

Manuel de l'utilisateur – 12 octobre 2020

Démarrage

Merci d'utiliser miranda, le logiciel de simulation universel.

Ce manuel donne les grandes lignes de l'utilisation de miranda, pour des explications plus détaillées sur certains points, consultez la chaîne Youtube miranda :

<https://www.youtube.com/watch?v=6g-dHlJrTP0&list=PLHIYmTo8fUg5V4-r3tcjzq-2wqwhRtZlo>

Vous pouvez utiliser miranda depuis un navigateur compatible WebGL ou installer un exécutable sous Windows. Pour ceci, rendez-vous sur le site Internet :

miranda.software

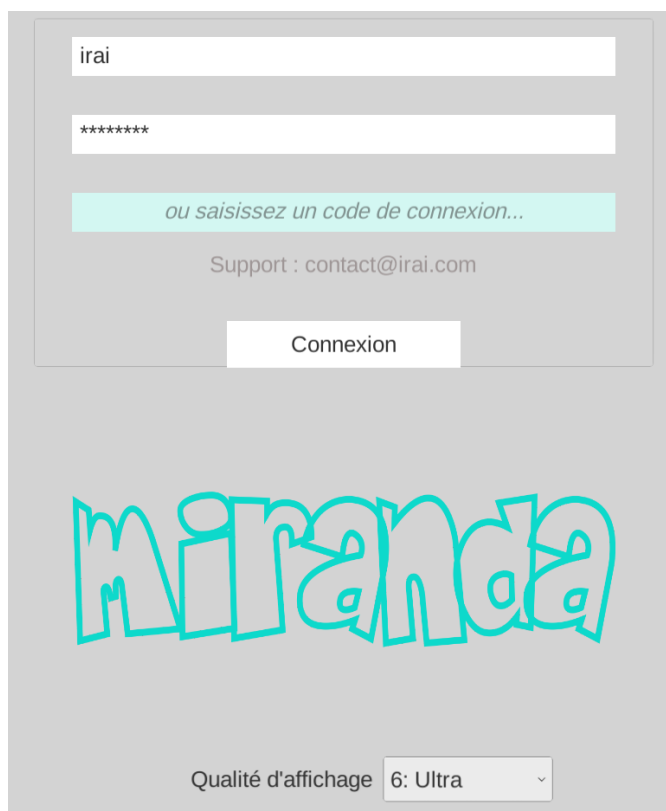
Et choisissez « Connection »...



Depuis l'écran d'accueil...



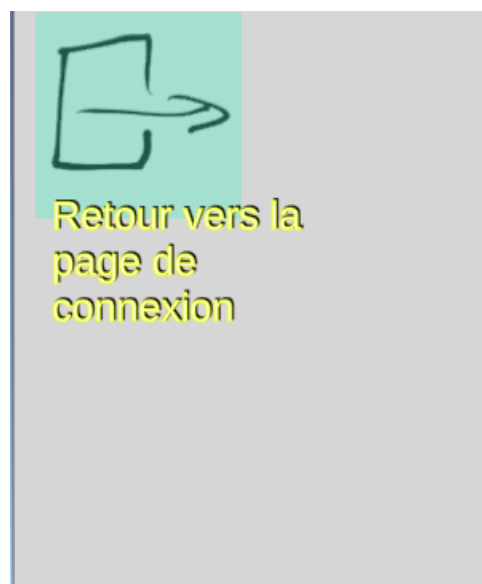
Choisissez si nécessaire la langue en cliquant en bas de l'écran, puis cliquez au milieu de l'écran...



Choisissez la qualité d'affichage, entrez votre identifiant et mot de passe (envoyés par votre fournisseur), puis cliquez sur « Connexion »...

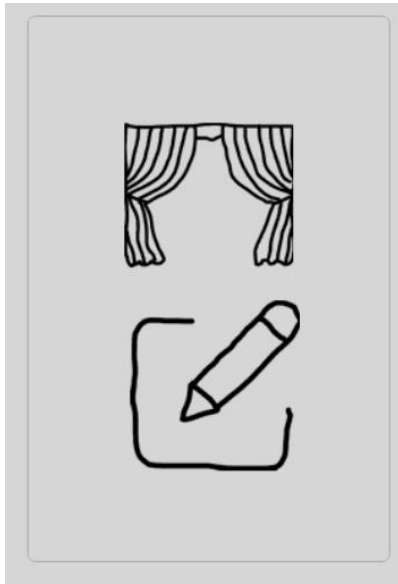


En survolant les différents éléments un texte explicatif est affiché...



Scènes

L'éditeur de scènes...



Navigation 3d

Avec la souris ::

- Tourner la molette = zoom,
- Bouton droit pressé + mouvements= rotation,
- Molette pressée + mouvements = translation.

Avec un écran tactile :

- Deux doigts= rotation,
- Trois doigts = translation.

Au clavier :

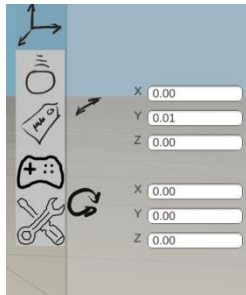
- Alt + flèches du clavier = déplacement.



En bas à gauche se trouve la bibliothèque d'objets classés par catégorie. Pour ajouter un objet à la scène, choisissez le dans cette bibliothèque.

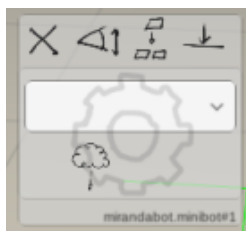


Lorsqu'un objet est sélectionné, ses propriétés sont affichées en haut à gauche sous la forme d'onglets :



Rappel : en survolant les différents éléments un texte explicatif est affiché.

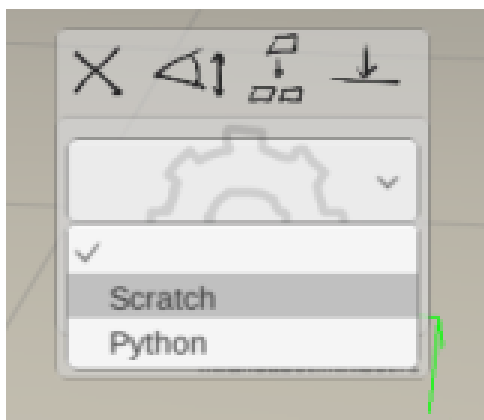
Une fenêtre...



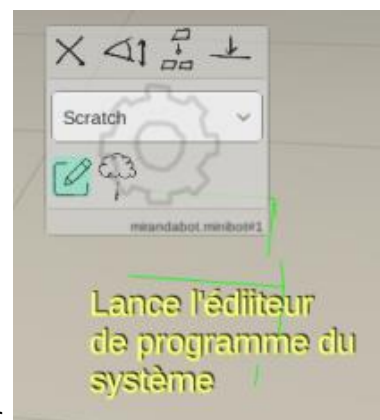
... est également affichée près des objets sélectionnés et permet de réaliser des opérations courantes : suppression, zoom, duplication, ...

Cette fenêtre permet également d'accéder au programme associé à chaque objet.

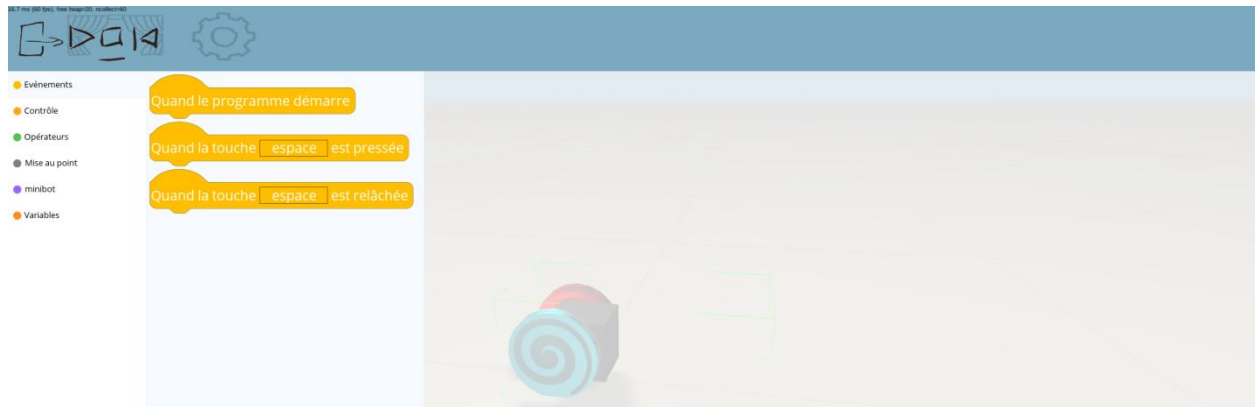
Par exemple, pour associer et éditer un programme en Scratch, cliquer sur :



puis sur



...



La position, rotation ou échelle des objets sélectionnés peuvent également être modifiées avec le « Gizmo » apparaissant à côté des objets...



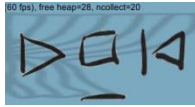
... en saisissant les flèches, l'objet se déplace, en saisissant les cercles, l'objet tourne, en saisissant le carré blanc au centre du Gizmo, la taille est modifiée.

L'icone Gizmo...



...en haut de la fenêtre de miranda permet de changer le type du Gizmo, passant par exemple d'un type de Gizmo universel à un Gizmo ne permettant que le changement de taille.

La simulation peut être lancée, stoppée et réinitialisée en utilisant ces icônes :



En mode simulation, si un système est sélectionné, l'onglet « Propriétés, E/S »...

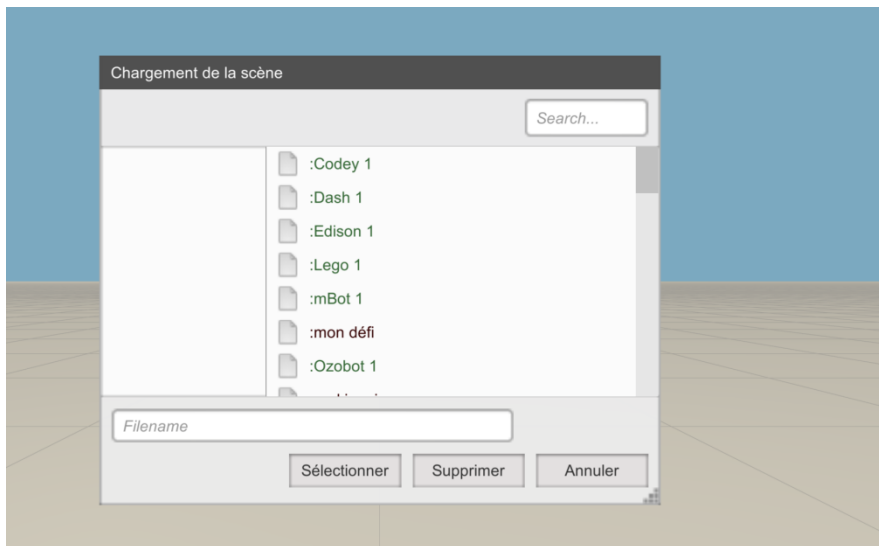


... permet d'accéder à la visualisation dynamique des éléments du système (moteurs, capteurs, ...). En cliquant sur ...



... il est possible de piloter manuellement les éléments en sortie (activer manuellement les moteurs par exemple).

Les scènes peuvent être sauvegardées et relues...



...les sauvegardes sont associées à votre compte client dans le cloud.

Partager une scène

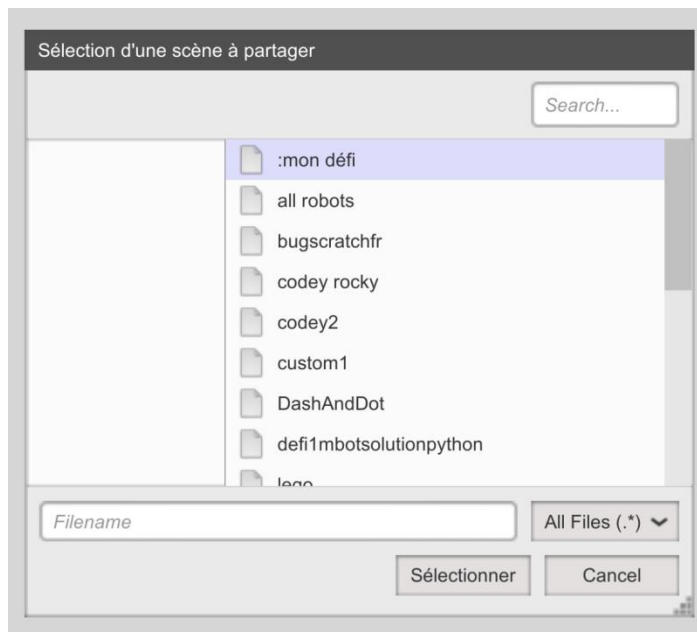
Le partage de scène...



... permet à un utilisateur de partager une scène (un défi par exemple) avec un autre utilisateur de miranda.

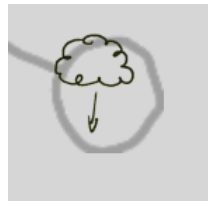


Pour envoyer un partage à un autre utilisateur...



Un même code peut être envoyé à plusieurs utilisateurs.

Pour recevoir un partage...



...

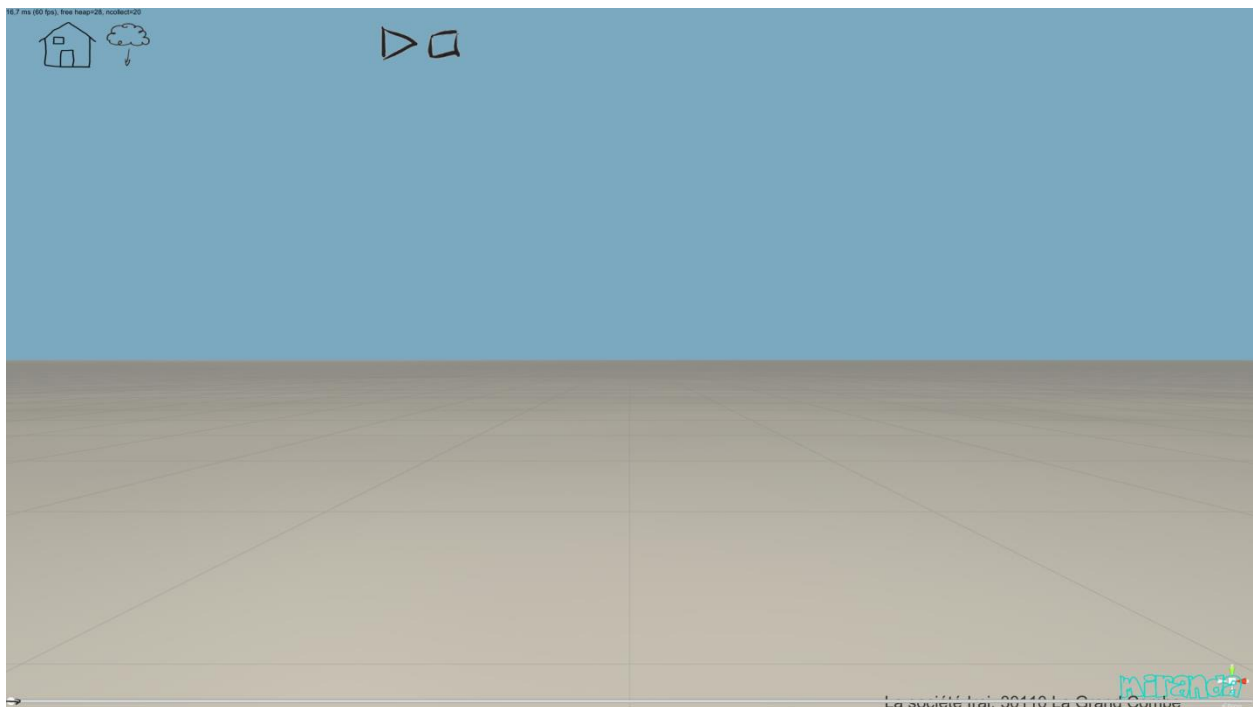


Mode player

Le mode player...



