
4 03;0946;1755;0918;1842;0826;1930;0719;2021;0618;2108;0539;2150;0540;2200;0616;2127;0703;2027;0749;1921;0839;181
5 04;0946;1756;0916;1844;0824;1932;0717;2022;0617;2110;0538;2151;0541;2159;0618;2125;0704;2025;0750;1918;0841;181
6 05;0945;1757;0915;1845;0822;1934;0714;2024;0615;2111;0538;2152;0542;2159;0619;2124;0706;2023;0752;1916;0843;181
7 06;0945;1758;0913;1847;0820;1935;0712;2026;0613;2113;0537;2152;0543;2158;0621;2122;0707;2020;0753;1914;0844;181
8 07;0945;1800;0912;1849;0818;1937;0710;2027;0612;2114;0537;2153;0544;2158;0622;2120;0709;2018;0755;1912;0846;181
9 08;0944;1801;0910;1851;0816;1939;0708;2029;0610;2116;0536;2154;0544;2157;0624;2119;0710;2016;0756;1910;0848;181
10 09;0944;1802;0908;1852;0813;1940;0706;2030;0608;2117;0536;2155;0545;2156;0625;2117;0712;2014;0758;1908;0849;180
```

CSV Query

SELECT \* FROM THIS

DAY	JanSR	JanSS	FebSR	FebSS	MarSS	MarSR	AprSS	AprSR	MaySS	MaySR	JuneSS	JuneSR	JulySS	JulySR	AugSS	
01	0946	1753	0921	1839	0830	1927	0723	2018	0622	2105	0540	2148	0539	2200	0613	2
02	0946	1754	0919	1840	0828	1929	0721	2019	0620	2107	0540	2149	0540	2200	0615	2
03	0946	1755	0918	1842	0826	1930	0719	2021	0618	2108	0539	2150	0540	2200	0616	2
04	0946	1756	0916	1844	0824	1932	0717	2022	0617	2110	0538	2151	0541	2159	0618	2
05	0945	1757	0915	1845	0822	1934	0714	2024	0615	2111	0538	2152	0542	2159	0619	2
06	0945	1758	0913	1847	0820	1935	0712	2026	0613	2113	0537	2152	0543	2158	0621	2
07	0945	1800	0912	1849	0818	1937	0710	2027	0612	2114	0537	2153	0544	2158	0622	2
08	0944	1801	0910	1851	0816	1939	0708	2029	0610	2116	0536	2154	0544	2157	0624	2
09	0944	1802	0908	1852	0813	1940	0706	2030	0608	2117	0536	2155	0545	2156	0625	2
10	0943	1804	0907	1854	0811	1942	0704	2032	0607	2119	0536	2156	0546	2156	0627	2
11	0943	1805	0905	1856	0809	1944	0702	2033	0605	2120	0535	2156	0547	2155	0628	2
12	0942	1806	0903	1858	0807	1945	0659	2035	0604	2122	0535	2157	0548	2154	0630	2

Normal text file length: 3 959 lines: 32 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

---

La fonction readEphemeride est chargé de lire et d'interpréter ce fichier:

```
; FONCTIONS POUR SETUP
;; Lecture fichier éphéméride
to readEphemeride [monthToRead dayToRead]
  file-open "config/ephemeride.csv"
  let isLineFound false
  while [not isLineFound or not file-at-end?][
    let line (csv:from-row file-read-line ";")
    if first line = dayToRead [
      set isLineFound true

      ;;; Lecture données Janvier
      if monthToRead = 1 [
        set sunriseHour (item 1 line)
        set sunsetHour (item 2 line)
      ]

      ;;; Lecture données Février
      if monthToRead = 2 [
        set sunriseHour (item 3 line)
        set sunsetHour (item 4 line)
      ]

      ;;; Lecture données Mars
      if monthToRead = 3 [
        set sunriseHour (item 5 line)
        set sunsetHour (item 6 line)
      ]
    ]
  ]
```

La procédure commence par ouvrir le fichier CSV à l'emplacement spécifié à l'aide de la primitive file-open. Ensuite, une boucle while est utilisée pour lire chaque ligne du fichier jusqu'à ce qu'une ligne correspondant à la journée recherchée (dayToRead) soit trouvée ou jusqu'à la fin du fichier. À chaque itération de la boucle, la ligne est lue à l'aide de file-read-line et convertie en une liste de valeurs CSV à l'aide de csv:from-row. La première valeur de la liste (first line) correspond au jour de la ligne. Si le jour correspond à dayToRead, alors les heures de lever et de coucher du soleil sont extraites de la liste et stockées dans les variables sunriseHour et sunsetHour en fonction du mois de recherche (monthToRead). Les indices des heures de lever et de coucher du soleil dans la liste sont déterminés en fonction du mois. Une fois la ligne trouvée et les heures de lever et de coucher du soleil extraites, la boucle est interrompue en définissant la variable isLineFound sur true.



---

```
]`  
file-close  
  
;;; Formatage données temporelles de l'éphéméride vers objet time  
;;; Formatage heures en HH:mm  
if length (word sunriseHour) < 4 [  
  set sunriseHour (word "0" sunriseHour)  
]  
if length (word sunsetHour) < 4 [  
  set sunsetHour (word "0" sunsetHour)  
]  
set sunriseHour (word (substring (word sunriseHour) 0 2) ":" (substring (word sunriseHour) 2 4))  
set sunsetHour (word (substring (word sunsetHour) 0 2) ":" (substring (word sunsetHour) 2 4))  
  
set datetimeSunrise time:create (word year "-" month "-" day " " sunriseHour)  
set datetimeSunset time:create (word year "-" month "-" day " " sunsetHour)  
end
```

Après la boucle, le fichier est fermé à l'aide de `file-close`. Les heures de lever et de coucher du soleil sont ensuite formatées en utilisant des opérations sur les chaînes de caractères pour s'assurer qu'elles sont au format "HH:mm" (par exemple, en ajoutant un zéro devant si nécessaire). Enfin, les objets de date et d'heure pour le lever et le coucher du soleil sont créés en utilisant les informations de date (year, month, day) et les heures formatées.

---

```

;; Luminosité
to setupLightOutside
  ;; Lecture fichier Ephéméride
  readEphemeride month day

  ;; Initialisation variable Luminosité extérieur
  ;; Si avant lever soleil ou après coucher soleil alors nuit noire
  if (time:is-before? currentDateTime (time:plus datetimeSunrise -45 "minutes") or time:is-after? currentDateTime (time:plus datetimeSunset 45 "minutes")) [
    set luminosityOutside 0
    set isNight true
  ]
  ;; Si après lever soleil ou avant coucher soleil alors jour
  if (time:is-after? currentDateTime (time:plus (time:plus datetimeSunrise 1 "days") 45 "minutes") or time:is-before? currentDateTime (time:plus datetimeSunset 45 "minutes")) [
    set luminosityOutside outsideMaxLuminosity
    set isNight false
  ]
  ;; Si aube
  if (time:is-after? currentDateTime datetimeSunrise and time:is-before? currentDateTime (time:plus datetimeSunrise 45 "minutes")) [
    set luminosityOutside (time:difference-between datetimeSunrise currentDateTime "seconds") * (outsideMaxLuminosity / 2700)
    set isNight true
  ]
  ;; Si crépuscule
  if (time:is-after? currentDateTime datetimeSunset and time:is-before? currentDateTime (time:plus datetimeSunset 45 "minutes")) [
    set luminosityOutside outsideMaxLuminosity - (time:difference-between currentDateTime datetimeSunset "seconds") * (outsideMaxLuminosity / 2700)
    set isNight false
  ]
end

```

La procédure `setupLightOutside` utilise les informations d'éphéméride lues dans la procédure précédente pour configurer la luminosité extérieure. La procédure appelle d'abord `readEphemeride` pour lire les données l'éphéméride du jour actuel (month et day) créé par [setupDate](#). Ensuite, la procédure vérifie différentes conditions pour déterminer si c'est le jour ou la nuit. Par exemple, si l'heure actuelle est avant le lever du soleil ou après le coucher du soleil, alors il fait nuit et la luminosité extérieure est définie sur 0. Si l'heure actuelle est après le lever du soleil et avant le coucher du soleil, alors c'est le jour et la luminosité extérieure est définie sur `outsideMaxLuminosity`, une valeur prédéfinie pour la luminosité maximale en plein jour.

Si l'heure actuelle est pendant l'**aube** (entre le lever du soleil et 45 minutes après), la luminosité extérieure est calculée en fonction de la différence de temps entre l'heure actuelle et le lever du soleil. La formule utilisée tient compte de la durée de l'aube (2700 secondes) pour ajuster la luminosité. Si l'heure actuelle est pendant le **crépuscule** (entre le coucher du soleil et 45 minutes après), la luminosité extérieure est calculée en soustrayant la différence de temps entre l'heure actuelle et le coucher du soleil de la luminosité maximale en plein jour. La formule utilisée tient également compte de la durée du crépuscule (2700 secondes) pour ajuster la luminosité.

---

## Intérieur initial

