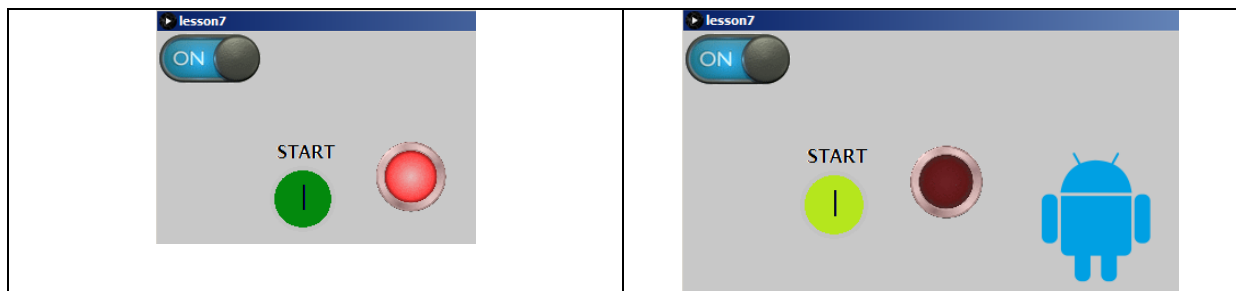


Présentation des classes de l'application

Les classes de l'application permettent de créer des objets qui dessineront des images fixes ou mobiles, réagiront à un clic de souris, propageront des informations numériques vers d'autres objets.

Chaque objet est muni d'une ou plusieurs bornes, les objets utilisent celles-ci pour communiquer entre-elles. Un objet particulier dont la classe est « TerminalBoard » (bornier) permet de réaliser l'interconnexion des objets.

Dans notre premier exemple, on construit un scénario dans lequel un bouton poussoir, un indicateur (lampe) et une image sont dessinés et interagissent via un bornier.



Pour ce scénario il y a donc :

- Un objet du type « Actor », une image fixe qui peut être affichée ou non.
- Un objet du type « Button », fonctionnant comme un bouton poussoir. (START)
- Un objet du type « Button », fonctionnant comme un interrupteur. (ON/OFF)
- Un objet du type « Indicator », fonctionnant comme une lampe.
- Un objet du type « TerminalBoard », non visible à l'écran mais qui joue un rôle essentiel, en réalisant l'interconnexion des objets.

En effet :

1. Agir sur « Start », change l'état de l'indicateur et la visibilité d'une image.
2. Agir sur « ON/OFF », provoque l'arrêt du programme.

Tous les objets nécessaires sont décrits dans un fichier texte au format « JSON ».

Comment placer une image fixe à l'écran

Dans le fichier texte au format « JSON », « defs1.json » par exemple. On définit un tableau d'objets via le mot clé « actors »

```
{ "actors" : [ ] }
```

La ligne ci-dessus déclare un scénario sans acteurs. (Le tableau est vide)

On déclare un tableau contenant un acteur.

```
{ "actors" : [
  {
    "name" : "android", "type" : "Actor",
    "x" : 300, "y" : 100,
    "images" : [ "android.png" ],
    "visible" : true
  }
]
```



La ligne ci-dessus déclare un scénario avec un acteur.

- Le mot clé "name" est obligatoire, il définit le nom (unique) de l'acteur. (android)
- Le mot clé "type" est obligatoire, il définit le type d'objet à utiliser. (Actor)
- Les mots clés "x" et "y" sont obligatoires, ils définissent les coordonnées du coin supérieur gauche de l'image affichée.
- Le mot clé "images" est optionnel, il permet de définir un tableau d'images à utiliser pour dessiner l'objet. Si cette propriété n'est pas définie dans le fichier texte, rien ne sera dessiné. Pour ce type d'objet, une seule image est nécessaire et une seule image sera dessinée.
- Le mot clé "visible" est optionnel, il permet de définir s'il faut ou pas afficher l'objet au démarrage du programme. Si cette propriété n'est pas définie dans le fichier texte, les objets seront visibles par défaut.