... est un mode limité à la simulation des scènes...



Ce mode est typiquement destiné à une utilisation des défis sans possibilité de les modifier.

Exemple de défi...

The screenshot displays the Miranda programming interface. On the left, a sidebar contains icons for Home, Cloud, Tools, and Python, along with a list of categories: Événements, Contrôle, Opérateurs, Mise au point, Robot, and Variables. The main area shows a challenge setup with two robots, one green and one red, positioned at a 'START' line. A blue arrow points to a green square on the track. The text 'Ecrivez un programme pour franchir les portes 1 à 3 avant le robot rouge puis pressez Play' is overlaid on the simulation. The bottom right corner features the Miranda logo and the text 'La société ital, 00110 La Grande Combe'.

20.2 ms (49 fps), free heap=29, ncollect=00

Quand le programme démarre

Quand la touche "espace" est pressée

Quand la touche "espace" est relâchée

Ecrivez un programme pour franchir les portes 1 à 3 avant le robot rouge puis pressez Play

Miranda

La société ital, 00110 La Grande Combe

Défis

Pour qu'une scène apparaisse comme un défi utilisable en mode player, son nom doit débuter par le caractère « : ».

Miranda permet d'utiliser les défis fournis avec le logiciel et aussi de créer ses propres défis.

L'objet « manager »...



... de la bibliothèque est la partie centrale d'un défi. Cet objet programmable en Scratch ou en Python permet de définir le déroulement du défi. L'objet « Ligne d'arrivée » peut également être utilisé pour vérifier la présence d'un objet à un certain endroit.

Un défi peut être composé de plusieurs étapes.

Liste des éléments de programmation de l'objet « Manager »...

A screenshot of the Manager object programming interface. On the left, there is a list of programming elements for the Manager object, each with a corresponding icon and a text input field. These elements include: "Nombre de systèmes démarrés dans" (with a slider), "Démarrer des systèmes dans" (with a button), "Stopper des systèmes dans" (with a button), "Sélectionner les systèmes dans" (with a button), "Cacher les systèmes dans" (with a button), "Montrer les systèmes dans" (with a button), "Nombre de systèmes dans" (with a slider), "entrés en contact avec" (with a button), "Définir le nombre d'étapes à" (with a text input field set to 1), "Définir le niveau" (with a text input field set to 1) "comme terminé", and "Définir le défi comme perdu et afficher" (with a button). On the right, there is a code editor window titled "#Script Python associé au manager de mode player". It contains the following Python code:

```
# manager.startedsystems("")
# manager.startsystems("")
# manager.stopsystems("")
# manager.selectsystems("")
# manager.showsystems("")
# manager.hidesystems("")
# manager.numberofobjectscomeintocontact("", "myobject")
# manager.setstepnumbers(1)
# manager.endstep(1)

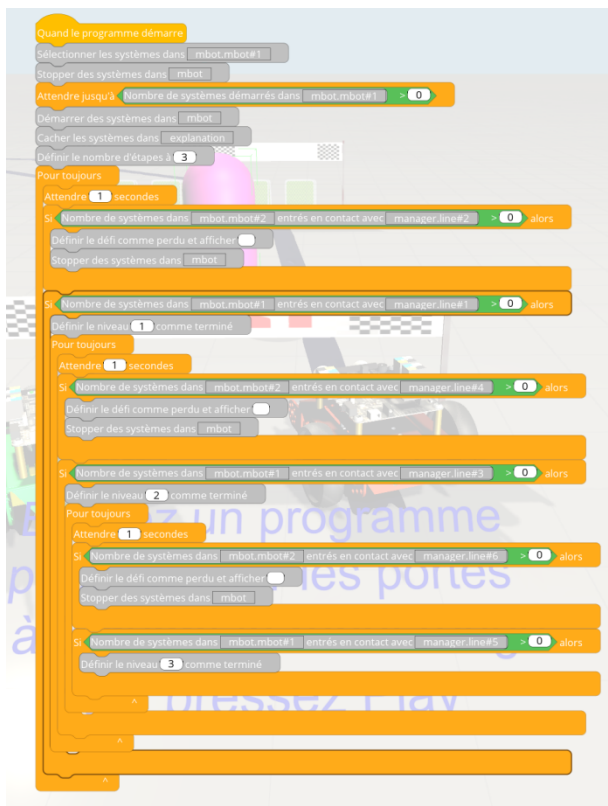
import manager
```

Exemple d'utilisation de l'objet Manager programmé en Python (défi 1 mBot, pour ce défi, l'objet mbot.mbot#1 est le robot vert programmé par l'utilisateur du défi, mbot.mbot#2 est celui de couleur rouge piloté par l'ordinateur) :

```
import time
import manager

manager.selectsystems('mbot.mbot#1') Sélectionne le robot vert
manager.stopsystems('mbot') Stoppe tous les robots mBot
while manager.startedsystems('mbot.mbot#1')==0: Attend que l'utilisateur démarre son robot
    continue
manager.startsystems('mbot') Démarre tous les robots (donc aussi le robot piloté par l'ordinateur)
manager.hidesystems('explanation') Cache les explications
manager.setstepnumbers(3) Le défi comporte 3 étapes
while True: La boucle de surveillance du défi pour l'étape 1
    time.sleep(1) On surveille toutes les secondes seulement
    if manager.numberofobjectscomeintocontact('mbot.mbot#2',manager.line#2)>0: Le robot rouge a passé la ligne d'arrivée de l'étape 1 ?
        manager.abort('') Perdu : arrêter les robots
        manager.stopsystems('mbot')
    if manager.numberofobjectscomeintocontact('mbot.mbot#1',manager.line#1)>0: Le robot vert a passé la ligne d'arrivée de l'étape 1 ?
        manager.endstep(1) Etape 1 terminée, puis à suivre surveillance pour l'étape 2 puis 3
        while True:
            time.sleep(1)
            if manager.numberofobjectscomeintocontact('mbot.mbot#2',manager.line#4)>0:
                manager.abort('')
                manager.stopsystems('mbot')
            if manager.numberofobjectscomeintocontact('mbot.mbot#1',manager.line#3)>0:
                manager.endstep(2)
                while True:
                    time.sleep(1)
                    if manager.numberofobjectscomeintocontact('mbot.mbot#2',manager.line#6)>0:
                        manager.abort('')
                        manager.stopsystems('mbot')
                    if manager.numberofobjectscomeintocontact('mbot.mbot#1',manager.line#5)>0:
                        manager.endstep(3)
```

Même exemple en Scratch :

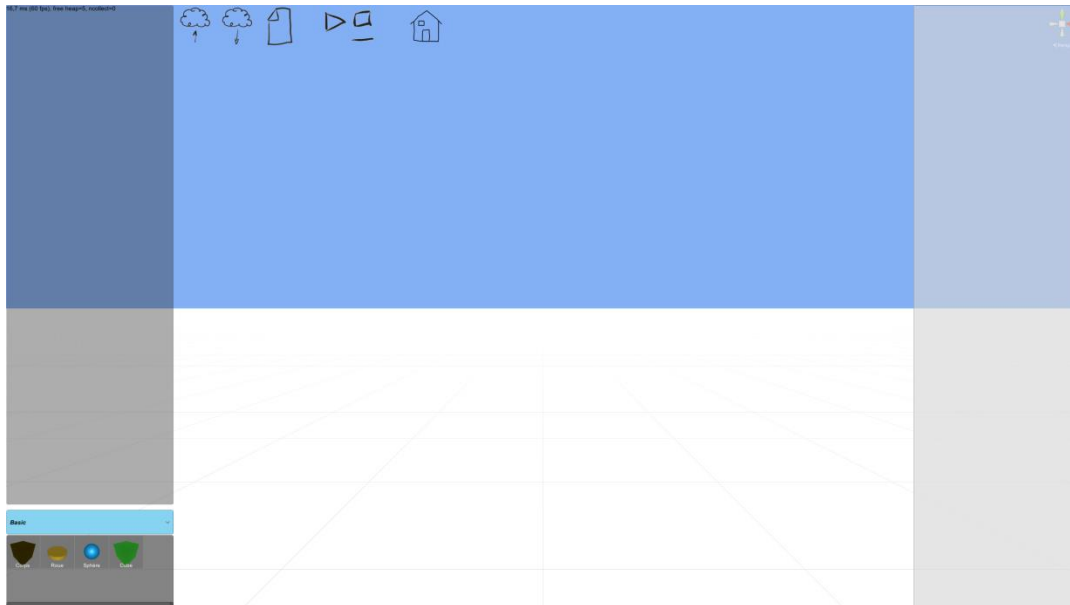


Editeur de systèmes

L'éditeur de systèmes...



... permet de créer ses propres systèmes (robots par exemple) utilisables dans les scènes de simulation.



Les systèmes sont créés en définissant la structure physique, par exemple le corps d'un robot puis les roues.

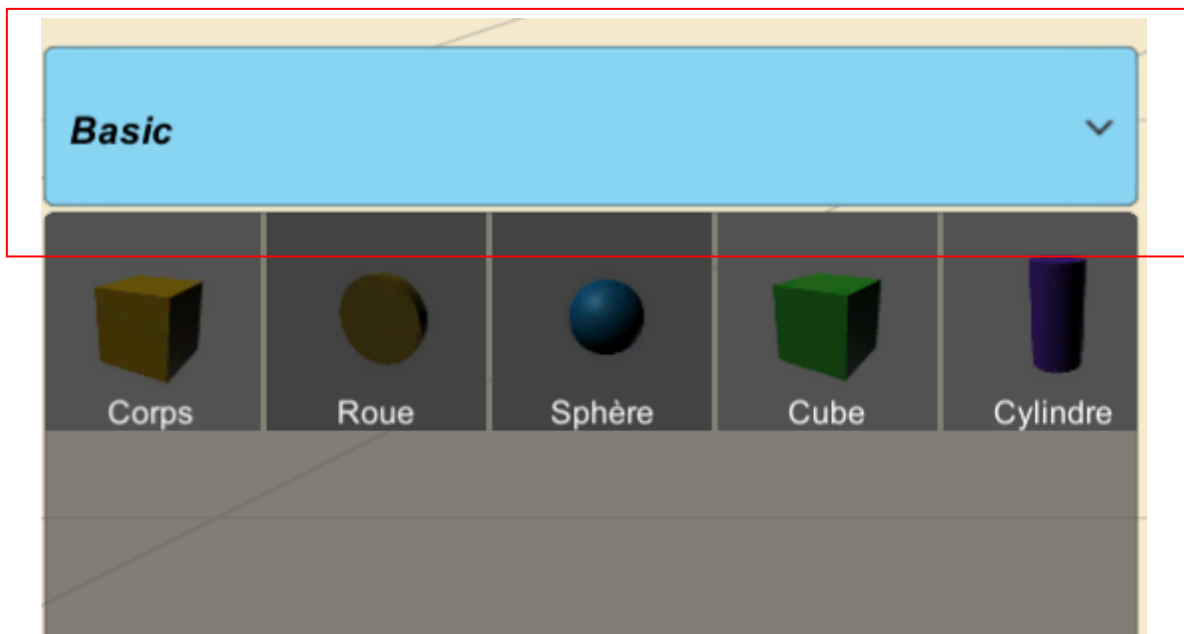
D'autres éléments peuvent aussi être ajoutés : capteurs, leds.

Des géométries personnalisées peuvent être ajoutées comme décors pour finaliser l'aspect visuel du robot.

Enfin, l'interface de programmation Scratch d'un système peut être personnalisée.

Certains des robots de la bibliothèque de miranda (Edison et Thymio par exemple) ont été créés avec l'éditeur de systèmes et peuvent être réouverts dans l'éditeur pour observer leur structure interne et servir de modèle pour vos propres créations.

Les éléments permettant la création des systèmes sont répartis en catégories...



... un clic sur un élément l'ajoute comme enfant de l'élément sélectionné.