```
;; Luminosité intérieure
to setupLightInside
  ;; Si il y a au moins un volet ouvert, luminosité pièce = luminosité extérieur
  ask shutters with [isOpen = true][
    ;; Si c'est un volet entrée
    if any?(neighbors with [pcolor = 64.7])[
      set luminosityEntrance luminosityOutside
    ]
    ;; Si c'est un volet pièce principale
    if any?(neighbors with [pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3])[
      set luminosityPrincipalRooms luminosityOutside
    ]
    ;; Si c'est un volet à côté de patch marron, regarder les patches à côté
    if any?(neighbors with [pcolor = 23.3])[
      let shutterRoom ""
      ask neighbors with [pcolor = 23.3][
        if any?(neighbors with [pcolor = 64.7])[
          set shutterRoom "E" ;Entrée
        ]
        if any?(neighbors with [pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3])[
          set shutterRoom "PP" ;Pièces principales
        ]
      ]
      if shutterRoom = "E" [
        set luminosityEntrance luminosityOutside
      ]
      if shutterRoom = "PP" [
        set luminosityPrincipalRooms luminosityOutside
      ]
    ]
  ]
]
```

La procédure **setupLightInside** est utilisée pour configurer la luminosité à l'intérieur de l'appartement. La procédure commence par interroger tous les volets (shutters) qui sont ouverts (isOpen = true), pour chaque volet ouvert, plusieurs conditions sont vérifiées en utilisant la primitive any? pour vérifier s'il y a des patches voisins ayant certaines couleurs (pcolor) spécifiques. Les différentes couleurs sont utilisées pour représenter différents types de pièces (par exemple, 64.7 représente l'entrée, 14.4 ou 44.4 peuvent représenter les pièces principales). Si le volet est à côté de l'entrée, la luminosité de l'entrée (luminosityEntrance) est définie sur la luminosité extérieure (luminosityOutside).

---

## Simulation

La simulation de la luminosité extérieure et intérieure est gérée par une seule procédure `lightManagement`.

```
; Gestion de la lumière
to lightManagement
;; Lumière extérieure
;;; Aube
if (time:is-after? currentDateTime datetimeSunrise and time:is-before? currentDateTime datetimeSunset and isNight)[
  if luminosityOutside >= 0 and luminosityOutside < outsideMaxLuminosity [
    set luminosityOutside (luminosityOutside + (outsideMaxLuminosity / 2700))
  ]
  if luminosityOutside < 0 [
    set luminosityOutside 0
  ]
  if luminosityOutside > outsideMaxLuminosity[
    set luminosityOutside outsideMaxLuminosity
  ]
  if luminosityOutside = outsideMaxLuminosity[
    set isNight false
  ]
]
```

La première partie est dédiée à la gestion de la **lumièrre extérieure**. La première partie concerne l'aube soit si l'heure actuelle est après l'heure du lever du soleil (`datetimeSunrise`) et avant l'heure du coucher du soleil (`datetimeSunset`) de l'éphéméride et qu'il fait nuit (vérifié par le booléen `isNight`). Dans ce cas là, si la luminosité extérieure est comprise entre 0 et la luminosité maximale (`outsideMaxLuminosity`), la luminosité extérieure est augmentée progressivement. Si la luminosité extérieure est inférieure à 0, elle est fixée à 0 car on ne peut pas avoir une luminosité négative. Si la luminosité extérieure est supérieure à la luminosité maximale, elle est fixée à la luminosité maximale. Si la luminosité extérieure est égale à la luminosité maximale autorisée, cela signifie que la nuit est terminée, donc `isNight` est défini sur faux.

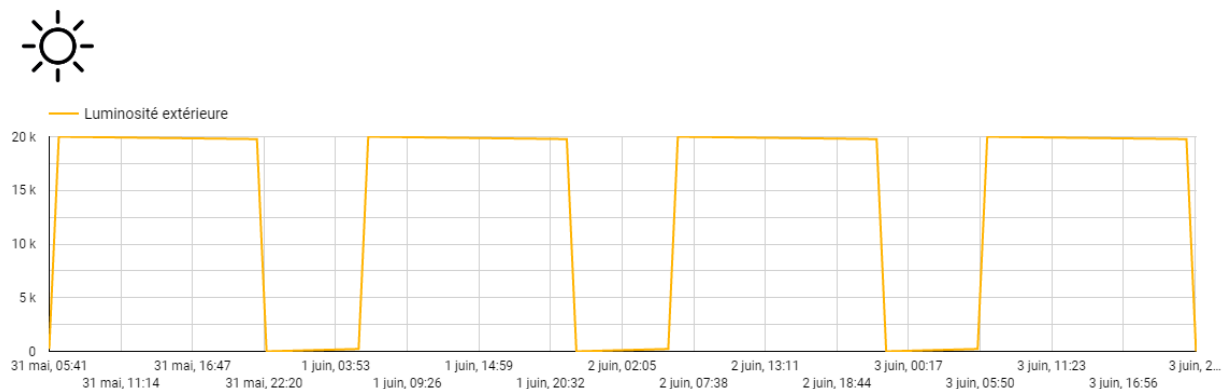
```

;;; Crépuscule
if (time:is-after? currentDateTime datetimeSunset and not isNight)[
  if luminosityOutside > 0 [
    set luminosityOutside (luminosityOutside - (outsideMaxLuminosity / 2700))
  ]
  if luminosityOutside < 0 [
    set luminosityOutside 0
  ]
  if luminosityOutside = 0 [
    set isNight true
  ]
]

```

La deuxième partie concerne le crépuscule, soit si le moment actuel est après l'heure du coucher du soleil (datetimeSunset) et que ce n'est pas la nuit (isNight est faux). Si la luminosité extérieure est supérieure à 0, elle est diminuée de  $\text{outsideMaxLuminosity} / 2700$ . Et ensuite elle procède aux vérifications de champs de valeurs: Si la luminosité extérieure est inférieure à 0, elle est fixée à 0. Enfin, Si la luminosité extérieure est égale à 0, cela signifie que la nuit a commencé, donc isNight est défini sur vrai.

Au final on se retrouve avec une luminosité extérieure comme ceci:



---

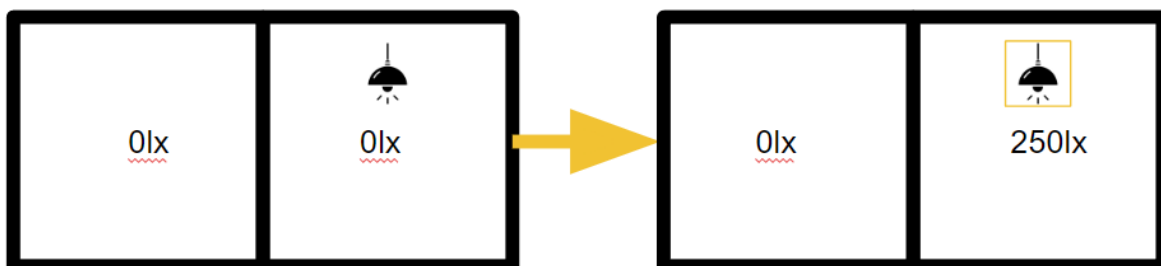
```

;; Lumière intérieure
;;; Gestion de la luminosité des pièces sans fenêtre (Salle de bain)
if not any?(lights with [pcolor = 84.9 and isActive])[
  set luminosityBathroom 0
]
;;; Gestion de la luminosité des pièces avec fenêtres sans volets ouverts
;;; Pièces principales
if not any?(shutters with [any?(neighbors with[pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3]) and isOpen])[
  let principalRoomsMaxLuminosity 0
  ask lights with[(pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3) and isActive][
    set principalRoomsMaxLuminosity luminosity
  ]
  set luminosityPrincipalRooms principalRoomsMaxLuminosity
]
;;; Entrée
if not any?(shutters with [any?(neighbors with[pcolor = 64.7]) and isOpen])[
  let entranceMaxLuminosity 0
  ask lights with[pcolor = 64.7 and isActive][
    set entranceMaxLuminosity luminosity
  ]
  set luminosityEntrance entranceMaxLuminosity
]

```

La deuxième partie de la procédure `lightManagement` concerne la gestion de la **lumière intérieure**. La procédure commence par mettre à 0 la luminosité si nécessaire, et ce pour chaque pièce.

La salle de bain a comme spécificité de n'avoir aucune fenêtre. De ce fait, la luminosité de la salle de bain est gérée uniquement en fonction de l'état des lampes. Si aucune lampe n'est active et présente dans la salle de bain, la luminosité est fixée à 0.



---

La luminosité des pièces principales est gérée en fonction de l'état des volets et des lampes. Si aucun volet n'est ouvert dans les pièces principales (pcolors 14.4, 44.4 ou 126.3) et si aucune lampe n'est activée, la luminosité des pièces principales est fixée à 0.

La luminosité de l'entrée, qui possède également des fenêtres, est gérée de manière similaire aux pièces principales, mais en vérifiant les volets et les lampes correspondant à l'entrée (pcolor = vert = 64.7).

---

```

;;; Gestion lumière naturelle (Volets)
ask shutters with [isOpen = true][
  ;;;; Volet entrée
  if any?(neighbors with [pcolor = 64.7])[
    ;;;; Check luminosité lampe entrée
    let entranceMaxLuminosity luminosityOutside
    ask lights with [isActive and pcolor = 64.7][
      if luminosity > entranceMaxLuminosity[
        set entranceMaxLuminosity luminosity
      ]
    ]
    set luminosityEntrance entranceMaxLuminosity
  ]