Ajouter un bouton poussoir START

On déclare un tableau de deux acteurs. (Un acteur basique et un bouton poussoir)

```
{ "actors" : [
  {
    "name" : "android", "type" : "Actor",
    "x" : 300, "y" : 100,
    "images" : [ "android.png" ],
    "visible" : true
  },
  {
    "name": "start", "type": "Button",
    "x": 100, "y": 100,
    "images": [ "start_off.png", "start_on.png" ]
  }
]
```



- Le mot clé "name" est obligatoire, il définit le nom (unique) de l'acteur. (start)
- Le mot clé "type" est obligatoire, il définit le type d'objet à utiliser. (Button)
- Les mots clés "x" et "y" sont obligatoires, ils définissent les coordonnées du coin supérieur gauche de l'image affichée.
- Le mot clé "images" doit contenir deux images, la première pour l'état OFF et la seconde pour l'état ON.
- Le mot clé "visible" est optionnel, il n'est nécessaire que si on veut cacher ce bouton au démarrage. ("visible" : false)

Ajouter un interrupteur ou bouton poussoir avec verrouillage

```
{ "actors" : [
  {
    "name" : "android", "type" : "Actor",
    "x" : 300, "y" : 100,
    "images" : [ "android.png" ],
    "visible" : true
  },
  {
    "name": "start", "type": "Button",
    "x": 100, "y": 100,
    "images": [ "start_off.png", "start_on.png" ]
  },
  {
    "name": "Quit", "type": "Button",
    "x": 0, "y": 0,
    "images": [ "switch0_off.png", "switch0_on.png" ],
    "interlock": true, "contact": true,
    "appOffAction": "quit"
  }
]
}
```



- Le mot clé "name" est obligatoire, il définit le nom (unique) de l'acteur. (Quit)
- Le mot clé "type" est obligatoire, il définit le type d'objet à utiliser. (Button)
- Les mots clés "x" et "y" sont obligatoires, ils définissent les coordonnées du coin supérieur gauche de l'image affichée.
- Le mot clé "images" doit contenir deux images, la première pour l'état OFF et la seconde pour l'état ON.
- Le mot clé "interlock" définit la présence d'un verrouillage. (true)
- Dans ce cas, le mot clé "contact" permet de définir l'état initial du contact. (true)
- Si on désire utiliser cet interrupteur pour mettre fin au programme on utilise la propriété, "appOnAction" ou "appOffAction" qui définissent l'action à réaliser. (quit)
- Le mot clé "visible" nécessaire que si on veut cacher ce bouton au démarrage. ("visible" : false)

Cet interrupteur ne peut donc pas être utilisé dans la simulation d'un processus.

Actuellement la seule action que peut exécuter le programme via un bouton poussoir ou un interrupteur est nommée "quit". (Quitter le programme)

Ajouter un indicateur

```
{ "actors" : [
  {
    "name" : "android", "type" : "Actor",
    "x" : 300, "y" : 100,
    "images" : [ "android.png" ],
    "visible" : true
  },
  {
    "name": "start", "type": "Button",
    "x": 100, "y": 100,
    "images": [ "start_off.png", "start_on.png" ]
  },
  {
    "name": "Quit", "type": "Button",
    "x": 0, "y": 0,
    "images": [ "switch0_off.png", "switch0_on.png" ],
    "interlock": true, "contact": true,
    "appOffAction": "quit"
  },
  {
    "name": "LED1", "type": "Indicator",
    "x": 200, "y": 100,
    "images": [ "icon_led_red_off.png", "icon_led_red_on.png" ]
  }
]
}
```



- Le mot clé "name" est obligatoire, il définit le nom (unique) de l'acteur. (LED1)
- Le mot clé "type" est obligatoire, il définit le type d'objet à utiliser. (Indicator)
- Les mots clés "x" et "y" sont obligatoires, ils définissent les coordonnées du coin supérieur gauche de l'image affichée.
- Le mot clé "images" doit contenir deux images, la première pour l'état OFF et la seconde pour l'état ON.
- Le mot clé "visible" peut être utilisé.

Le décor est ainsi préparé, il reste maintenant à établir les liaisons (électriques) entre les objets.

Pour réaliser ces connexions, chaque objet dispose de bornes (Terminal) que l'on connectera via un bornier (TerminalBoard)

Le mot clé "terminalBoard" sera utilisé pour déclarer et décrire ce bornier.

Déclaration du bornier



Chaque borne du bornier ne peut accueillir que deux fils. L'un sera le fournisseur d'information (provider) l'autre l'utilisateur de l'information (consumer).

```
{
  "actors" : [ ... ] ,
  "terminalBoard": [
    {
      "provider": { "actor": "start", "terminal": "contact", "normallyClosed": true },
      "consumer": { "actor": "LED1", "terminal": "state", "activeLow": false }
    },
    {
      "provider": { "actor": "start", "terminal": "contact", "normallyClosed": false },
      "consumer": { "actor": "android", "terminal": "visible", "activeLow": false }
    }
  ]
}
```

Notre bornier contient deux paires de bornes. (TerminalConnecor)

- Chaque paire de bornes (TerminalConnecor) réalise la connexion d'un "provider" à un "consumer".
- Un objet fournissant une information peut être raccordé plusieurs fois au bornier, c'est le cas dans notre exemple dans le cas bouton poussoir start.
- Un objet recevant une information ne peut être raccordé qu'une seule fois au bornier.
- Il n'est actuellement pas possible de réaliser des mises en série ou parallèle de bouton poussoir ou interrupteur.