Les catégories se présentent ainsi :

- Basic : corps d'un système et formes simples,
- Géométries personnalisées : pour importer vos propres géométries depuis un fichier 3d au format « .glb »,
- Capteurs : les différents capteurs,
- Leds : les éléments lumineux,
- Liaisons : les liaisons physiques,
- Sons : pour émettre des sons,
- Lumières : pour émettre de la lumière,

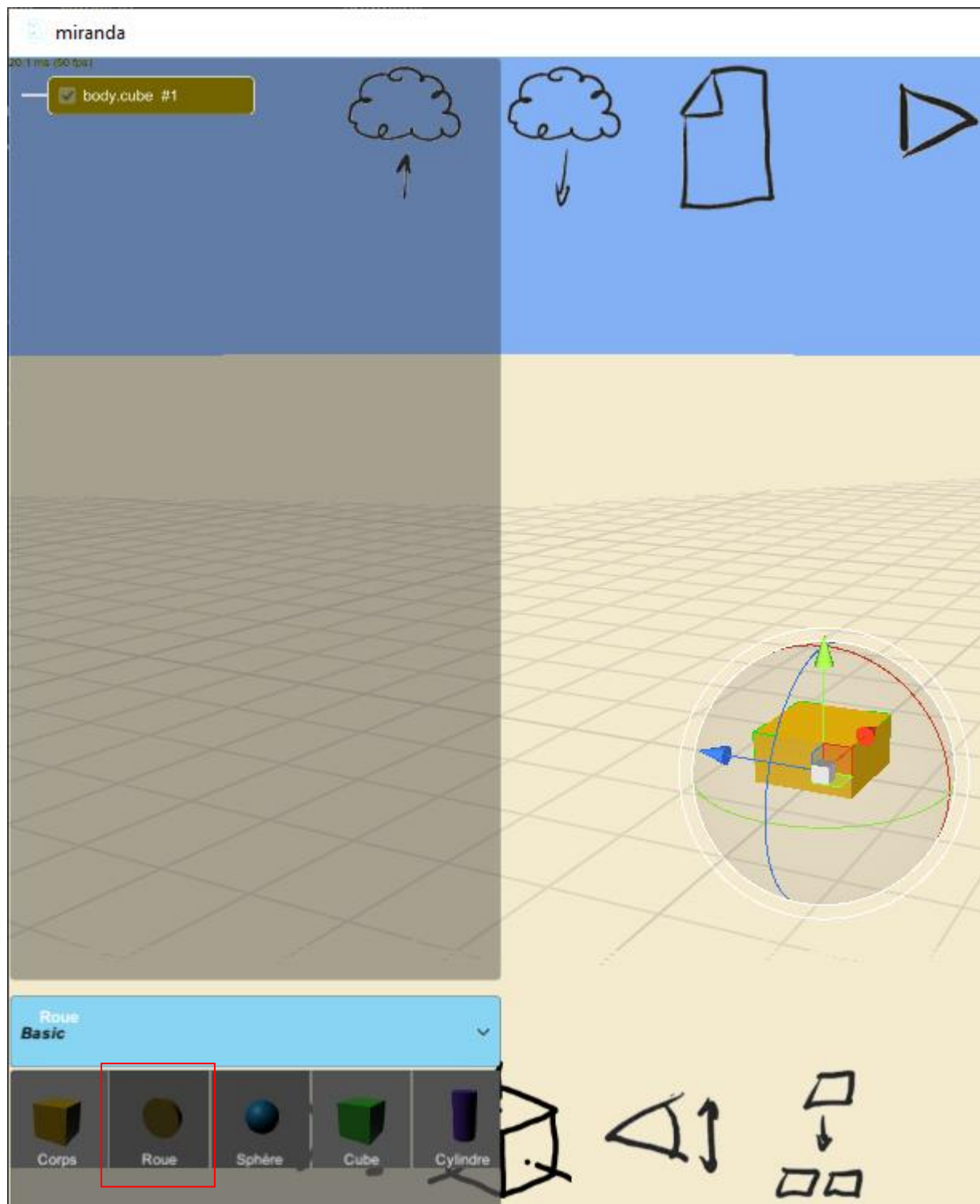
- Autre : tout ce qui ne rentre pas dans les autres catégories.

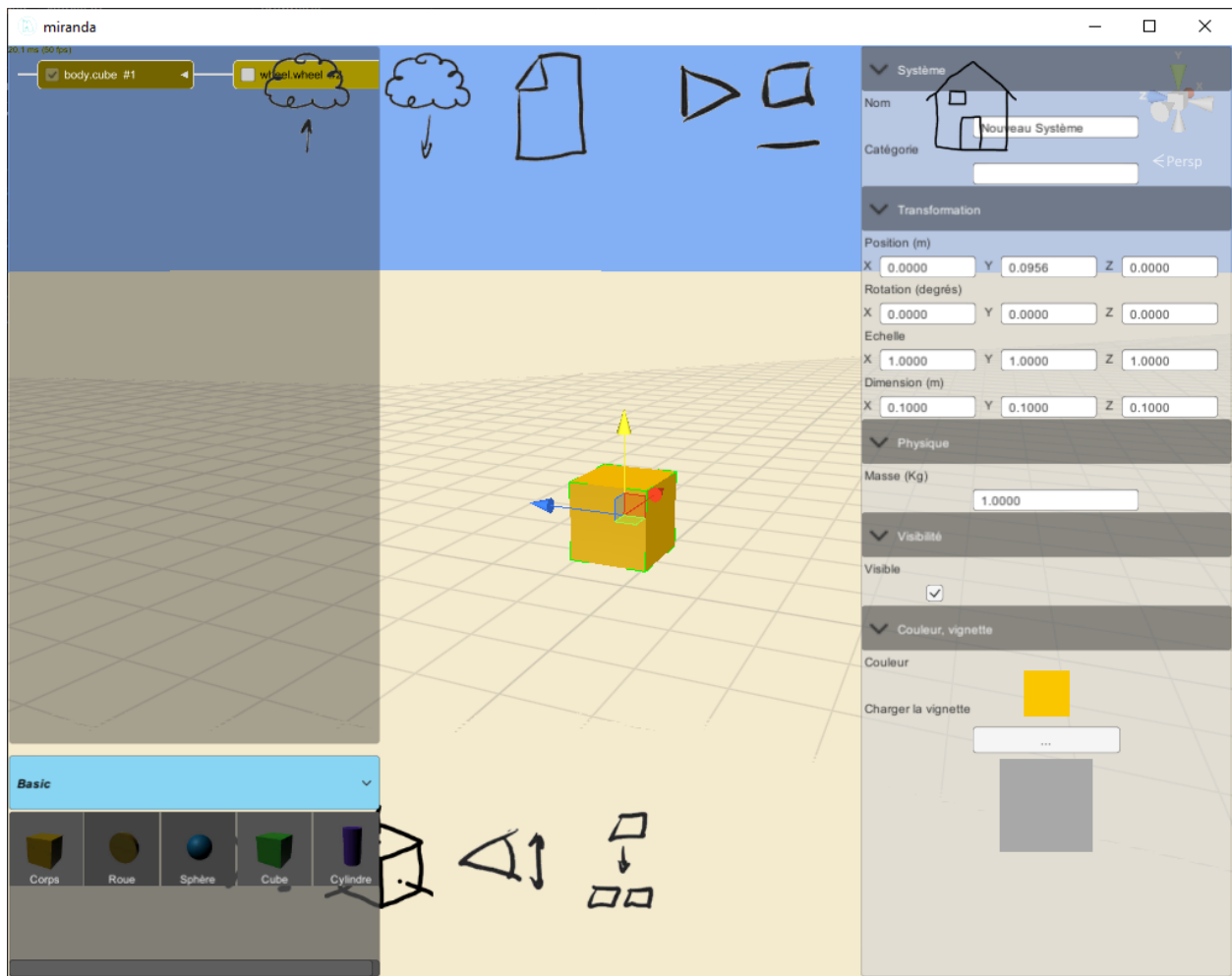
Ce qui suit illustre la création d'un système.

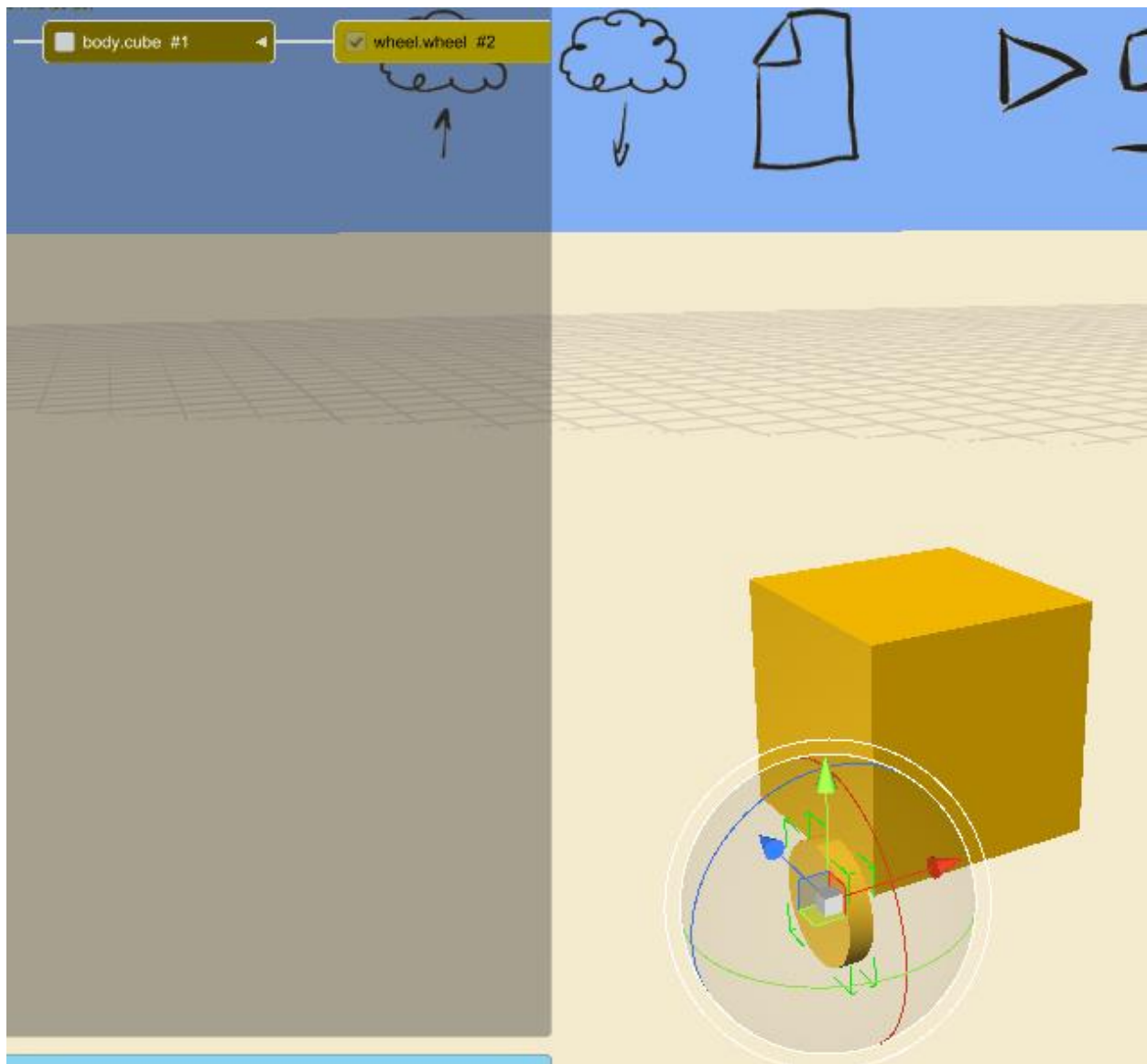
Les différentes étapes ne sont là que pour exemple, certaines d'entre elles sont facultatives et à utiliser uniquement en fonction de vos besoins.

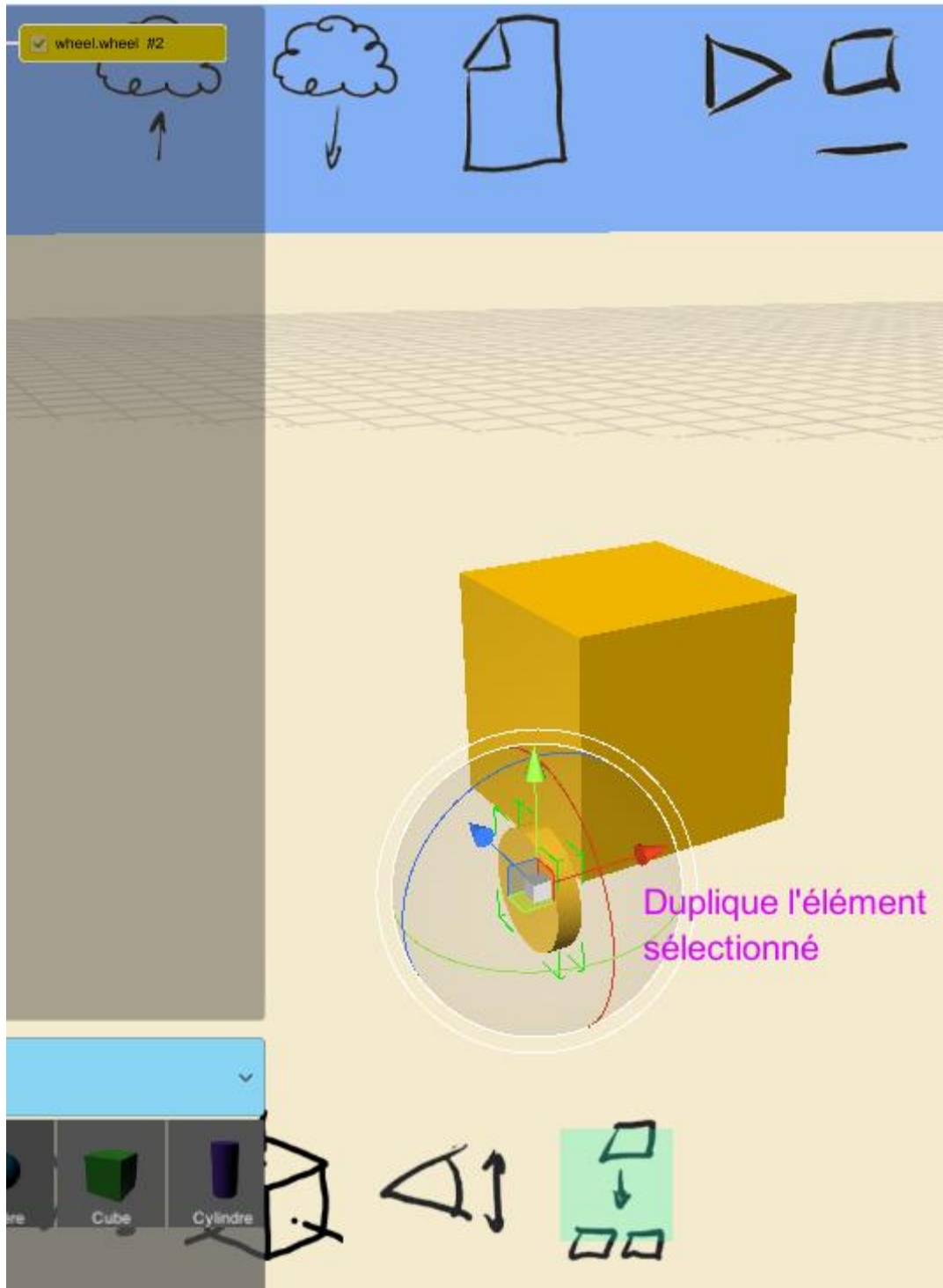
Création d'un robot minimaliste...