  ;;;; Volet pièce principale
  if any?(neighbors with [pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3])[
    ;;;; Check luminosité lights pièces principales
    let principalRoomsMaxLuminosity luminosityOutside
    ask lights with [isActive and (pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3)][
      if luminosity > principalRoomsMaxLuminosity[
        set principalRoomsMaxLuminosity luminosity
      ]
    ]
    set luminosityPrincipalRooms principalRoomsMaxLuminosity
  ]
]

```

La gestion de la lumière naturelle (via les volets) est effectuée en vérifiant les volets qui sont ouverts (isOpen = true). Pour chaque pièce la luminosité maximale est déterminée en vérifiant les lampes actives et la luminosité extérieure, généralement c'est la luminosité extérieure qui l'emporte.



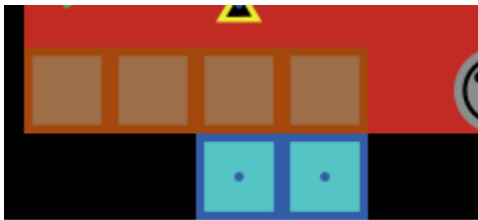
---

```

;;;; Si volet à côté de patch marron, regarder les patches à côté pour trouver la pièce
if any?(neighbors with [pcolor = 23.3])[
  let shutterRoom ""
  ask neighbors with [pcolor = 23.3][
    if any?(neighbors with [pcolor = 64.7])[
      set shutterRoom "E" ;Entrée
    ]
    if any?(neighbors with [pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3])[
      set shutterRoom "PP" ;Pièces principales
    ]
  ]
  if shutterRoom = "E"[
    ;;;; Check luminosité lampe entree
    let entranceMaxLuminosity luminosityOutside
    ask lights with [isActive and pcolor = 64.7][
      if luminosity > entranceMaxLuminosity[
        set entranceMaxLuminosity luminosity
      ]
    ]
    set luminosityEntrance entranceMaxLuminosity
  ]
  if shutterRoom = "PP"[
    ;;;; Check luminosité lights pièces principales
    let principalRoomsMaxLuminosity luminosityOutside
    ask lights with [isActive and (pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3)][
      if luminosity > principalRoomsMaxLuminosity[
        set principalRoomsMaxLuminosity luminosity
      ]
    ]
    set luminosityPrincipalRooms principalRoomsMaxLuminosity
  ]
]
]

```

Il se peut qu'une fenêtre soit complètement entouré soit de murs (pcolor 0) soit de meubles (pcolor = marron = 23.3):



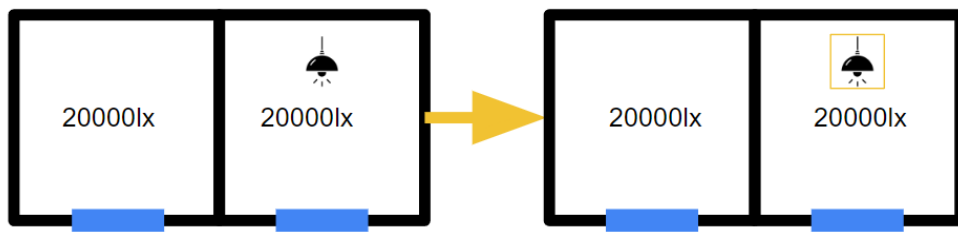
Dans ce cas là, on détermine la pièce de la fenêtre en demandant la couleur des voisins des patches marrons. Enfin, le volet a le même comportement que les autres.

---

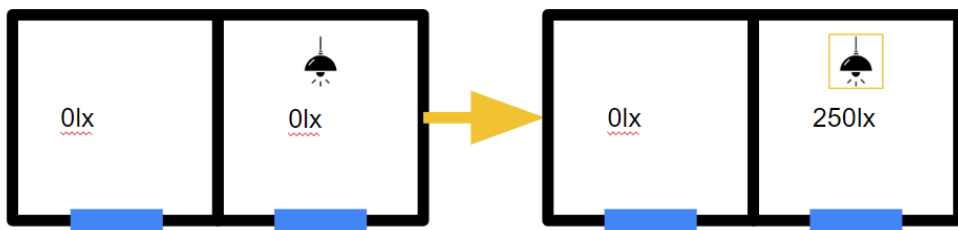
```
;;; Lumière des lampes
ask lights with [isActive = true][
  ;;;; Si luminosité inférieur à celui de la lampe (ex: nuit/volets fermés)
  ;;;; Pièces principales
  if (pcolor = 14.4 or pcolor = 44.4 or pcolor = 126.3) and luminosityPrincipalRooms < luminosity [
    set luminosityPrincipalRooms luminosity
  ]
  ;;;; Entrée
  if pcolor = 64.7 and luminosityEntrance < luminosity [
    set luminosityEntrance luminosity
  ]
  ;;;; Salle de bain
  if pcolor = 84.9 and luminosityBathroom < luminosity [
    set luminosityBathroom luminosity
  ]
]
```

Pour chaque lampe allumée, la luminosité est ajustée en fonction de différentes conditions. Si la luminosité ambiante (luminosité des pièces principales, de l'entrée ou de la salle de bain) est inférieure à celle de la lampe, la luminosité ambiante est mise à jour avec la luminosité de la lampe. Cela permet de faire en sorte que la lumière naturelle l'emporte sur la lumière artificielle.

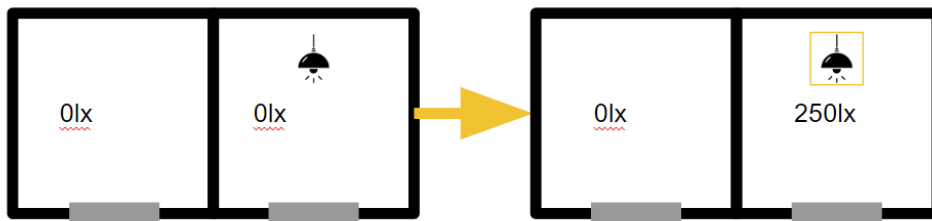
Nous pouvons donc résumer le système de luminosité avec les schéma suivants :



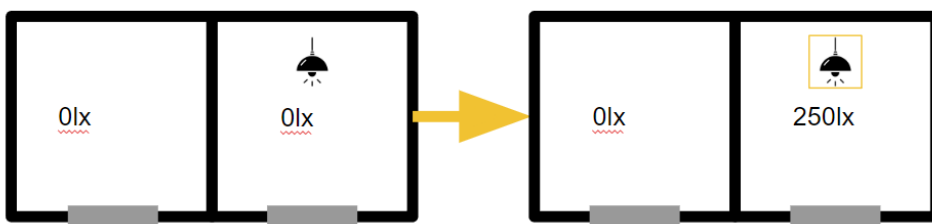
Ouverts



Ouverts



Fermés



Fermés

---

## Saleté

Les pièces deviennent plus sales avec le temps.

```

; Gestion de la saleté
to dirtManagement
  ;; On assume qu'une pièce met 1 semaine à être sale
  ;;; Entrée
  if dirtEntrance < 100 [
    set dirtEntrance dirtEntrance + (100 / 604800)
  ]
  if dirtEntrance > 100 [
    set dirtEntrance 100
  ]
  ;;; Chambre
  if dirtBedroom < 100 [
    set dirtBedroom dirtBedroom + (100 / 604800)
  ]
  if dirtBedroom > 100 [
    set dirtBedroom 100
  ]
  ;;; Salle à manger
  if dirtDiningroom < 100 [
    set dirtDiningRoom dirtDiningRoom + (100 / 604800)
  ]
  if dirtDiningRoom > 100 [
    set dirtDiningRoom 100
  ]
  ;;; Cuisine
  if dirtKitchen < 100 [
    set dirtKitchen dirtKitchen + (100 / 604800)
  ]
  if dirtKitchen > 100 [
    set dirtKitchen 100
  ]
  ;;; Salle de bain
  if dirtBathroom < 100 [
    set dirtBathroom dirtBathroom + (100 / 604800)
  ]
  if dirtBathroom > 100 [
    set dirtBathroom 100
  ]
end
```

Cette mécanique est assurée par la procédure `dirtManagement` qui va faire progressivement monter le niveau de saleté de chaque pièce.

---

## Agents

Le modèle, étant orienté agents, est composé de différents agents avec tous différents comportements et attributs.

## Patches

Les patches représentent les cases du monde du modèle, leur couleurs ont été chargées [lors du setup](#).