Pour le provider:

- Le mot clé "actor" est obligatoire, il définit le nom de l'acteur. (start)
- Le mot clé "terminal" est obligatoire, il définit l'information demandée. (Contact)
- Le mot clé "normallyClosed" est optionnel, il permet de définir si le contact est vu comme une NO ou NF. (True => NC, false ou mot clé absent => NO)
- Dans notre exemple on utilise un normalement fermé sur la première paire de bornes et un normalement ouvert sur la seconde.

Pour le premier consumer:

- Le mot clé "actor" est obligatoire, il définit le nom de l'acteur. (LED1)
- Le mot clé "terminal" est obligatoire, il définit le destinataire de l'information reçue. (state)
- Le mot clé "activeLow" est optionnel, pour LED1 l'action de l'allumer est réalisée à l'état false.

Pour le second consumer:

- Le mot clé "actor" est obligatoire, il définit le nom de l'acteur. (android)
- Le mot clé "terminal" est obligatoire, le destinataire de l'information reçue. (visible)
- Le mot clé "activeLow" est optionnel, l'action de le rendre visible est réalisée à l'état true.

Actor

Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X		visible
type	X		
x	X		
y	X		
images	X		
visible (boolean)			
scale (float)			

Chaque type d'objet possède les propriétés visible et scale qui sont définies par défaut respectivement à true et 1.0.

Button

Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X	visible	visible
type	X	contact	
x	X		
y	X		
images	X		
interlock			
contact			
appOnAction			
appOffAction			
visible			
scale (float)			

Indicator

Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X		visible
type	X		state
x	X		
y	X		
images	X		
visible			
scale (float)			

Provider - consumer

Provider		consumer	
Mots clés	Obligatoire (X)	Mots clés	Obligatoire (X)
actor	X	actor	X
terminal	X	terminal	X
normallyClosed		activeLow	

Le fichier def1.json

```
{
  "actors": [
    {
      "name": "android",
      "type": "Actor",
      "x": 300, "y": 100,
      "images": [ "android.png" ],
      "visible": true
    },
    {
      "name": "start",
      "type": "Button",
      "x": 100, "y": 100,
      "images": [ "start_off.png", "start_on.png" ]
    },
    {
      "name": "Quit",
      "type": "Button",
      "x": 0, "y": 0,
      "images": [ "switch0_off.png", "switch0_on.png" ],
      "interlock": true,
      "contact": true,
      "appOffAction": "quit"
    },
    {
      "name": "LED1",
      "type": "Indicator",
      "x": 200, "y": 100,
      "images": [ "icon_led_red_off.png", "icon_led_red_on.png" ]
    }
  ],
  "terminalBoard": [
    {
      "provider": { "actor": "start", "terminal": "contact", "normallyClosed": true },
      "consumer": { "actor": "LED1", "terminal": "state", "activeLow": false }
    },
    {
      "provider": { "actor": "start", "terminal": "contact", "normallyClosed": false },
      "consumer": { "actor": "android", "terminal": "visible", "activeLow": false }
    }
  ]
}
```

Pour tester l'un des fichiers "defxx.json" il suffit de modifier une ligne dans le setup.

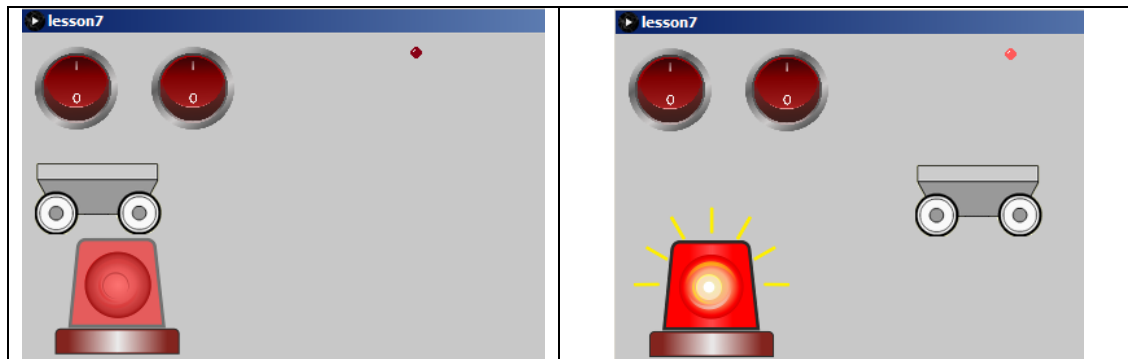
```
// chargement des objets utilisés, via le contenu du fichier defs.json et des images
void setup(){
  //fullScreen(); // on utilise tout l'écran
  size(800,600);

  JSONObject defs=loadJSONObject("def1.json");

  ...
}
```