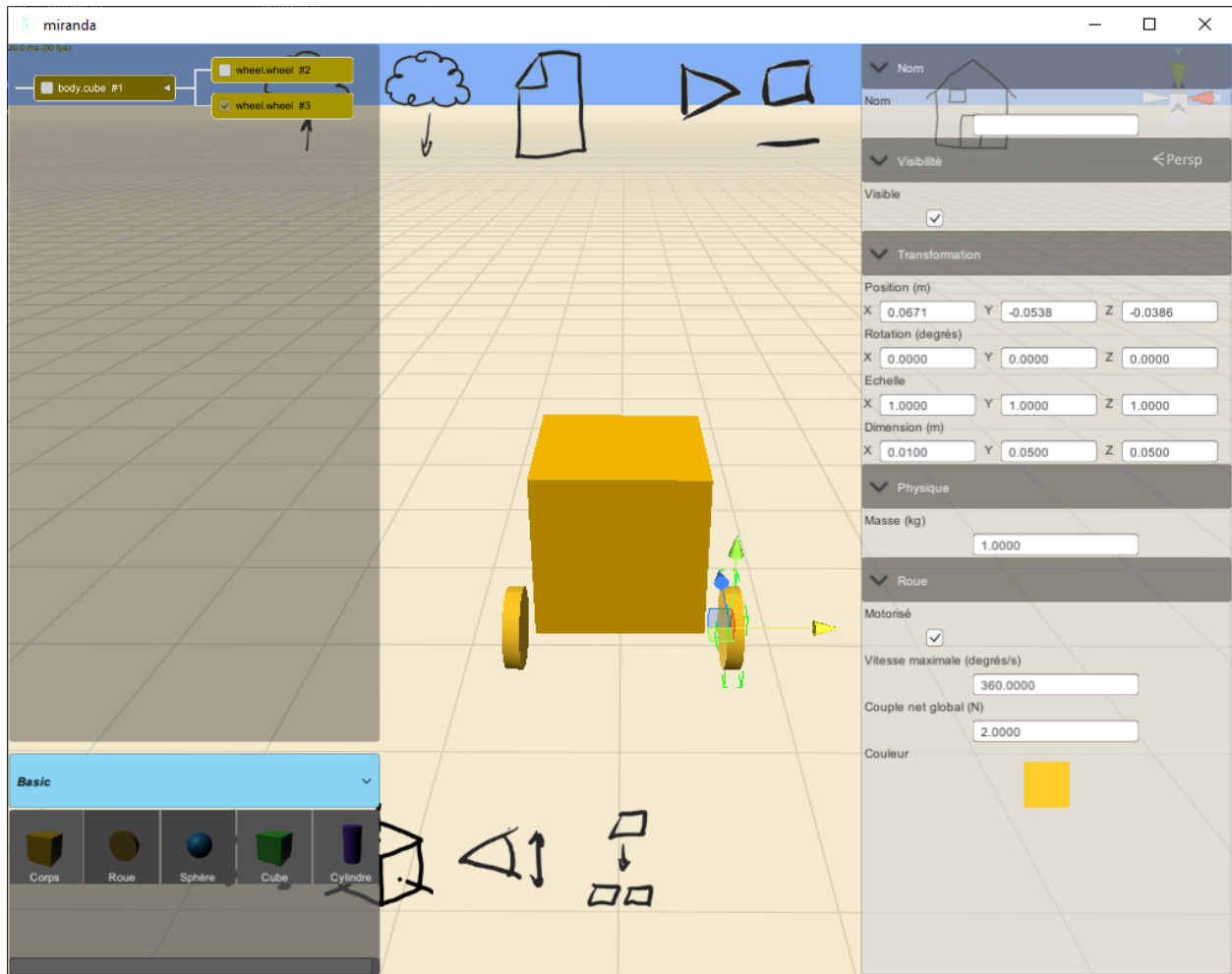


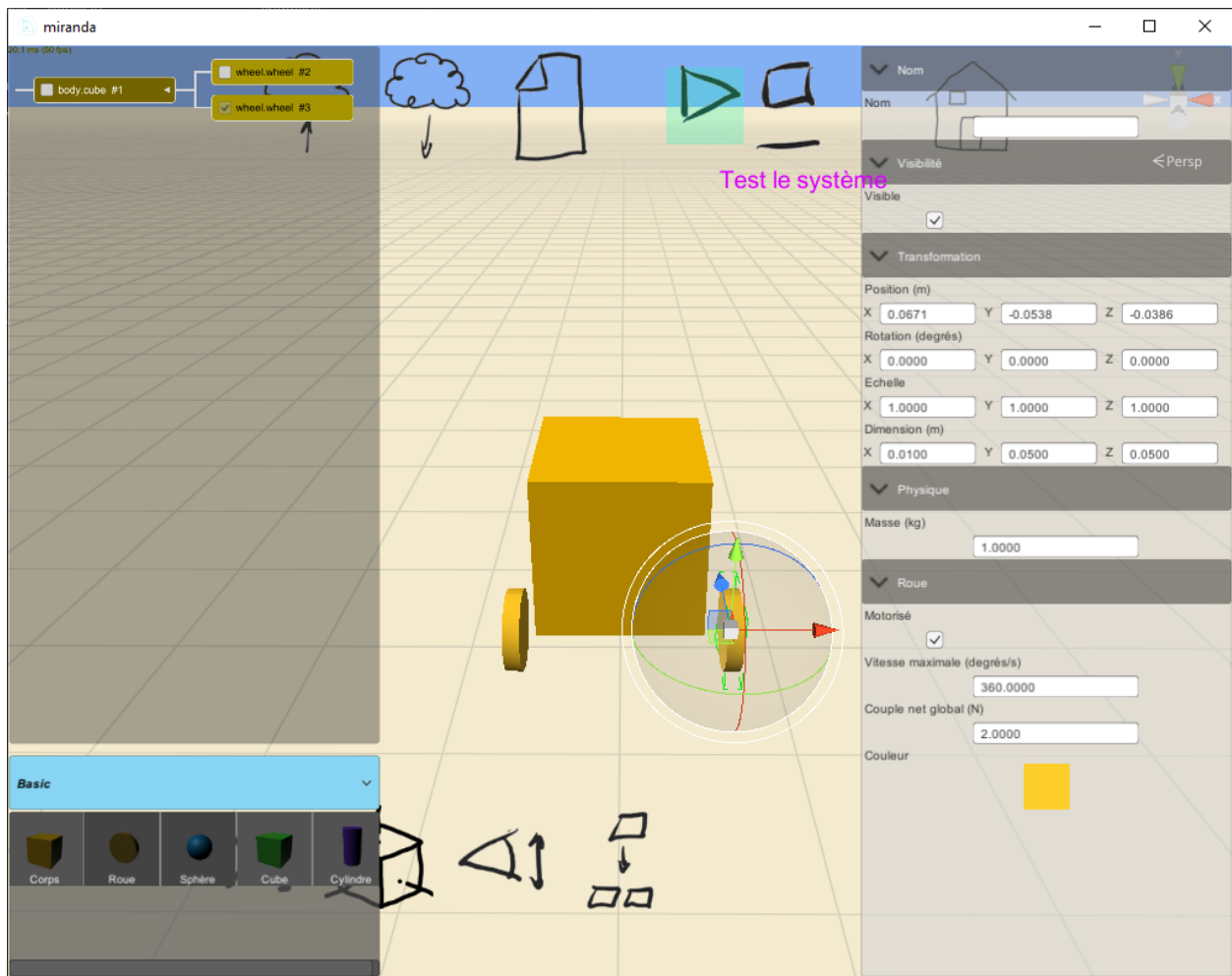






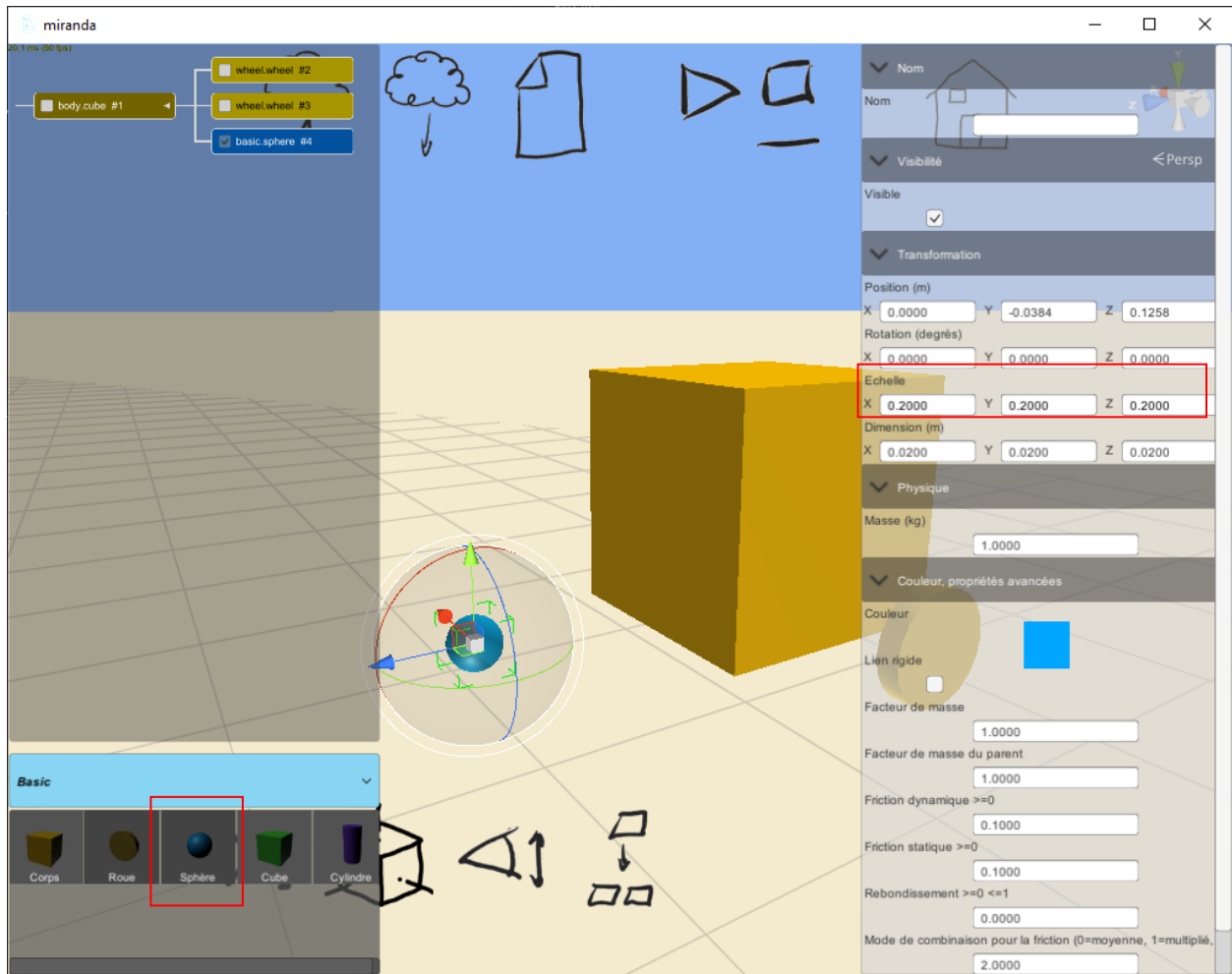


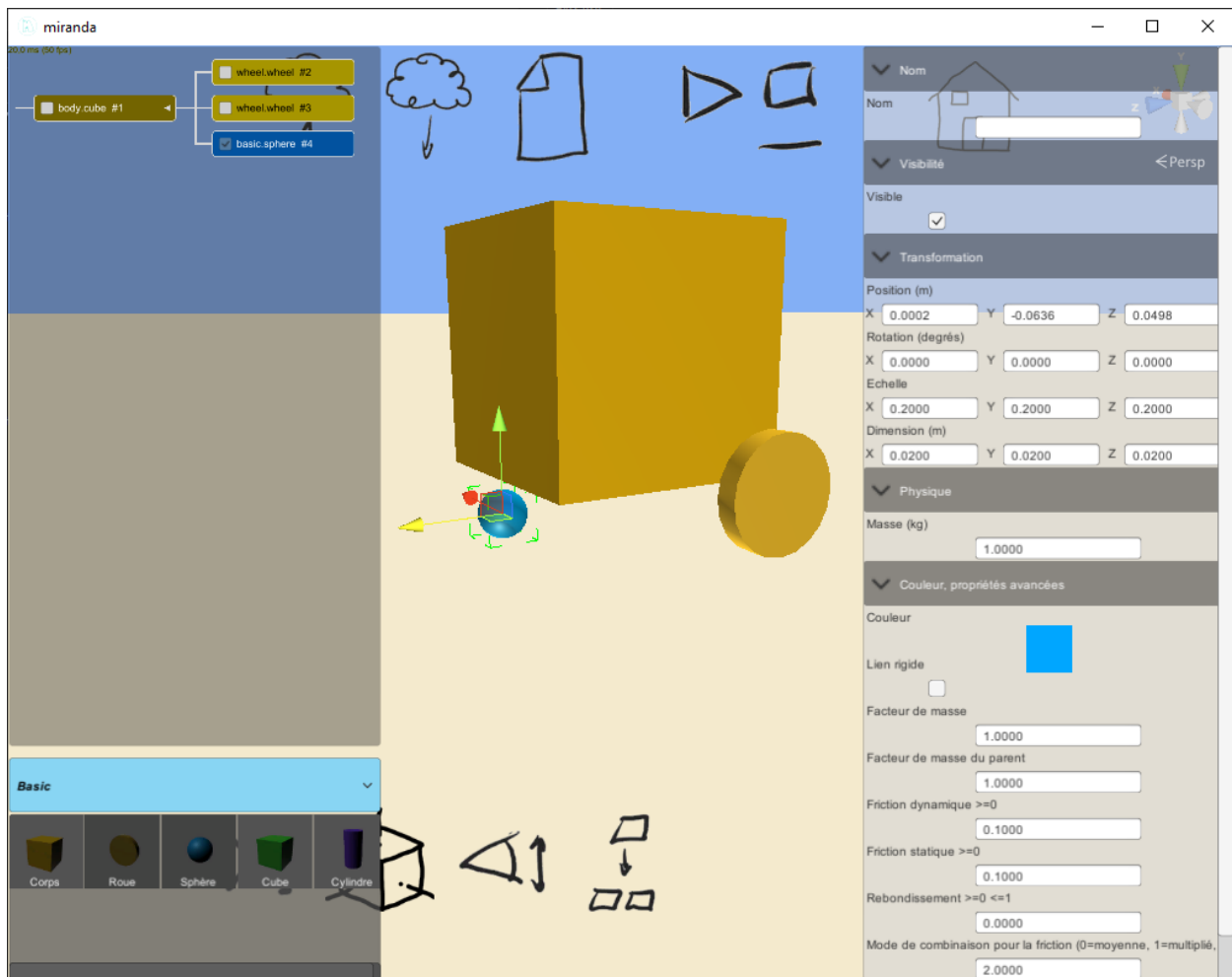




... par défaut, les roues sont motorisées, le système est donc fonctionnel et le robot avance.