```
; VARIABLES PATCHS POUR PATHFINDING
patches-own
[
  parent-patch ;; Prédécesseur du patch
  ;; Variables pour A*
  f
  g
  h
]
```

Elles possèdent des attributs afin d'y appliquer [un pathfinding avec l'algorithme A\\*](#).

Les patches noirs (murs), bleus (fenêtres) et marrons (meubles) ne peuvent être "marché dessus", cela veut dire que la fonction de pathfinding ne prendra pas en compte ces patches pour l'établissement du chemin.

Cependant, les patches marrons peuvent être "monté dessus" si la tâche le permet. Par exemple lorsque l'utilisateur montera sur le lit pour aller dormir.

---

## Meubles

Les pièces sont dotées de différents meubles qui ont tous un comportement différent. Ces meubles sont représentés par des turtles.

### Communs

Certains meubles peuvent être présents dans toutes les pièces, c'est le cas des lampes, des chauffages et des climatiseurs.



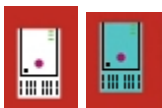
Lampe éteinte/allumée

Les lampes peuvent être allumées ou éteintes, si une lampe est allumée elle éclaire la pièce grâce à la [procédure de gestion de la lumière intérieure](#).



Chauffage éteint/allumé

Le chauffage fait monter la température de la pièce. Leur puissance en W est configurable sur l'interface.



Climatiseur éteint/allumé

A l'inverse, le climatiseur fait baisser la température, leur puissance est également configurable.

Le rôle du chauffage et du climatiseur est d'appliquer le thermostat.



La table est utilisée pour poser des objets.



Les portes peuvent être verticales ou horizontales en fonction des murs

Enfin, les portes sont aussi représentées par des turtles en plus des patches de couleurs grises. Elles possèdent un attribut `isOpen` déterminant si oui ou non la porte est ouverte.

### Salle de bain

La salle de bain est équipée de différents meubles.



Les toilettes servent à assouvir le besoin associé. Ils deviennent actifs lorsque la chasse est tirée après chaque utilisation. Les toilettes possèdent un réservoir et un débit de remplissage, ces deux données sont exportées.



Le lavabo est utilisé après avoir utilisé les toilettes pour se laver les mains. Le fait de se laver les mains fait augmenter l'attribut associé (`cleanliness`). Il possède des attributs de température de l'eau et de débit qui sont exportées. Lorsqu'il est utilisé l'eau est chauffée jusqu'à son thermostat (38°C) et le débit est à 100.



---

La douche sert à nettoyer l'utilisateur en faisant monter l'attribut associé. De la même manière que le lavabo, l'eau est chauffée jusqu'à son thermostat, le débit et la température de l'eau sont exportés.



Le lave-linge est utilisé lorsque l'utilisateur manque de linge, dans ce cas là il place le linge dedans et l'active. Pendant le fonctionnement, le linge est nettoyé mais devient humide. Le lave-linge peut récupérer le taux de saleté moyen du linge qu'il contient en faisant la formule suivante :  $100 - \text{moyenne}(\text{propreté des linges})$ . Il mesure aussi le poids du linge qu'il contient. Ces deux données seront exportées. Il s'arrête lorsque son minuteur atteint 0



Le sèche linge sert après l'utilisation du lave-linge, le linge humide est placé dedans puis il est activé. Quand il est activé, il fait monter sa température et fait diminuer l'humidité du linge à l'intérieur. De la même façon que le lave-linge, il fonctionne avec un minuteur et mesure le poids du linge. Il exporte comme données le poids du linge, l'humidité moyenne et la sa température.

### Chambre



Lorsque l'utilisateur est fatigué, il va au lit. Le lit possède un matelas connecté capable d'analyser le sommeil de l'utilisateur. Le turtle lit s'active lorsqu'il détecte un utilisateur dessus.





---

La commode sert de stockage de linge propre. Il possède un attribut laundryQuantity qui est mis à jour à chaque tick avec la quantité de linge propre qu'il contient. Contrairement aux meubles abordés jusqu'à présent, il ne possède pas de capteur et n'exporte pas de données.

### Salle à manger



La station de roomba est utilisée lorsque le roomba est à court de batterie. Elle recharge les roombas qui sont dessus.



L'utilisateur s'assoit sur la chaise dans le cadre de certaines tâches, il a été prévu de faire monter son besoin de confort. (cf. [Idées mis de côté](#))

### Cuisine



La cuisine comporte une machine à café. Elle sert à faire un café sur la "vaisselle" qu'elle contient. Elle possède différents attributs comme le niveau de l'eau, le niveau de café et la température du café, ces données sont les données exportées. Lorsqu'elle est active, elle chauffe l'eau et génère un café si elle n'existe pas déjà. Si le café existe, elle fait monter son attribut de "quantité".



---

L'utilisateur peut faire la cuisine en utilisant la plaque chauffante. Lorsqu'elle est active, elle chauffe et chauffe les repas situés sur le même patch. Elle fonctionne avec un minuteur et exporte sa température.

Sur le même patch que la plaque il y a la hotte. La hotte a été conçue pour limiter l'émission de fumée lors de la cuisson (cf. [Idées mis de côté](#)).



Le réfrigérateur sert quant à lui à stocker des fruits, des légumes, de la viande et des restes. Les quantités de ces différents produits sont exportées. Il possède une température (2°C) qui est également exportée. Les objets à l'intérieur sont refroidis jusqu'à cette température.



L'objectif du four était d'être une alternative à la plaque chauffante dans la cuisson des plats. Malheureusement, son utilisation n'a pas pu être implémentée.



Le micro-ondes est utilisé pour réchauffer des restes. Les restes sont représentés par des repas dans le réfrigérateur. Si l'utilisateur veut manger quelque chose il choisira de manger des restes en priorité et si les restes ne sont pas mangeables froids, elles seront réchauffées.



Le placard est le stockage de la vaisselle propre, de la même manière que la commode est le stockage du linge propre. Il possède également un attribut de quantité de contenu et ne possède pas de capteur.



Le lave-vaisselle s'occupe de la vaisselle sale. L'utilisateur y met sa vaisselle sale (cleanliness = 0). Une fois qu'il commence à manquer de vaisselle propre l'utilisateur active le lave vaisselle qui nettoie la vaisselle en 3 cycles :

1. "fill" : Monte sa température d'eau, son niveau d'eau et augmente la propreté de la vaisselle présente.
2. "rinse" : Rince la vaisselle
3. "dry" : Sèche la vaisselle

Il exporte le cycle actuel, la température de l'eau et le degré de salissure.

Il a été prévu d'avoir un système de pastilles lave-vaisselle mais il n'a pas été entièrement implémenté.



L'évier, en l'état, sert de stockage temporaire de vaisselle sale si le lave-vaisselle est actif. L'utilisateur peut également s'y laver les mains. Il partage le même breed et donc même comportement que le lavabo de la salle de bain.