Test d'un objet se déplaçant horizontalement et d'un capteur de position horizontale

Le scénario est décrit dans le fichier nommé "def2.json".



- Les deux interrupteurs permettent de donner l'ordre au wagon de se déplacer vers la gauche ou la droite. (Via le bornier)
- Lorsque wagon est à la verticale (zone de 10% de la largeur) de la diode Led, le capteur de position horizontale est à l'état vrai.
- Un indicateur est activé par le capteur de position via le bornier.

HorizontalMovingActor (Le wagon)

Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X		visible
type	X		left
x	X		right
y	X		
images	X		
leftSpeed (entier>0)			
rightSpeed (entier>0)			
visible			
scale (float)			

Les vitesses par défaut valent 1.

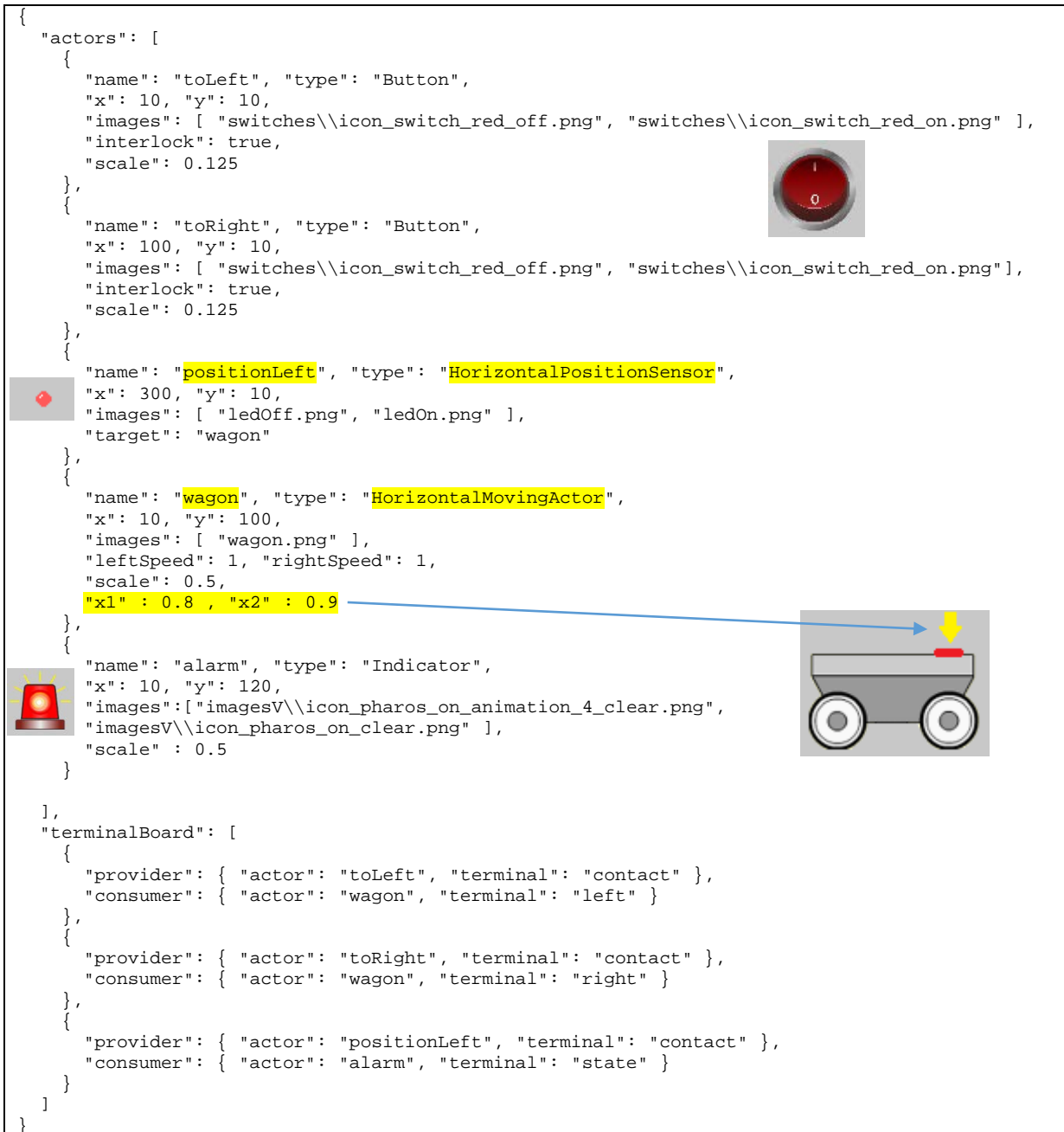
HorizontalPositionSensor (La petite diode led rouge)

Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X	contact	visible
type	X		
x	X		
y	X		
images	X		
target	X (le nom de l'objet à détecter)		
x1 % de la largeur	(0.2 par défaut)		
x2 % de la largeur	(0.8 par défaut)		
visible			
scale (float)			

L'objet "TerminalBoard" est utilisé pour lier :

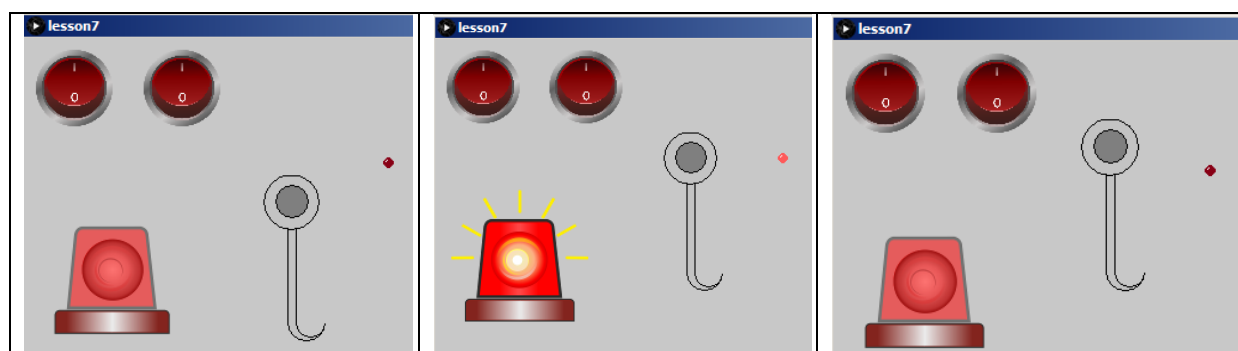
1. Les sorties "contact" des interrupteurs aux entrées "left" et "right" du wagon.
2. La sortie "contact" de capteur à l'entrée "state" du gyrophare.

Le fichier def2.json



Pour les objets se déplaçant horizontalement, la commande simultanée de "left" et "right" ne provoque aucun déplacement de l'objet.

Test d'un objet se déplaçant horizontalement et d'un capteur de position horizontale
Le scénario est décrit dans le fichier nommé "def3.json".



- Les deux interrupteurs permettent de donner l'ordre au crochet de se déplacer vers le haut ou le bas. (Via le bornier)
- Lorsque crochet est à l'horizontale (15% de la hauteur) de la diode Led, le capteur de position vertical est à l'état vrai.
- Un indicateur est activé par le capteur de position via le bornier.

VerticalMovingActor (Le crochet)

Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X		visible
type	X		up
x	X		down
y	X		
images	X		
upSpeed (entier>0)			
downSpeed (entier>0)			
visible			
scale (float)			

Les vitesses par défaut valent 1.

VerticalPositionSensor (La petite diode led rouge)

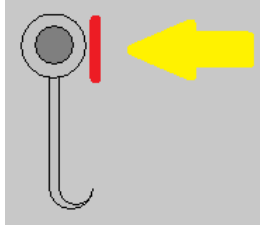
Mots clés	Obligatoire (X)	Provider - Terminals	Consumer - Terminals
name	X	contact	visible
type	X		
x	X		
y	X		
images	X		
target	X (le nom de l'objet à détecter)		
y1 % de la hauteur	(0.2 par défaut)		
y2 % de la hauteur	(0.8 par défaut)		
visible			
scale (float)			

L'objet "TerminalBoard" est utilisé pour lier :

3. Les sorties "contact" des interrupteurs aux entrées "left" et "right" du crochet.
4. La sortie "contact" de capteur à l'entrée "state" du gyrophare.