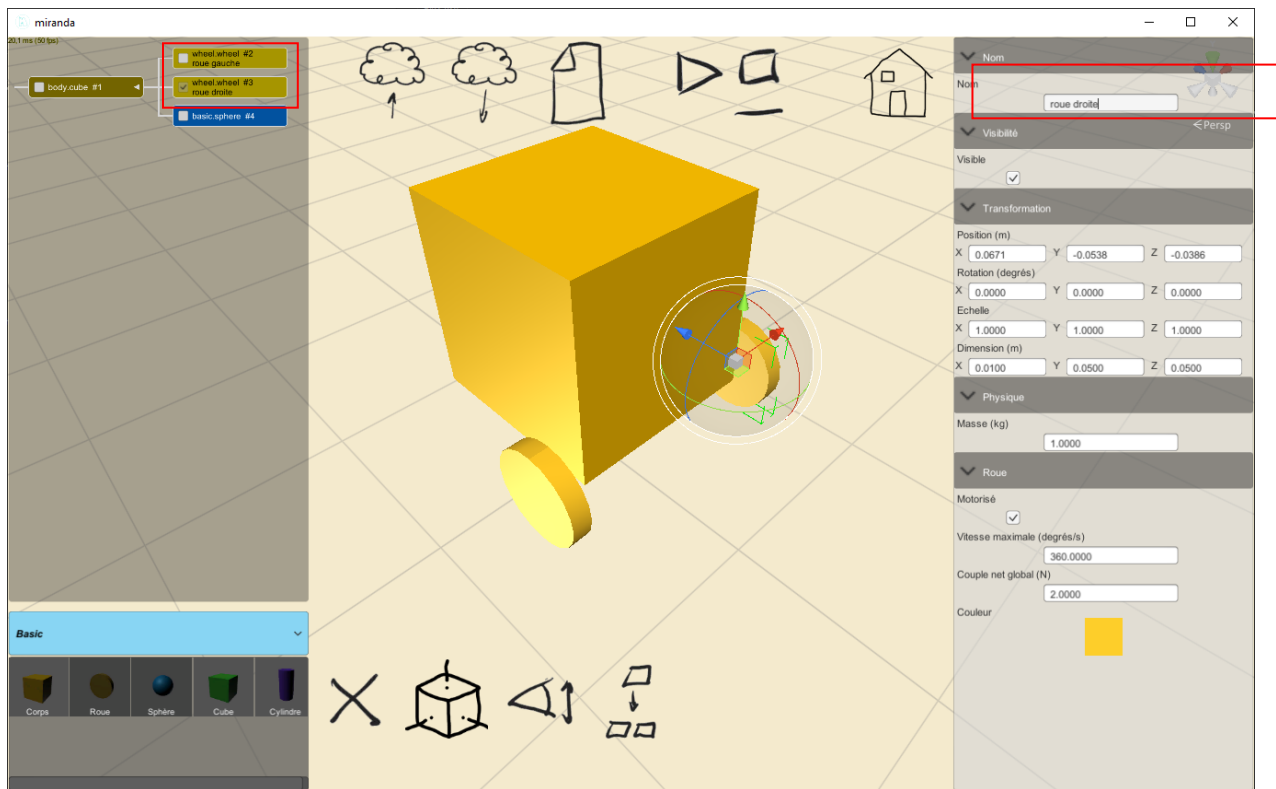
Enrichissons notre modèle, ajoutons une sphère comme troisième point d'appui...





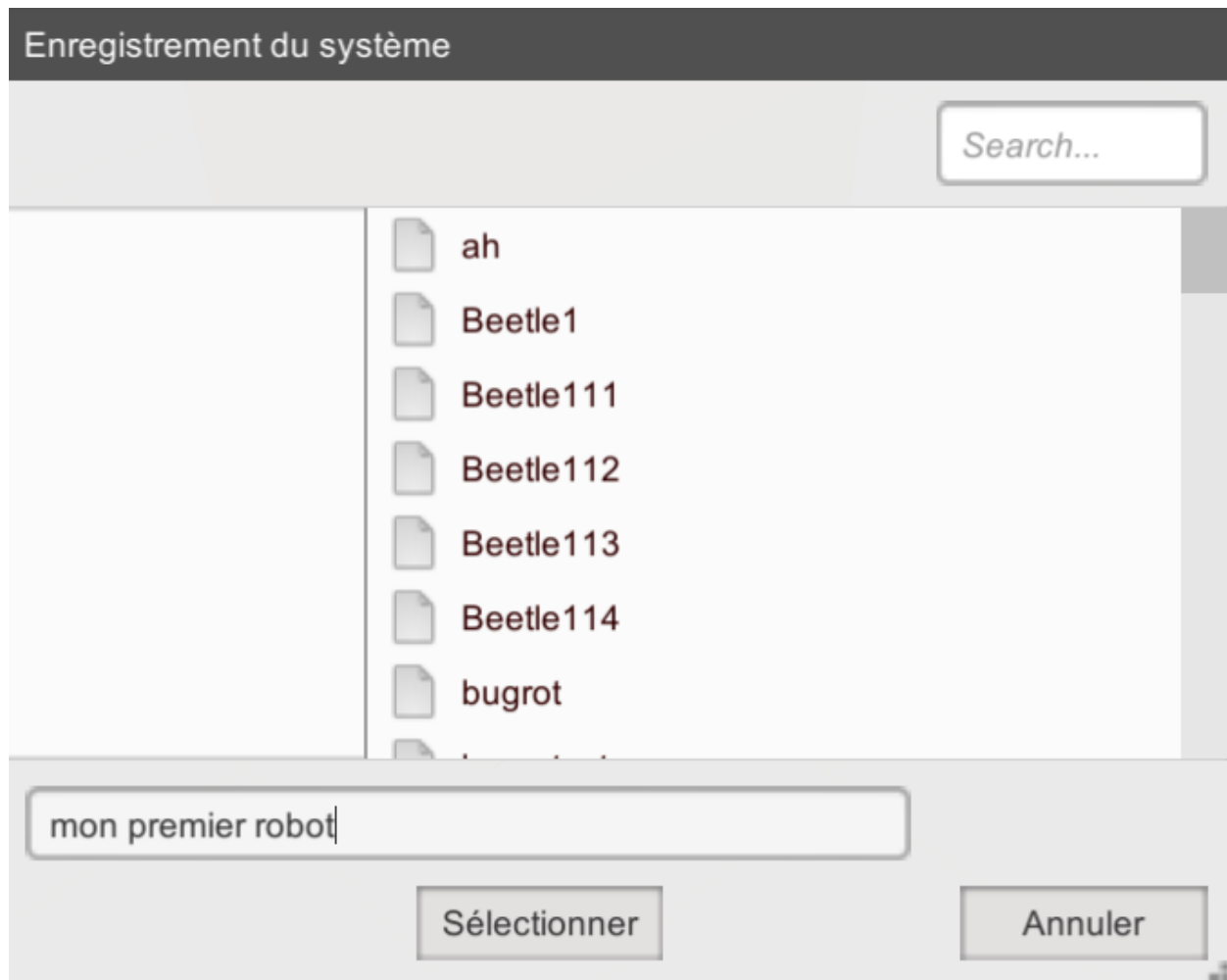
... par défaut, les éléments possèdent automatiquement une liaison physique avec leur parent.

Voyons comment utiliser notre système dans l'éditeur de scène. Pour commencer, nommons les éléments afin de pouvoir les identifier quand il s'agira de les piloter...



... puis sauvegardons notre modèle...

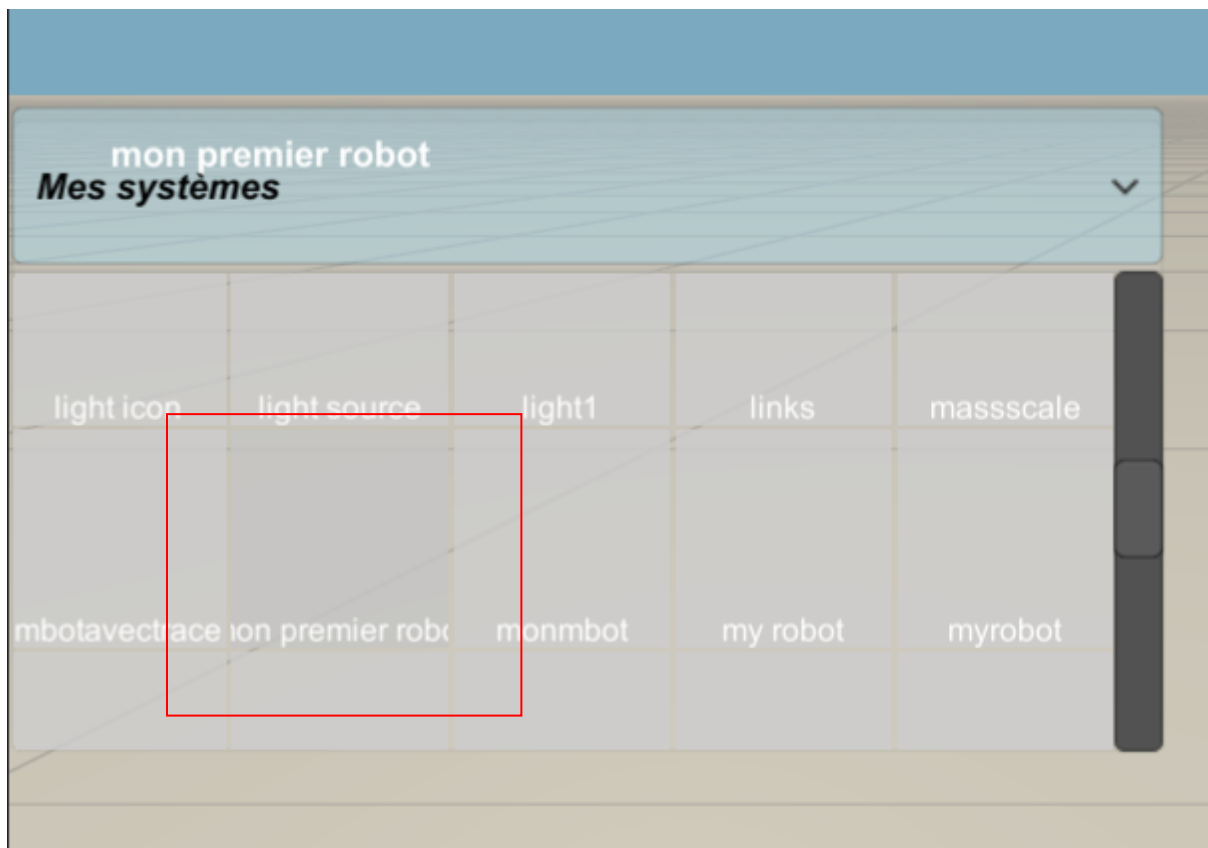


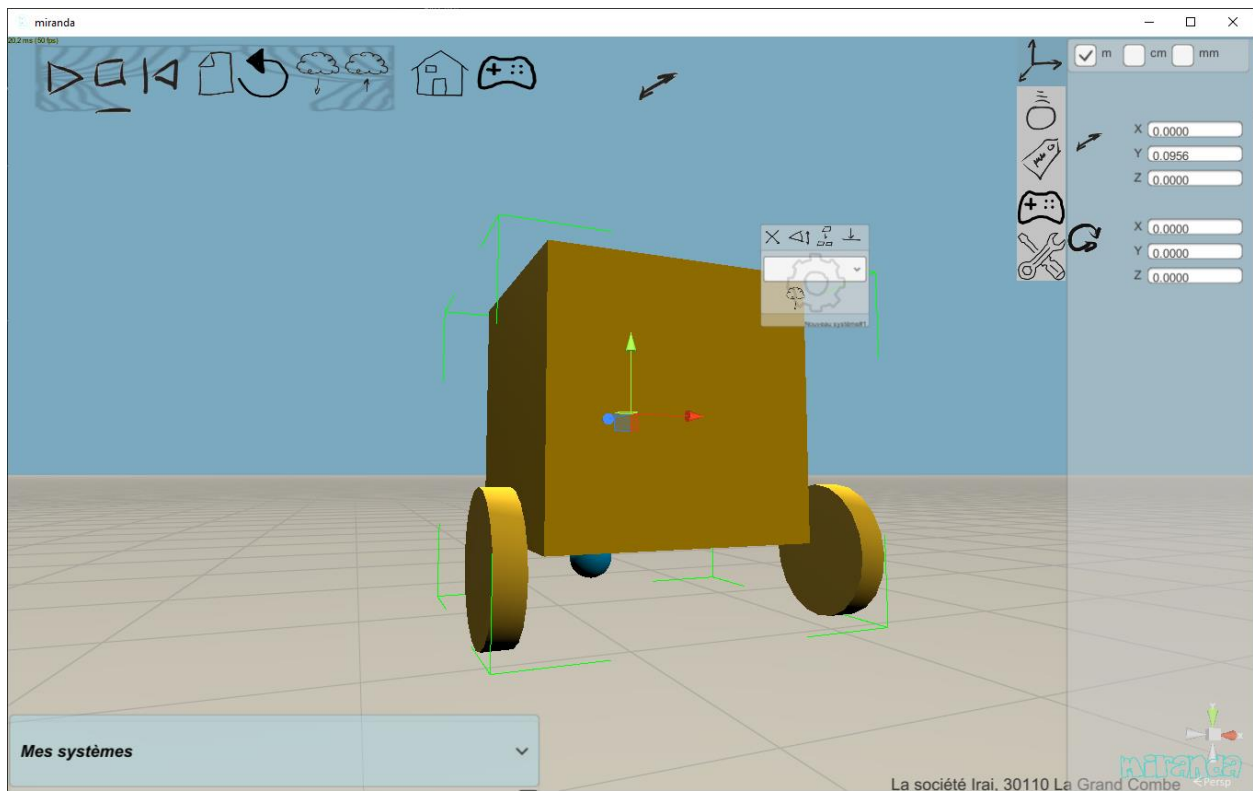


... maintenant, depuis l'éditeur de scènes...