Le panier à linge est le stockage de linge sale de la même manière que la commode l'est pour le linge propre.



Il a été prévu de faire un système d'alarme si la quantité de fumée ou de CO est trop importante. Cependant ce système n'est pas entièrement implémenté.

---

## Objets

Le modèle possède plusieurs types d'objets pouvant être transportés par l'agent utilisateur.



Si l'utilisateur a un petit creux, il va aller chercher un fruit au frigo. Le fruit a un attribut nutrition généré aléatoirement à sa création. Il possède aussi un attribut quantity qui détermine la quantité restante, si cette quantité est à 0, le fruit "meurt".



La viande et les légumes sont utilisés lors de la confection d'un repas. Comme les fruits ils possèdent un attribut aléatoire nutrition.

Si le réfrigérateur manque un des 3 l'user va faire les courses pendant 1h et va en créer des nouveaux.



La vaisselle est un contenant de repas (assiette et couverts) ou de café (tasse). Il possède un attribut de propreté qui est mis à 0 quand il a été utilisé. Auquel cas il sera nettoyé dans le lave-vaisselle.



L'utilisateur peut utiliser un repas pour assouvir sa faim. Pour confectionner un repas, l'utilisateur prend une unité de viande et un légume du réfrigérateur ainsi qu'une vaisselle pour le contenir. Puis il se dirige vers la plaque pour faire cuire le repas. Le repas possède un attribut de température de cuisson qui est aléatoire à chaque création ainsi qu'un attribut de température interne et d'état de cuisson. L'état de cuisson augmente lorsque la température est supérieure à la température de cuisson. Tout comme les fruits, il possède un attribut de quantité et de nutrition, ce dernier est créé à partir de la somme des attributs "nutrition" des ingrédients. Enfin il a un attribut `isEatableCold` qui dit s'il doit être mangé chaud. Auquel cas l'utilisateur utilisera le micro-ondes avant de la manger.



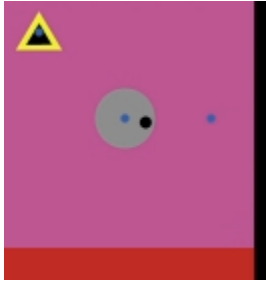
Un café est généré par la machine à café et est contenu dans un agent vaisselle. Lorsque l'utilisateur le boit, il diminue la quantité de café et fait diminuer le besoin de sommeil à celui qui le boit.



L'objet linge représente les vêtements que porte l'utilisateur. Il possède un attribut de propreté comme la vaisselle. Lorsqu'un utilisateur prend sa douche il pose un linge sale par terre puis va aller le poser dans le panier à linge après sa douche et après avoir consommé un linge propre auprès de la commode. Le linge sale a comme attribut de propreté une copie de la statistique de propreté de l'utilisateur. Le linge est nettoyé par le lave-linge et séché par le sèche-linge.

---

## Roomba



Le roomba nettoie la pièce dans laquelle il se situe en faisant diminuer le niveau de saleté de la pièce.

Il possède un niveau de batterie qui diminue au fur et à mesure qu'il se déplace. Lorsque la batterie est faible, il va se diriger vers la station pour aller se recharger. Pour se déplacer il avance en ligne droite jusqu'à trouver un obstacle, auquel cas il va tourner entre 15 et 160 degrés et avancer si il n'y a pas d'obstacle.

Il exporte des données tels que le niveau de batterie et le taux de saleté détecté.

---

## Capteurs

Afin de récupérer les données à exporter, différents capteurs ont été mis en place. Sur le modèle ils prennent la forme de petits points de couleur.

Lors de leur création, les capteurs portent le nom suivant:

Type de capteur + Meuble + Pièce

Bien entendu si le capteur n'est pas connecté à un meuble le nom du capteur ne comportera pas de nom de meuble.

### Luminosité



Le capteur de luminosité capte la luminosité de la pièce à laquelle il se situe. Si le capteur se situe sur une fenêtre il captera la luminosité extérieure.

### Température



Le capteur de température fonctionne de manière similaire à celui de la luminosité, mais cette fois il récupère la température de la pièce ou la température extérieure si il est sur une fenêtre.

### CO/Fumée



Ces capteurs sont censés exporter le taux de Monoxyde de carbone (CO) et le taux de fumée dans la pièce où il se situe. Cependant le système de fumée n'a pas été complètement implémenté rendant ces capteurs inutiles pour l'instant.

---

## Données



Ce capteur permet de récupérer les données à exporter des objets connectés en fonction de l'objet auquel il est lié. Par exemple sur une douche il récupère le débit et la température de l'eau

## Déclenchement

Ce capteur fonctionne comme le capteur de données mais il n'exporte une donnée que si `IsActive` est mis à `true`.

## Ouverture

Ce capteur fonctionne comme le capteur de déclenchement mais se base sur l'attribut `isOpen`.

## Mouvement

Ce capteur émet une donnée s'il détecte un mouvement de l'utilisateur dans la pièce auquel il est présent



---

## Pathfinding

Pour déterminer un chemin d'un point A à un point B des fonctions de pathfinding ont été implémentées. Ces fonctions sont utilisées par le Roomba pour revenir à sa station et par l'agent utilisateur pour se rendre à sa destination.

Les fonction sont des adaptations de l'algorithme A\* trouvables sur ce modèle:  
<https://ccl.northwestern.edu/netlogo/models/community/Astardemo1>

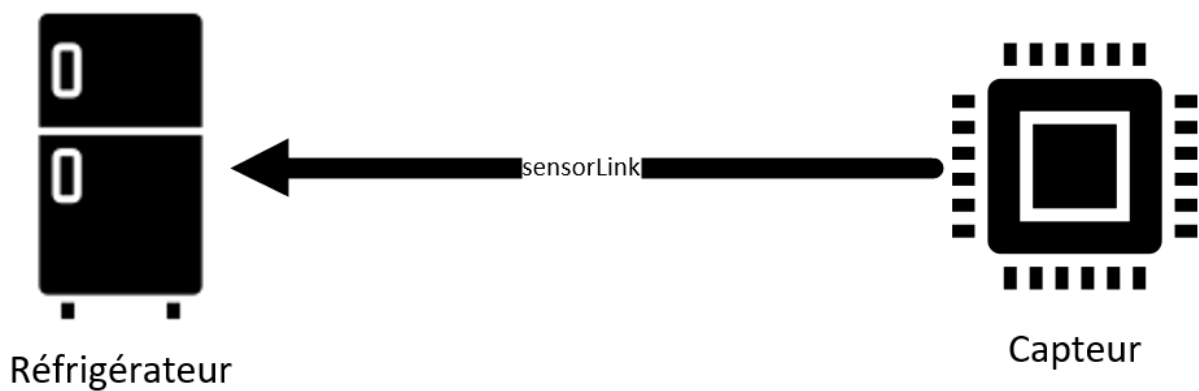
Initialement ces fonctions coloriaient les patches, comme nous nous basons sur ces couleurs pour faire fonctionner le modèle, les fonctions ont été changées pour ne plus toucher aux couleurs de patch.

---

## Links

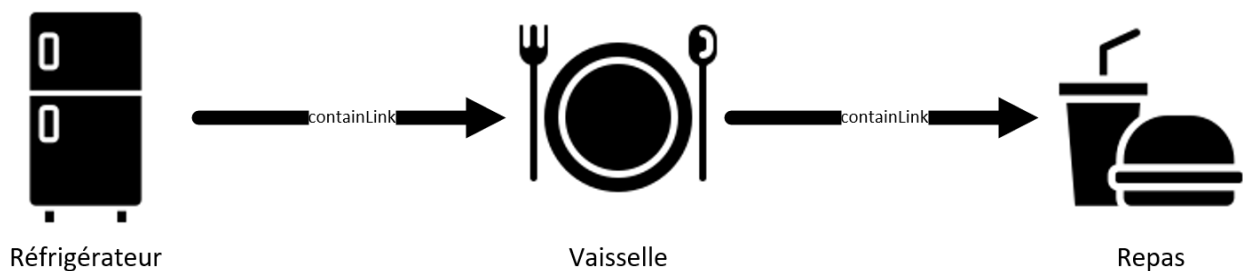
### Liens agents-capteurs

Les liens dits `sensorLinks` sont des liens qui lient des capteurs et des agents. Le lien possède un attribut de mémoire. Cet attribut joue un rôle clé dans l'optimisation de l'export des données, lui permettant de n'exporter que les données différentes du précédent export.



### Liens contenance

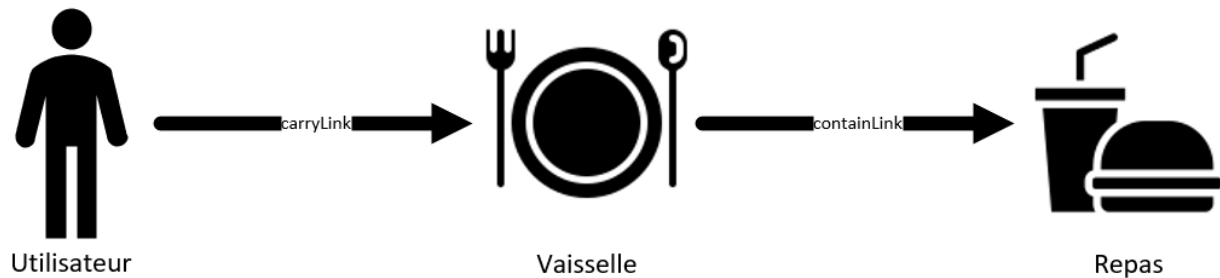
Les liens `containLinks` permettent d'indiquer qu'un objet est situé dans un autre. Un exemple de l'utilisation de ce lien est dans la gestion des restes:



---

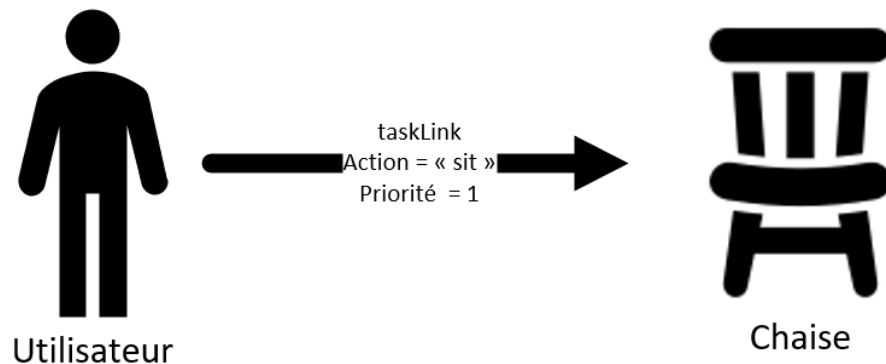
## Liens portance

Le lien carryLink est utilisé lorsqu'un utilisateur porte quelque chose comme une vaisselle, un vêtement ou autre objet.



## Liens tâches

Le système de tâches se base sur le lien entre un utilisateur et des liens taskLinks. Ces liens possèdent un attribut action définissant le type de tâche à réaliser. Ils possèdent également un attribut de priorité qui va jouer dans le choix de la tâche (cf. [Système de tâches](#))



## Utilisateurs



L'utilisateur est l'unique agent capable d'interagir avec des objets et est l'agent le plus complexe d'entre tous.

## Routine

C:\Users\Alexis\Documents\NetLogo\T.config\routine.csv - Notepad++

Fichier Édition Recherche Affichage Encodage Langage Paramètres Outils Macro Exécution Modules d'extension Documents ?

2Weeks2023\_06\_06.csv [2] schéma.csv [2] routine.csv [2]