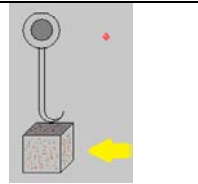
Le fichier json2.defs

```
{
  "actors": [
    {
      "name": "toUp", "type": "Button",
      "x": 10, "y": 10,
      "images": [ "switches\\icon_switch_red_off.png", "switches\\icon_switch_red_on.png" ],
      "interlock": true, "scale": 0.125
    },
    {
      "name": "toDown", "type": "Button",
      "x": 100, "y": 10,
      "images": [ "switches\\icon_switch_red_off.png", "switches\\icon_switch_red_on.png" ],
      "interlock": true,
      "scale": 0.125
    },
    {
      "name": "positionUp", "type": "VerticalPositionSensor",
      "x": 300, "y": 100,
      "images": [ "ledOff.png", "ledOn.png" ],
      "target": "hook", "y1": 0.05, "y2": 0.2,
    },
    {
      "name": "hook", "type": "VerticalMovingActor",
      "x": 200, "y": 200,
      "images": [ "hookBottom.png" ],
      "upSpeed": 1, "downSpeed": 1
    },
    {
      "name": "alarm", "type": "Indicator",
      "x": 10, "y": 120,
      "images": [ "imagesV\\icon_pharos_on_animation_4_clear.png",
        "imagesV\\icon_pharos_on_clear.png" ],
      "scale": 0.5
    }
  ],
  "terminalBoard": [
    {
      "provider": { "actor": "toUp", "terminal": "contact" },
      "consumer": { "actor": "hook", "terminal": "up" }
    },
    {
      "provider": { "actor": "toDown", "terminal": "contact" },
      "consumer": { "actor": "hook", "terminal": "down" }
    },
    {
      "provider": { "actor": "positionUp", "terminal": "contact" },
      "consumer": { "actor": "alarm", "terminal": "state" }
    }
  ]
}
```



L'objet du type "Follower"

Cet objet correspond par exemple à une pièce accrochée au crochet.

	<pre>{ "name": "box", "type": "Follower", "x": 0, "y": 0, "images": ["dirtyBox.png"], "driver" : "hook" }</pre>
---	---

Le "driver" est l'objet qui impose le mouvement à "box". Par défaut le bloc est sous le crochet.

Dernière version du fichier json2.defs

```
{
  "actors": [
    {
      "name": "toUp", "type": "Button",
      "x": 10, "y": 10,
      "images": [ "switches\\icon_switch_red_off.png", "switches\\icon_switch_red_on.png" ],
      "interlock": true, "scale": 0.125
    },
    {
      "name": "toDown", "type": "Button",
      "x": 100, "y": 10,
      "images": [ "switches\\icon_switch_red_off.png", "switches\\icon_switch_red_on.png" ],
      "interlock": true,
      "scale": 0.125
    },
    {
      "name": "positionUp", "type": "VerticalPositionSensor",
      "x": 300, "y": 100,
      "images": [ "ledOff.png", "ledOn.png" ],
      "target": "hook", "y1": 0.05, "y2": 0.2,
    },
    {
      "name": "hook", "type": "VerticalMovingActor",
      "x": 200, "y": 200,
      "images": [ "hookBottom.png" ],
      "upSpeed": 1, "downSpeed": 1
    },
    {
      "name": "alarm", "type": "Indicator",
      "x": 10, "y": 120,
      "images": [ "imagesV\\icon_pharos_on_animation_4_clear.png",
        "imagesV\\icon_pharos_on_clear.png" ],
      "scale": 0.5
    },
    {
      "name": "box", "type": "Follower",
      "x": 0, "y": 0,
      "images": [ "dirtyBox.png" ],
      "driver": "hook"
    }
  ],
  "terminalBoard": [
    {
      "provider": { "actor": "toUp", "terminal": "contact" },
      "consumer": { "actor": "hook", "terminal": "up" }
    },
    {
      "provider": { "actor": "toDown", "terminal": "contact" },
      "consumer": { "actor": "hook", "terminal": "down" }
    },
    {
      "provider": { "actor": "positionUp", "terminal": "contact" },
      "consumer": { "actor": "alarm", "terminal": "state" }
    }
  ]
}
```

Test des objets du type, Tank, Valve, LevelPositionSensor