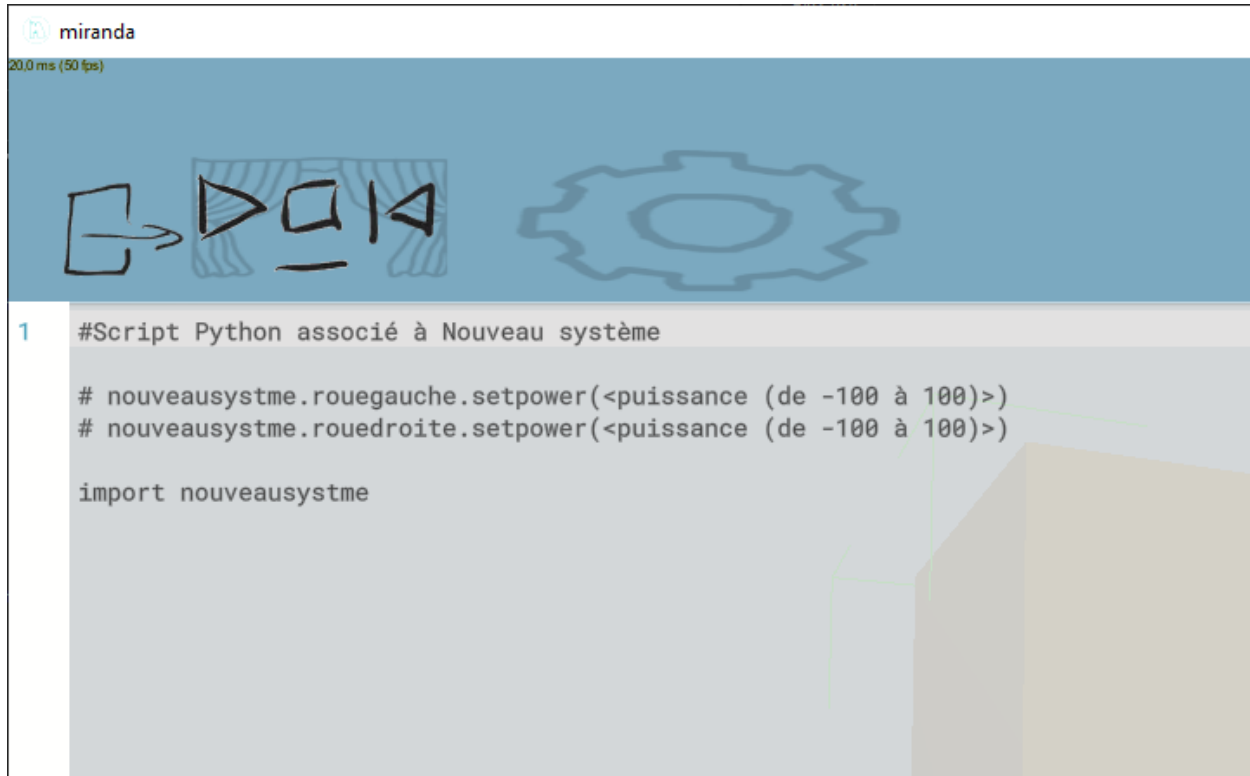
... et dans la catégorie « Mes systèmes »...



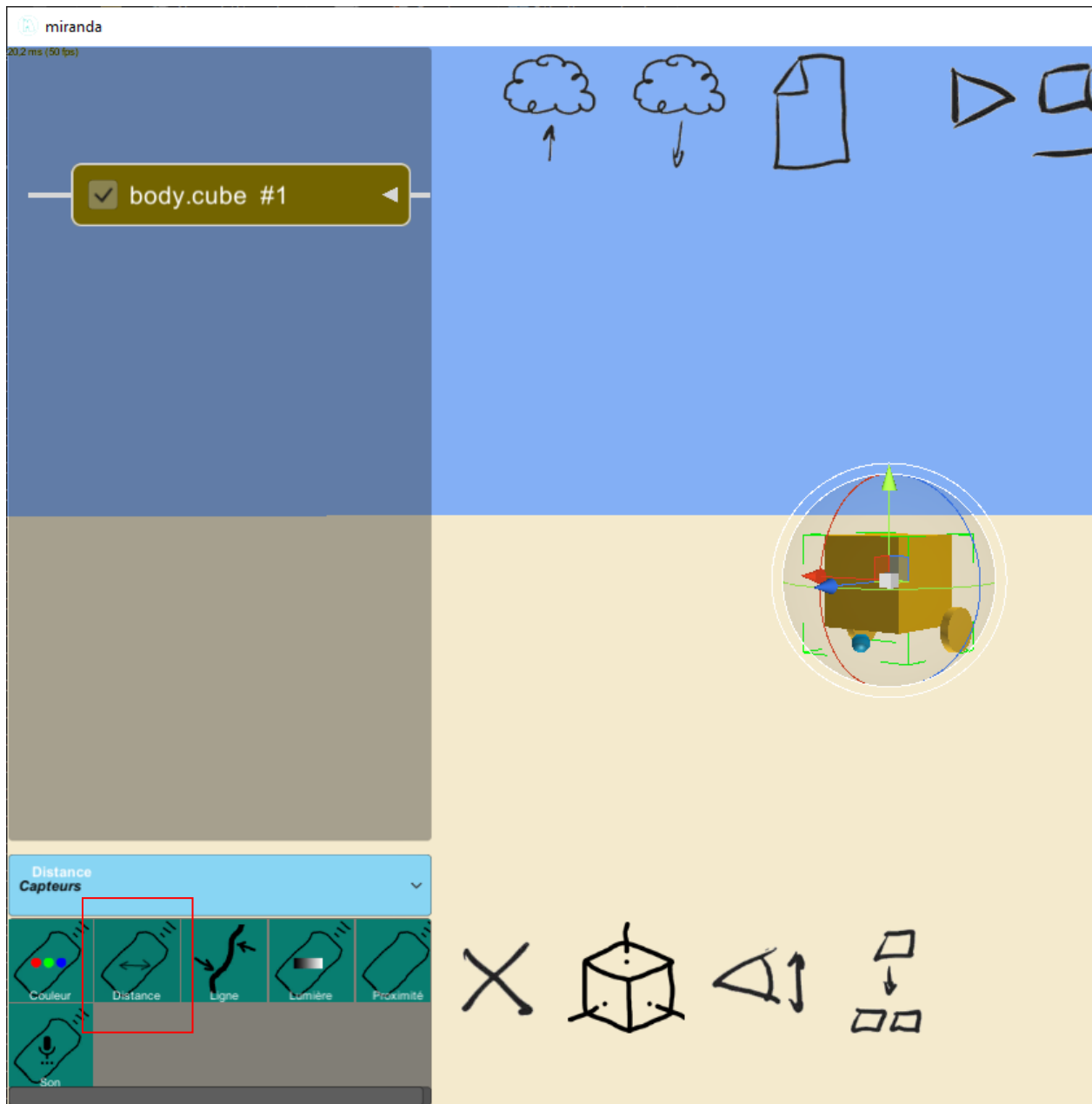


... les éléments actifs de notre robot, pour nous ici, seulement les roues sont accessibles en Scratch ou en Python...

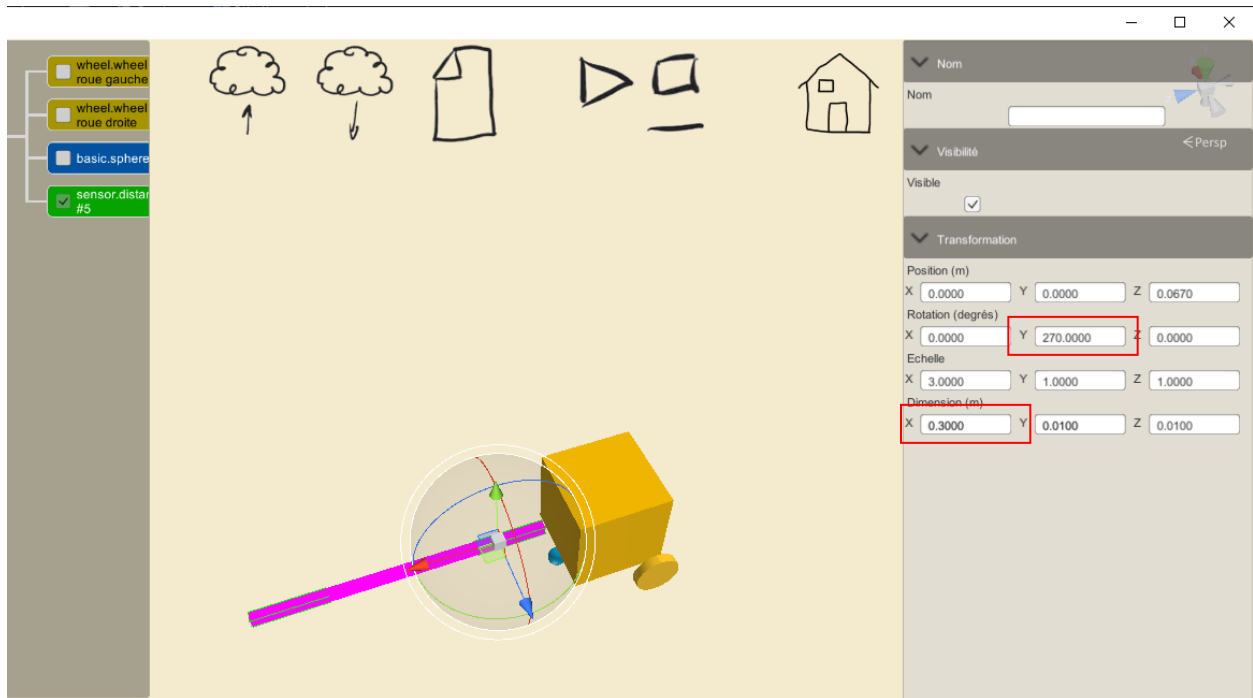




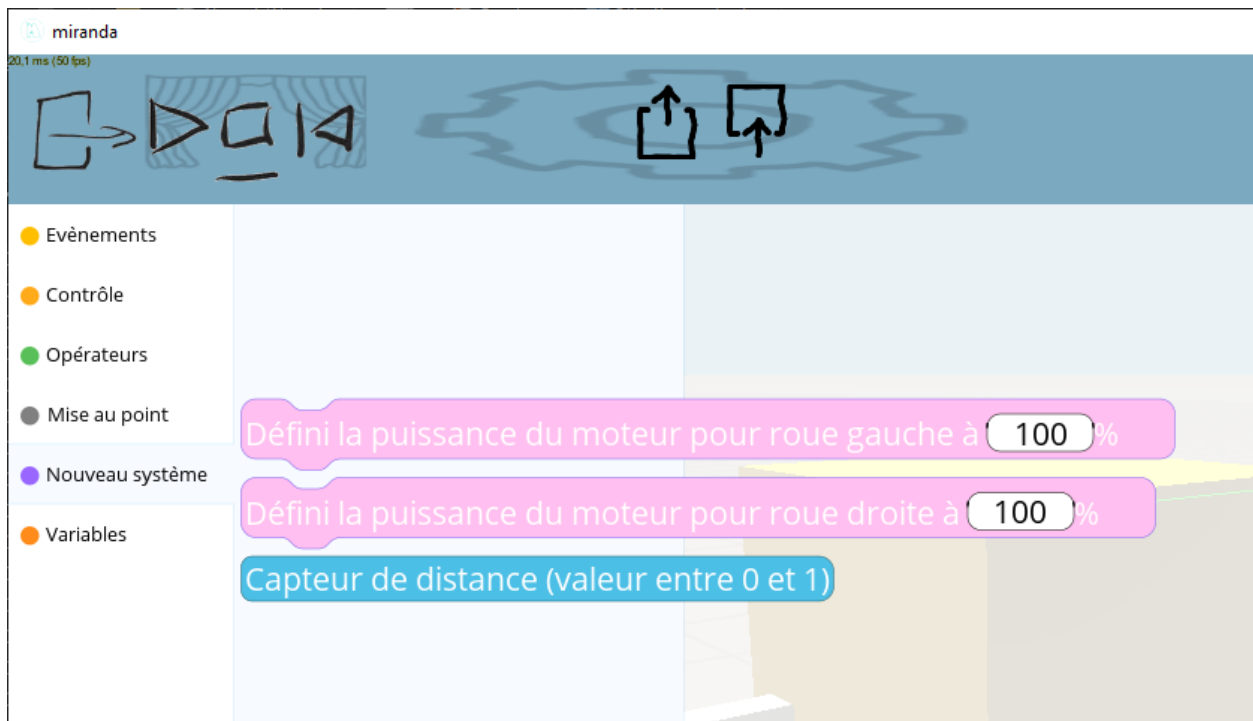
... retournons compléter notre modèle en ajoutant un capteur de distance...

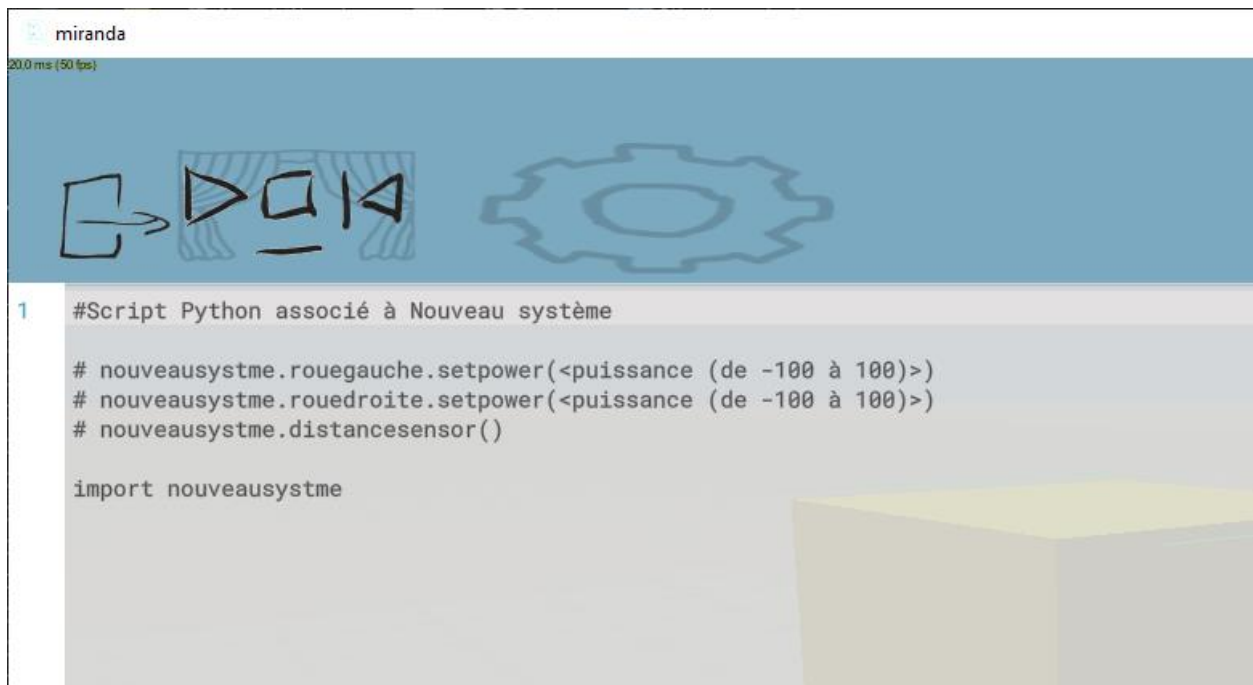


... admettons que nous souhaitons obtenir un capteur de distance opérant à une distance maximum de 30 cm à l'avant du robot ...