```
1 1monday,,tuesday,,wednesday,,thursday,,friday,,saturday,,sunday,
2 time,action,time,action,time,action,time,action,time,action,time,action
3 06:00,wake up,06:00,wake up,06:00,wake up,06:00,wake up,06:00,wake up,00:30,sleep,00:30,sleep
4 06:10,take coffee,06:10,take coffee,06:10,take coffee,06:10,take coffee,06:10,take coffee,09:30,
5 07:30,go outside,07:30,go outside,07:30,go outside,07:30,go outside,07:30,go outside,09:40,take c
6 18:30,go back home,18:30,go back home,18:30,go back home,18:30,go back home,18:30,go back home,,
7 19:00,shower,19:00,shower,19:00,shower,19:00,shower,19:00,shower,,16:00,go back home
8 19:45,diner,19:45,diner,19:45,diner,19:45,diner,19:45,diner,,22:00,sleep
9 21:00,read book,21:00,read book,21:00,read book,21:00,read book,21:00,read book,,,
10 22:00,sleep,22:00,sleep,22:00,sleep,22:00,sleep,22:00,sleep,22:00,sleep,,,,
```

CSV Query

SELECT \* FROM THIS

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10	Col11	Col12	Col13	Col14
monday		tuesday		wednesday		thursday		friday		saturday		sunday	
time	action	time	action	time	action	time	action	time	action	time	action	time	action
06:00	wake up	06:00	wake up	06:00	wake up	06:00	wake up	06:00	wake up	00:30	sleep	00:30	sleep
06:10	take coffee	06:10	take coffee	06:10	take coffee	06:10	take coffee	06:10	take coffee	09:30	wake up	09:30	wake up
07:30	go outside	07:30	go outside	07:30	go outside	07:30	go outside	07:30	go outside	09:40	take coffee	09:40	take coffee
18:30	go back home	18:30	go back home	18:30	go back home	18:30	go back home	18:30	go back home			11:30	go outside
19:00	shower	19:00	shower	19:00	shower	19:00	shower	19:00	shower			16:00	go back home
19:45	diner	19:45	diner	19:45	diner	19:45	diner	19:45	diner			22:00	sleep
21:00	read book	21:00	read book	21:00	read book	21:00	read book	21:00	read book				
22:00	sleep	22:00	sleep	22:00	sleep	22:00	sleep						

L'utilisateur doit suivre une routine retranscrite dans le fichier routine.csv. Dans ce fichier chaque jour de la semaine peut avoir des actions (colonne action) à certaines heures (colonne time). Une fois l'heure atteinte, une tâche de basse priorité associée à l'action dans le fichier routine est créée.

---

## Tâches

Pour réaliser ses actions, l'utilisateur va d'abord se rendre vers la destination de la tâche (l'autre bout du lien taskLink) avec les fonctions de [pathfinding](#). Ensuite en fonction de l'action de la tâche définie dans l'attribut action, l'utilisateur va effectuer divers interactions avec la cible et son environnement (cf. [Système de tâches](#)).

## Besoins

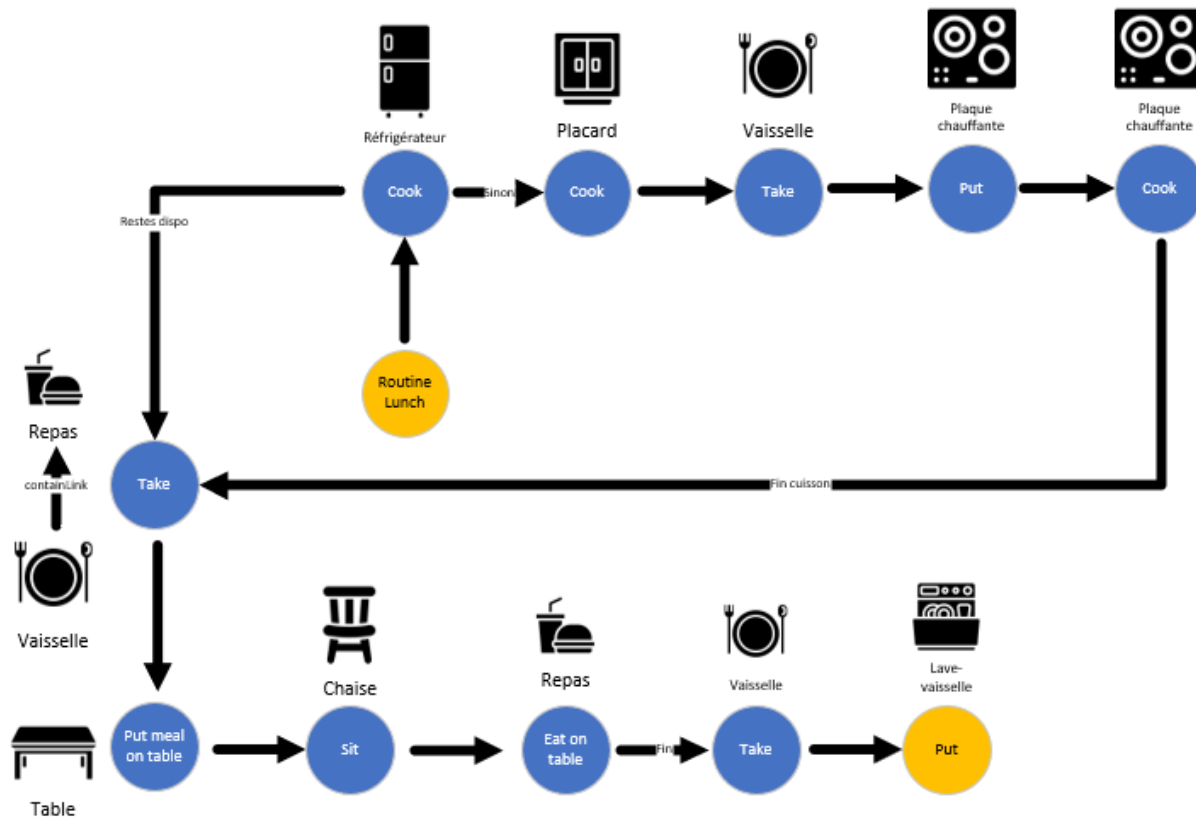
L'utilisateur possède des besoins qu'il doit assouvir, pour ce faire une génération automatique de tâche en cas de besoin faible a été mise en place. Par exemple, si l'utilisateur veut aller aux toilettes et qu'il n'a pas d'autre tâche en cours, il va se rendre immédiatement aux toilettes via la génération de la tâche correspondante.

---

## Système de tâches

Afin que l'utilisateur puisse faire des actions de façon cohérente. Un système de tâches a été mis en place. Les tâches en elles-mêmes sont des [taskLinks](#). Elles sont créées par la fonction `createTask`. La fonction `doTask` permet de définir l'action d'une tâche. Enfin, la fonction `endTask` peut prendre en entrée une tâche à faire immédiatement après afin de faire enchaîner les tâches sans interruption.

Voici un exemple d'enchaînement de tâches: Faire à manger quand c'est l'heure du déjeuner



## Export des données

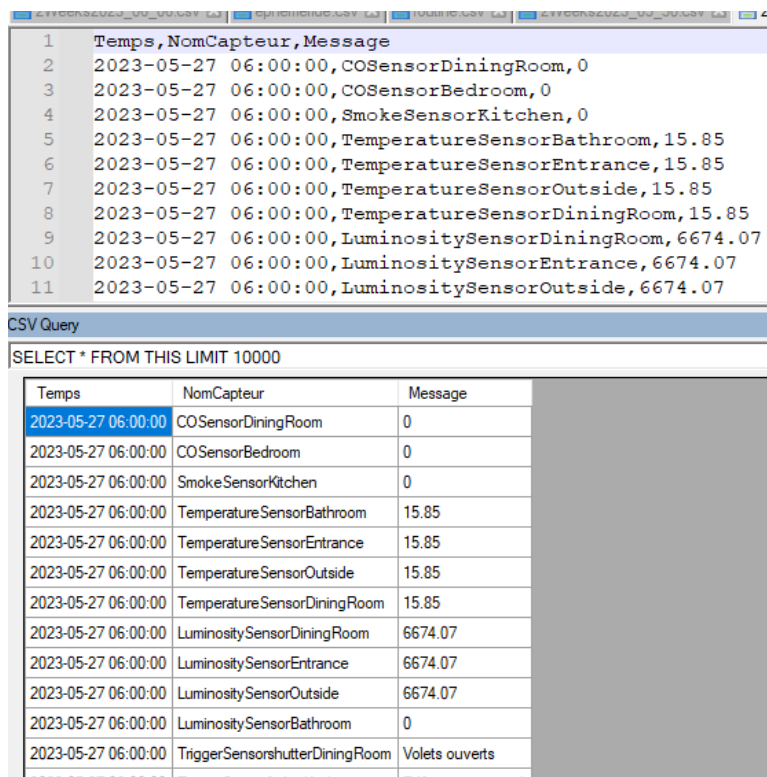
A chaque seconde le modèle se sert d'une variable globale "toWrite" qui est progressivement alimentée par les différents capteurs dans la procédure sensorBehavior.

Le fichier exporté est un fichier CSV avec le temps de d'émission de la donnée, le nom du capteur qui l'a émis et le message. Certains capteurs renvoient uniquement un nombre (Luminosité, Température), d'autres envoient un message (Données).

A chaque seconde, la procédure writeData se charge de retranscrire le contenu de toWrite directement dans le fichier ouvert.

Les données peuvent être limitées à un certain nombre de décimales grâce à la fonction roundMessage qui prend en entrée le message quelque soit la nature et l'arrondi si nécessaire (si c'est un nombre).

Voici à quoi ressemble un fichier de données avec des données exportées.



The image shows a screenshot of a database viewer. At the top, a CSV file is open, displaying 11 lines of data. Below this, a 'CSV Query' section shows a SQL query: 'SELECT \* FROM THIS LIMIT 10000'. Below the query, a table view displays the data with three columns: 'Temps', 'NomCapteur', and 'Message'.

Temps	NomCapteur	Message
2023-05-27 06:00:00	CO SensorDiningRoom	0
2023-05-27 06:00:00	CO SensorBedroom	0
2023-05-27 06:00:00	Smoke SensorKitchen	0
2023-05-27 06:00:00	Temperature SensorBathroom	15.85
2023-05-27 06:00:00	Temperature SensorEntrance	15.85
2023-05-27 06:00:00	Temperature SensorOutside	15.85
2023-05-27 06:00:00	Temperature SensorDiningRoom	15.85
2023-05-27 06:00:00	Luminosity SensorDiningRoom	6674.07
2023-05-27 06:00:00	Luminosity SensorEntrance	6674.07
2023-05-27 06:00:00	Luminosity SensorOutside	6674.07
2023-05-27 06:00:00	Luminosity SensorBathroom	0
2023-05-27 06:00:00	Trigger SensorshutterDiningRoom	Volets ouverts

2Weeks2023\_06\_06.csv ephemeride.csv routine.csv 2Weeks2023\_05\_30.csv 2Weeks2023\_06\_06.csv