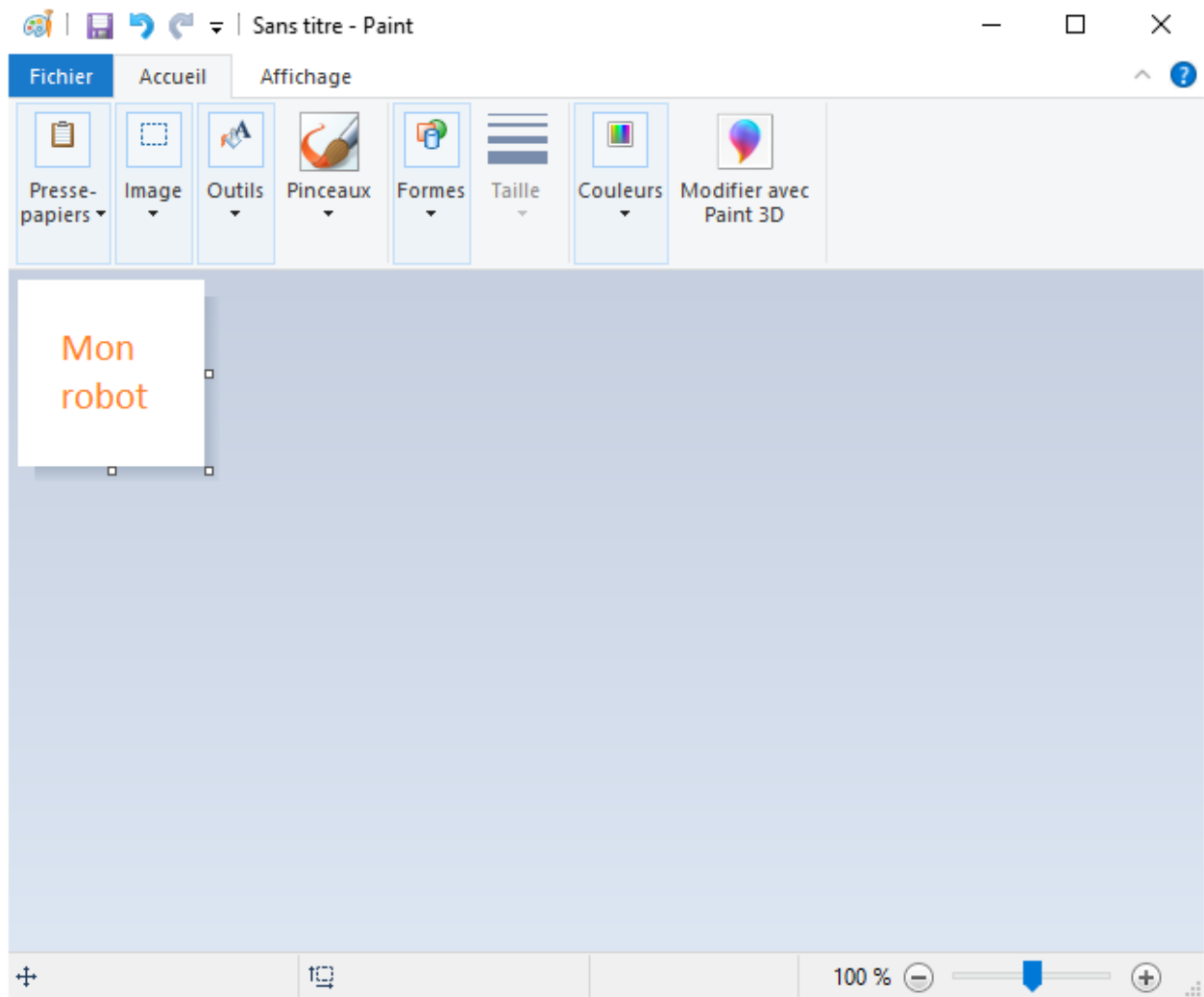
... La case à cocher « Visible » permet de masquer la géométrie associée à la zone de détection. Des blocs Scratch et fonctions Python « par défaut » sont automatiquement accessibles en mode programmation ...



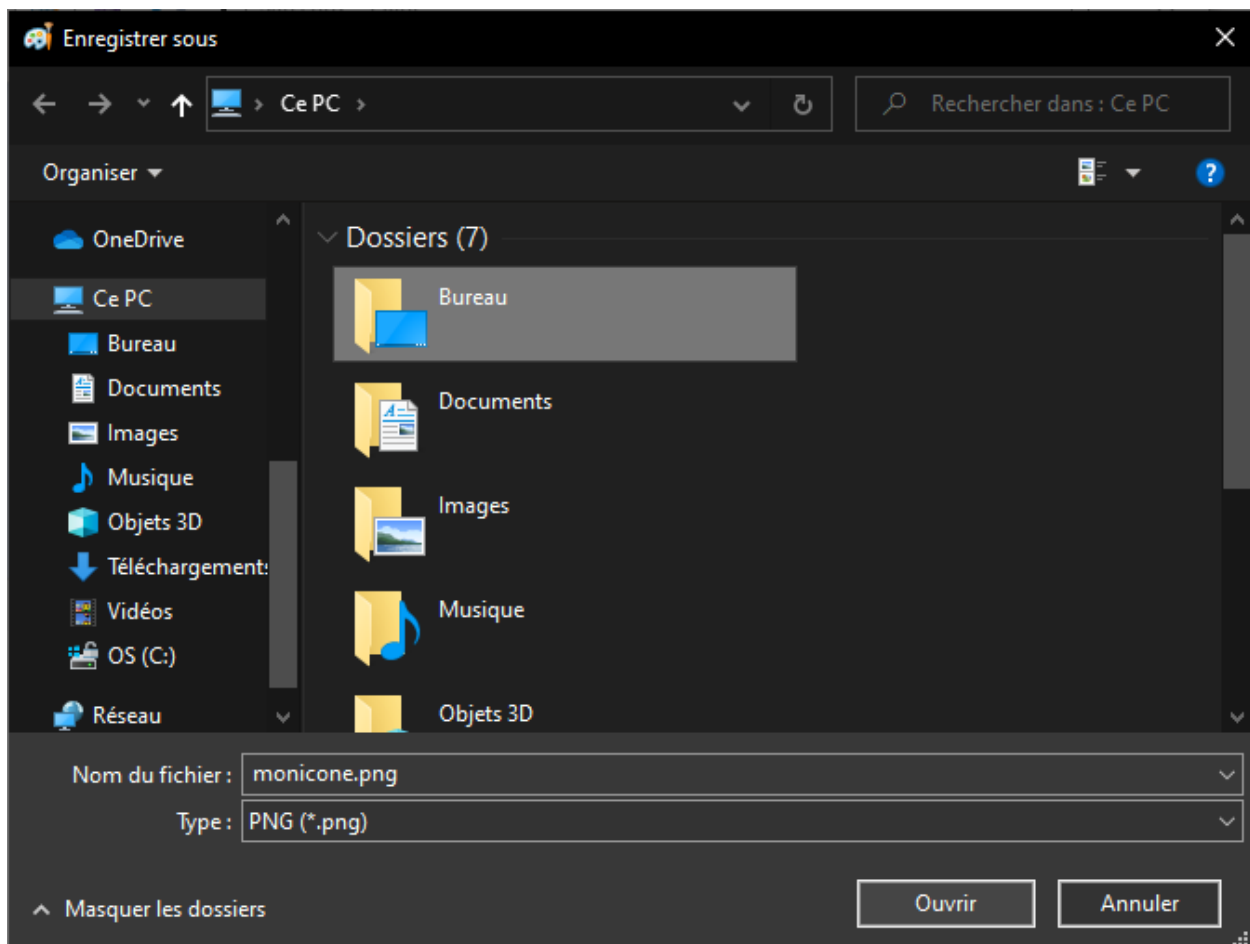
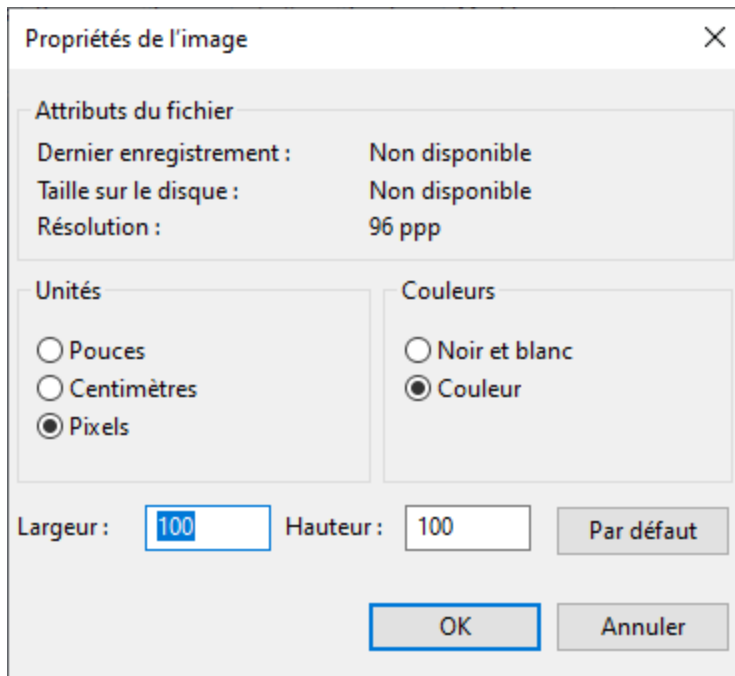


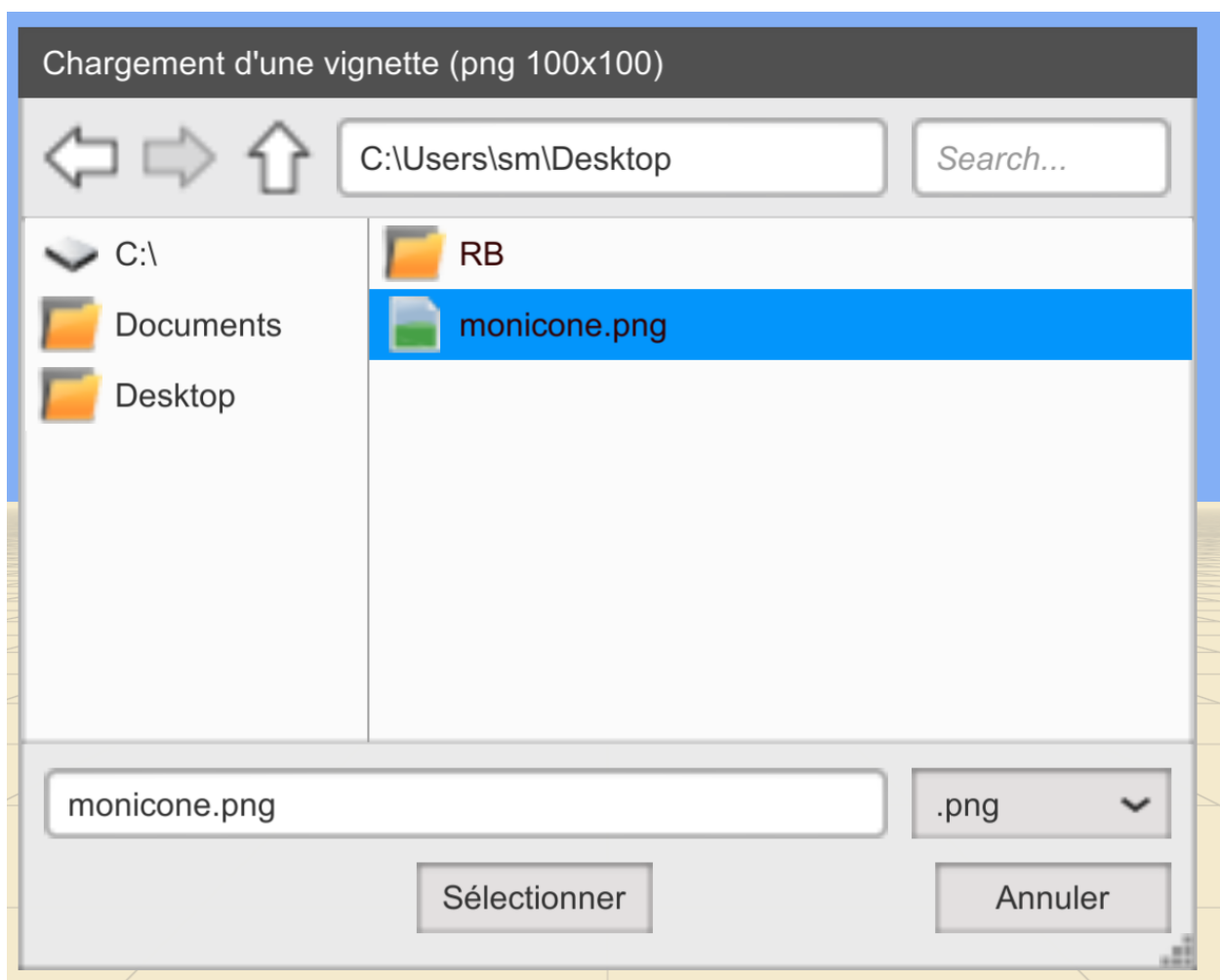
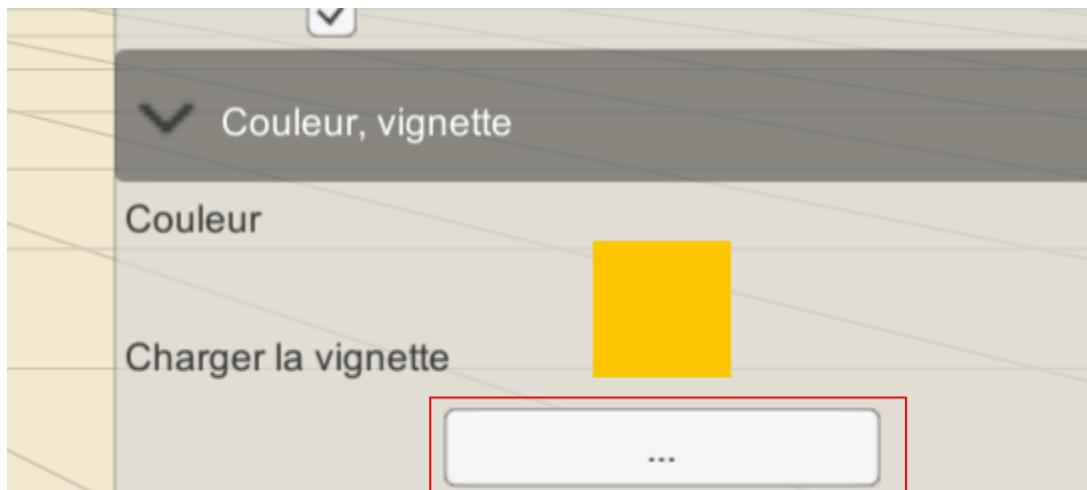
... continuons la personnalisation de notre système en définissant la catégorie dans laquelle notre robot doit apparaître ainsi que son nom et une icône personnalisée ...

The screenshot shows the 'Système' settings form. It has a dark blue header with a dropdown arrow and the text 'Système'. Below the header, there are two input fields. The first field is labeled 'Nom' and contains the text 'Mon Premier Robot'. The second field is labeled 'Catégorie' and contains the text 'Mes Robots'.