```

34 2023-05-27 06:00:00,DataSensorToilet,Capacité réservoir: 100
35 2023-05-27 06:00:00,DataSensorHood,Puissance: 0
36 2023-05-27 06:00:00,DataSensorBed,Qualité du sommeil: 100
37 2023-05-27 06:00:00,DataSensorRoomba,Capacité sac: 0
38 2023-05-27 06:00:00,DataSensorRoomba,Taux de saleté: 0
39 2023-05-27 06:00:00,DataSensorRoomba,Batterie restante: 99.98
40 2023-05-27 06:00:00,DataSensorMicrowave,Puissance: 0
41 2023-05-27 06:00:00,DataSensorOven,Température: 15.85
42 2023-05-27 06:00:00,DataSensorOven,Puissance: 0
43 2023-05-27 06:00:00,DataSensorHotplate,Température: 15.85
44 2023-05-27 06:00:00,DataSensorHotplate,Puissance: 0

```

SV Query

SELECT \* FROM THIS LIMIT 10000

Temps	NomCapteur	Message
2023-05-27 06:00:00	DataSensorHotplate	Puissance: 0
2023-05-27 06:00:00	DataSensorDryer	Humidité: 0
2023-05-27 06:00:00	DataSensorDryer	Température: 15.85
2023-05-27 06:00:00	DataSensorDryer	Poids linge: 0
2023-05-27 06:00:00	DataSensorFridge	Porte ouverte: 0
2023-05-27 06:00:00	DataSensorFridge	Quantité de fruits: 5
2023-05-27 06:00:00	DataSensorFridge	Quantité de légumes: 5
2023-05-27 06:00:00	DataSensorFridge	Quantité de viande: 5
2023-05-27 06:00:00	DataSensorFridge	Quantité de restes: 0
2023-05-27 06:00:00	DataSensorShower	Température de l'eau: 15.85
2023-05-27 06:00:00	DataSensorShower	Débit: 0
2023-05-27 06:00:00	DataSensorCoffeeMaker	Eau restant: 100

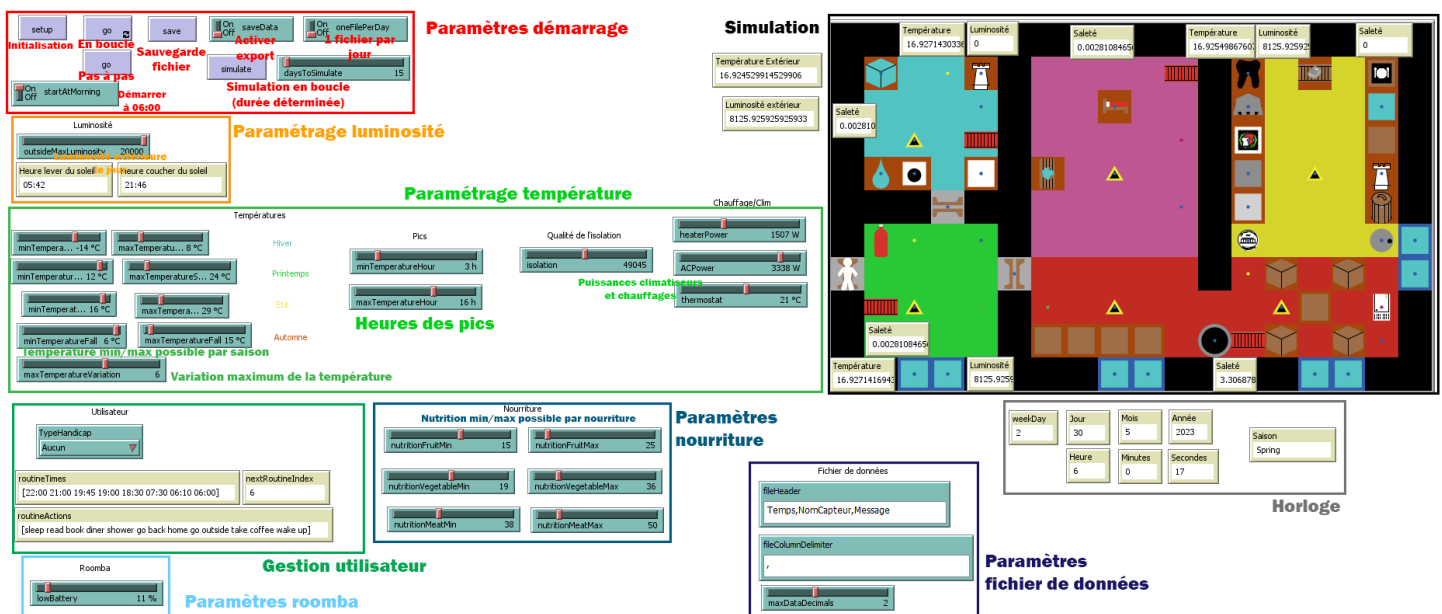


## Interface du modèle

L'interface du modèle est complète et permet de paramétrer minutieusement la simulation.

Elle permet notamment de lancer une simulation à durée déterminée, afin de pouvoir arrêter la simulation après un certain nombre de jours.

Voici un schéma explicatif de l'interface:



L'image complète et agrandissable est inclus dans le livrable ainsi que sur le lien suivant:

[https://drive.google.com/file/d/1ELavnMgkfWiHIPtIOQwt\\_jO8D755yl\\_P/view?usp=share\\_link](https://drive.google.com/file/d/1ELavnMgkfWiHIPtIOQwt_jO8D755yl_P/view?usp=share_link)

---

## Visualisation des données

Une des façons d'exploiter les données produites est en utilisant un outil de visualisation de données pour créer des tableaux de bord interactifs. Pour ce projet nous avons choisi d'utiliser Looker Studio (anciennement Google Data Studio).

Nous avons établi un tableau de bord reprenant 4 jours de données du dataset de test "2Weeks", résultant de 15 jours de simulation.

Ces tableaux de bord sont disponibles sous format PDF dans le livrable ou sur le lien suivant:

<https://lookerstudio.google.com/reporting/7e16e8f1-72ef-4e21-ab77-1edf1f64634c>

Il est possible de choisir la période des données grâce à ce sélecteur:

Sélectionner la période ▾

Période fixe ▾

Date de début

Date de fin

< May 2023 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

< May 2023 >

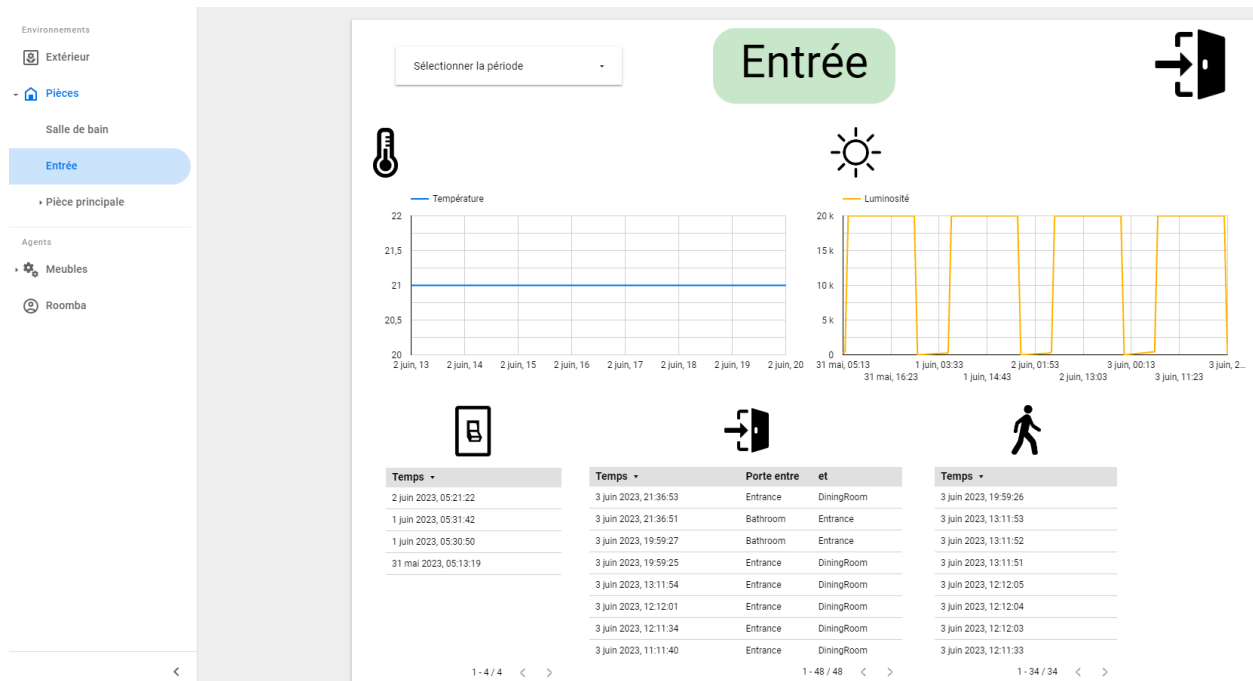
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

ANNULER

APPLIQUER

Voici quelques exemples de tableaux de bord :

Données liées à l'entrée:



Données de la douche :



---

## Problèmes rencontrés

Le développement de ce projet ne s'est pas fait sans embûches. Nous nous sommes heurtés à différents problèmes, techniques ou non.

### Limites de Netlogo

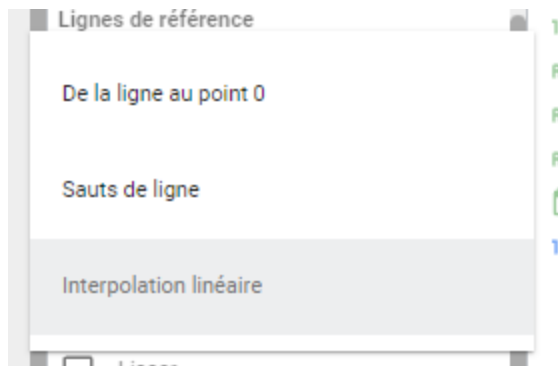
Netlogo comporte de nombreuses limites qui ont entravé le développement du projet. Une de ces limites est le fait de ne pas pouvoir appeler des API REST avec Netlogo, ce qui a compliqué certaines phases du développement. Nous avons su trop tard que l'on pouvait remédier à ce problème grâce à l'extension Py qui permet d'exécuter du code python dans netlogo.

Un autre problème qui a gêné le développement du système de tâches est le fait qu'il ne peut exister deux liens dirigés du même breed entre deux agents. Par exemple, cela était nécessaire quand nous voulions faire en sorte qu'un utilisateur activait le lave vaisselle juste après qu'il ait déposé un objet vaisselle à l'intérieur, et cette limite occasionna beaucoup de bug lors des tests.

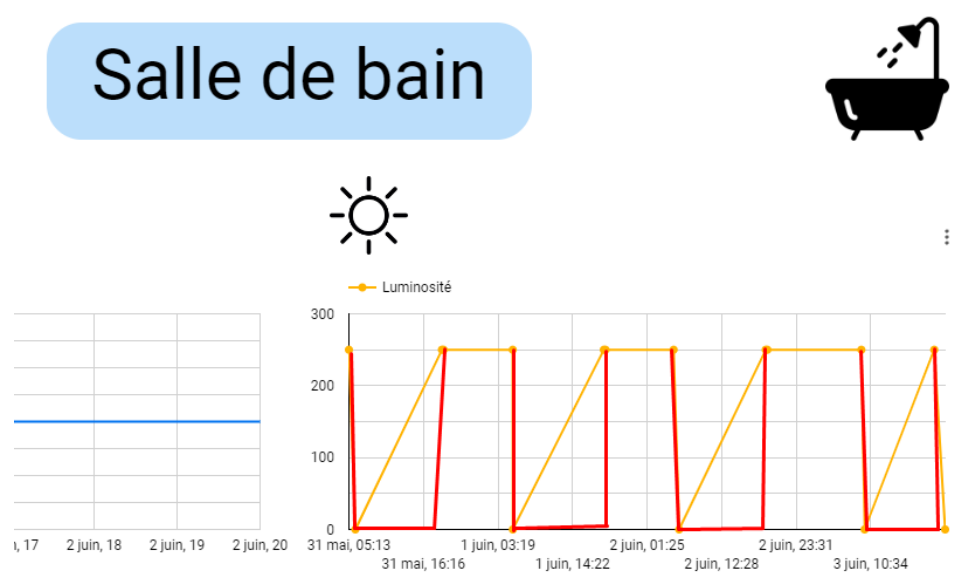
---

## Limites de Looker Studio

Il n'y a pas que sur Netlogo qu'il y a eu des soucis. En effet nous voulions utiliser une interpolation linéaire par étapes sur les graphiques comme ceux de la température ou de la luminosité. Cependant cette fonction n'est disponible que sur Looker, Looker Studio ne pouvant faire que de l'interpolation linéaire classique.



Sur le graphique suivant nous avons tracé en rouge les résultats attendus avec un interpolation par étapes:



---

Une autre limite de Looker Studio est le quota de fichiers importés, on ne peut pas charger l'ensemble du dataset 2Weeks:



## Ambition

Un autre problème mais dont nous sommes les responsables est la trop grande ambition du projet. En effet, il y a eu un manque sévère de temps pour obtenir un résultat satisfaisant et présentable avec l'ensemble des idées cadrées. La grande quantité de deadlines la semaine de la présentation ainsi que l'alternance n'a pas aidé à cela. De ce fait, il a fallu écarter le développement de certains systèmes ( qui sont résumés dans [cette partie](#)).

---

## Axes d'amélioration du modèle

Le modèle comprend beaucoup d'aspects à parfaire et à peaufiner. La plupart ayant été laissés tels quels faute de temps.

### Personnalisation de l'environnement

En utilisant une image comme source du monde, nous ouvrons la voie à une génération dynamique de pièces, cependant cette partie n'est pas implémentée à l'heure actuelle et le modèle fonctionne principalement avec des emplacements écrits en dur.

### Configurabilité

Le code comprend encore beaucoup de formules écrites également en dur. Il est possible de rendre variable ces formules en intégrant le paramétrage à l'interface.

### Maintenabilité

Le code, même si lourdement commenté, possède encore des anciennes façons de procéder qui sont moins lisibles. Par exemple, avant de découvrir la primitive "of" nous utilisions "ask" pour stocker dans une variable temporaire l'attribut qui nous intéressait. De même l'utilisation de la procédure getRoomName n'est pas systématique partout pour récupérer le nom de la pièce où se situe l'agent.

### Assumptions et réalisme

Ce modèle, tout comme plein d'autres, se base sur des assumptions, et ces assumptions peuvent être plus ou moins éloignées de la réalité.

---

## **Idées mises de côté faute de temps**

Initialement, nous avons prévu un système de perturbations comme des canicules/vagues de froid, des fuites de monoxyde de carbone ou encore des incendies. Mécanisme d'incendie qui fût également mis de côté. Les incendies devaient générer de la fumée et du monoxyde de carbone détectable par les capteurs, et ouvrir la fenêtre devait contribuer à diminuer la fumée. De la fumée devait également être émise par le repas durant la cuisson, et cette fumée était captée par la hotte.

Le lave-vaisselle comporte un compteur de pastilles lave-vaisselles qui est exporté en tant que donnée. Cependant ce compteur n'est jamais affecté.

Enfin, l'utilisateur a un besoin de confort qui n'est pas entièrement implémenté. Il devait s'asseoir ou s'allonger si ce besoin était faible.



---

## Références et inspirations.

Les inspiration principales du modèle sont les jeux vidéo "Dwarf fortress" ainsi que "Rimworld". Ces jeux sont en vue de dessus en deux dimensions tout comme le modèle et portent fortement attention sur les détails de chaque élément, comme le système de température des pièces ou d'objets.

Le système de tâches est quant à lui inspiré de celui de la série de jeux "Les Sims", les personnages pouvant exécuter des tâches dans l'ordre dans une queue et peuvent enchaîner des tâches en plusieurs étapes. Par exemple, poser un plat sur la table puis s'asseoir sur une chaise puis manger.

Enfin, comme énoncé dans la partie dédiée, le pathfinding est inspiré du modèle Netlogo "Astardemo" disponible sur le site officiel:

<https://ccl.northwestern.edu/netlogo/models/community/Astardemo1>

---

## Conclusion

En conclusion, ce rapport a présenté un projet ambitieux visant à créer une représentation virtuelle d'un logement équipé de capteurs dans le contexte de l'Internet des objets.

Les objectifs du projet étaient clairs : modéliser trois pièces spécifiques de l'appartement, intégrer des capteurs et simuler les déplacements du résident tout en enregistrant les données correspondantes. La modélisation des pièces, la prise en compte des caractéristiques individuelles et du mobilier, ainsi que l'utilisation d'agents pour représenter les capteurs et la personne ont permis de créer une simulation dynamique et réaliste. Les résultats ont été sauvegardés dans un fichier CSV, offrant une portabilité et une facilité d'exploitation ultérieure des données.

Les difficultés rencontrées et les axes d'amélioration potentiels fournissent ainsi une perspective critique et constructive du travail réalisé.

Ce projet individuel innovant a réussi à atteindre la grande plupart de ses objectifs en proposant une modélisation précise et une simulation réaliste des interactions entre un résident et les objets connectés d'un logement. Les perspectives futures pourraient inclure l'extension du modèle à d'autres pièces ou l'intégration de fonctionnalités supplémentaires pour une expérience plus complète. Les références et les sources d'inspiration utilisées ont guidé et enrichi le travail réalisé.