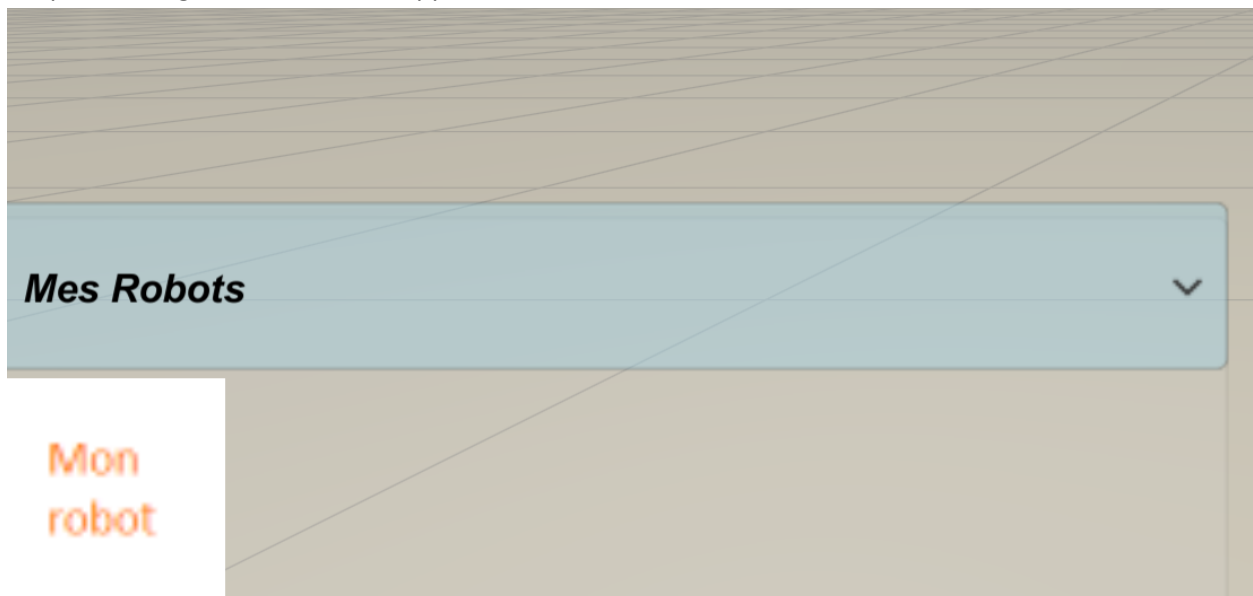
... save a 100x100 png file and select it...



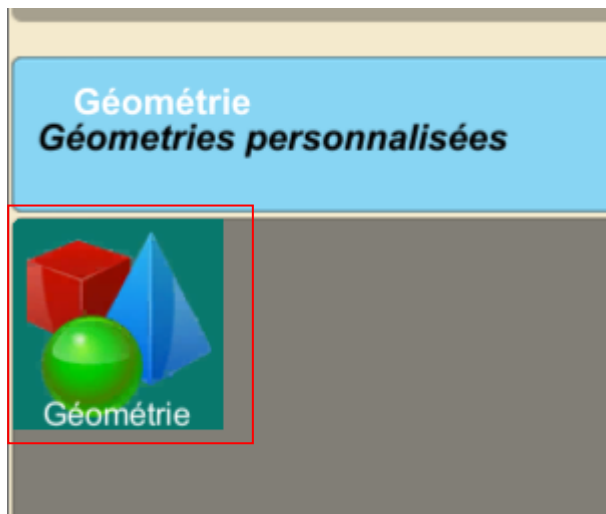




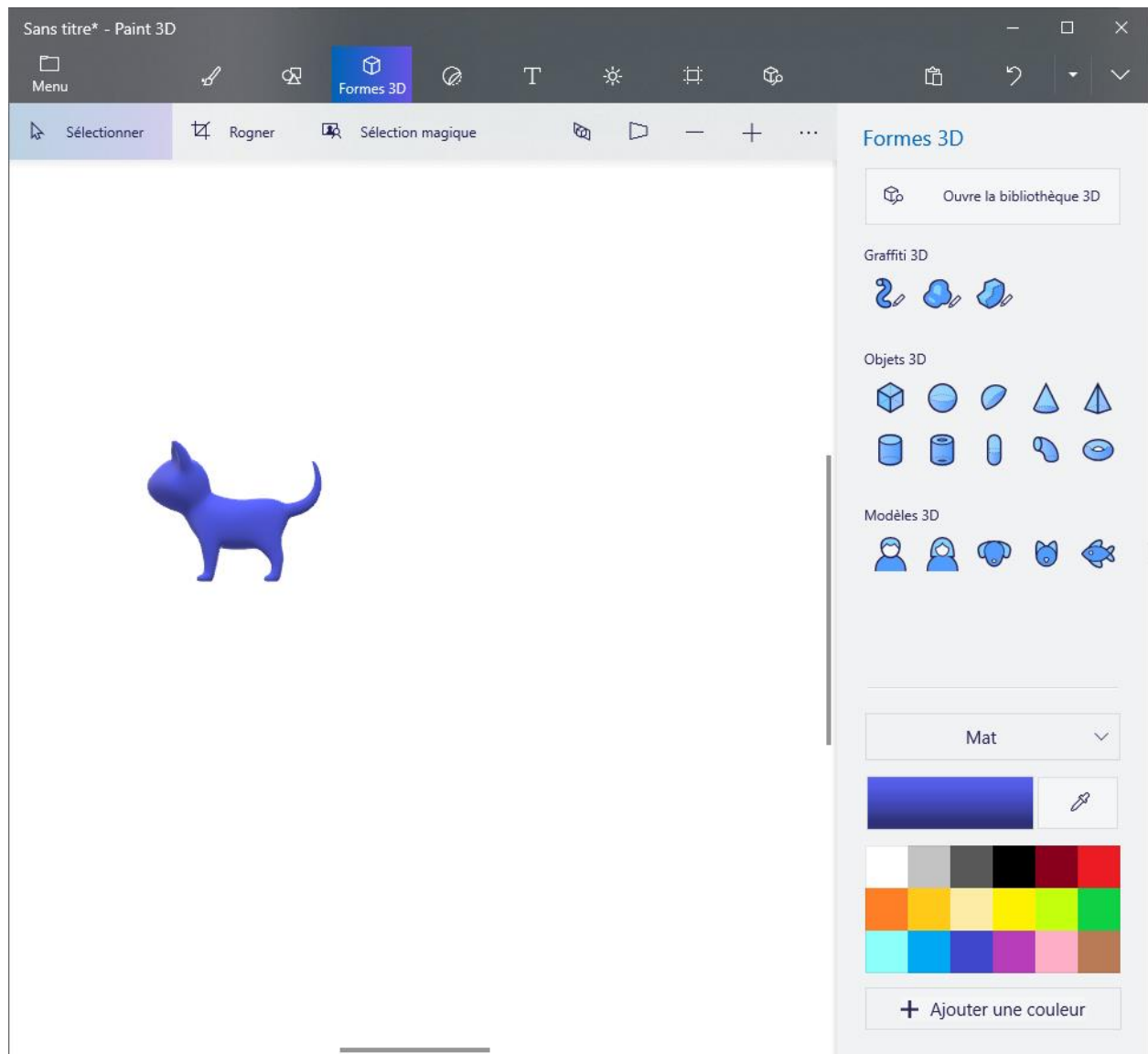
... après sauvegarde, votre robot apparaît ainsi dans l'éditeur de scènes ...

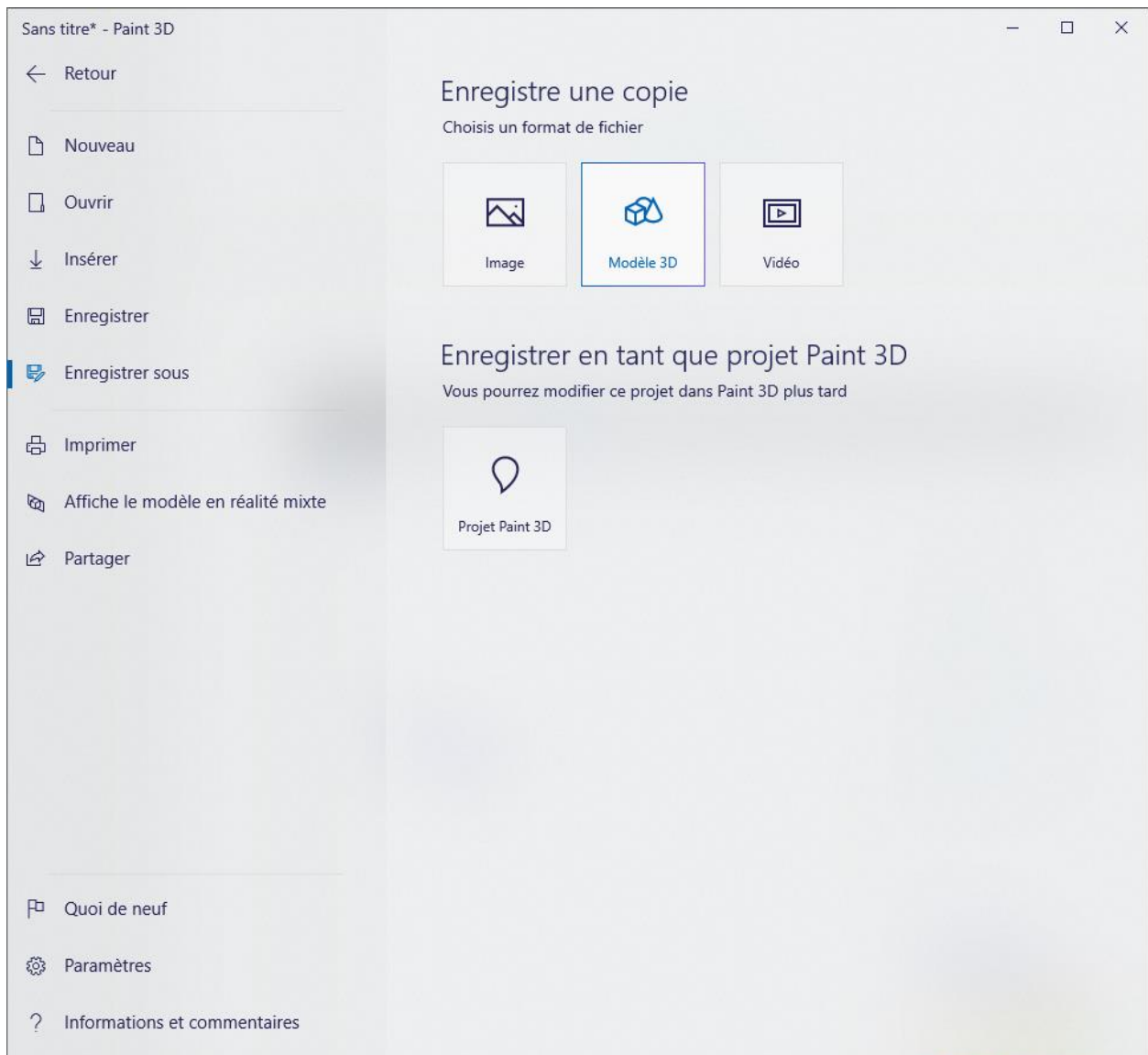


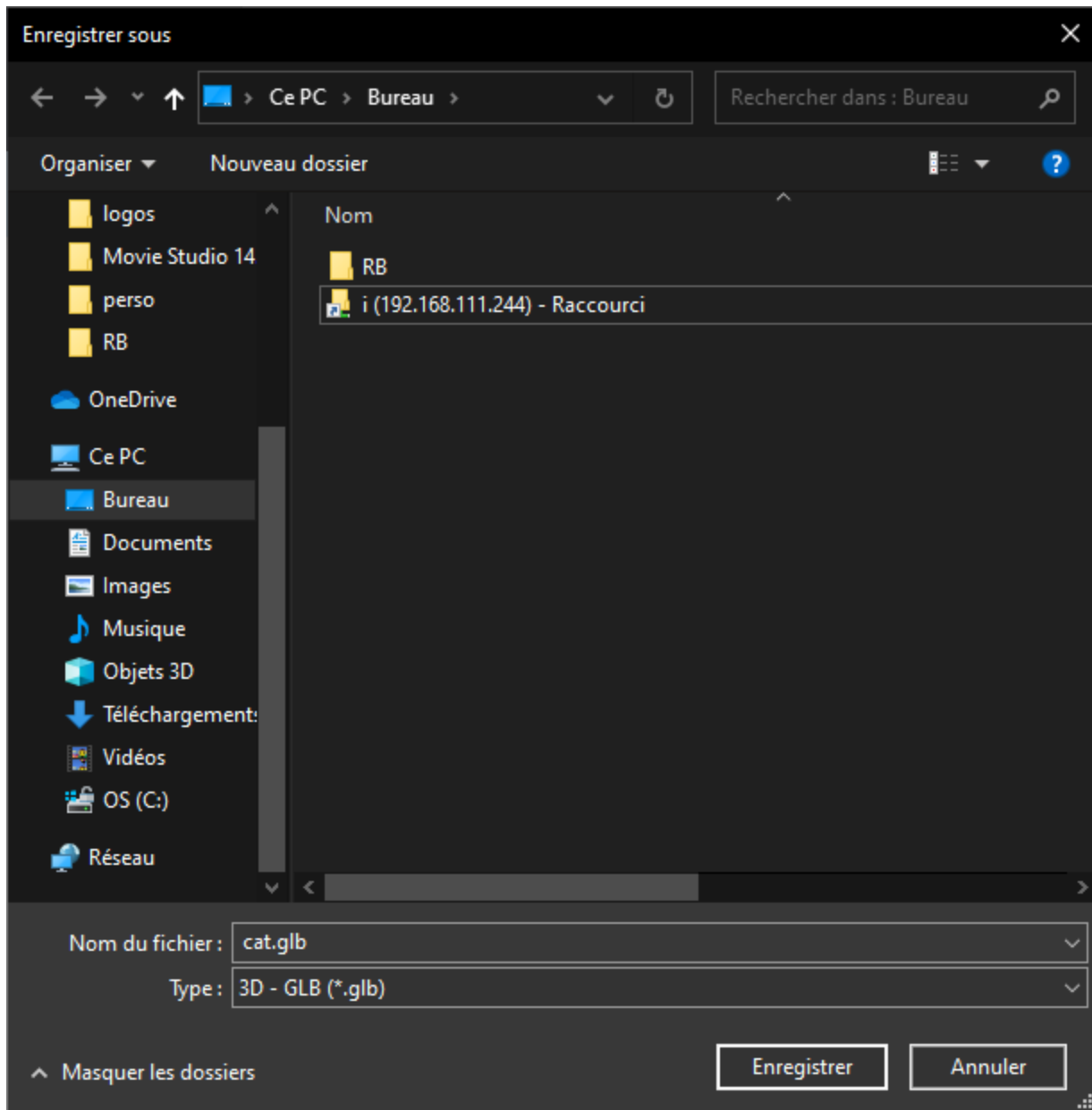
... ajoutons une géométrie personnalisée...

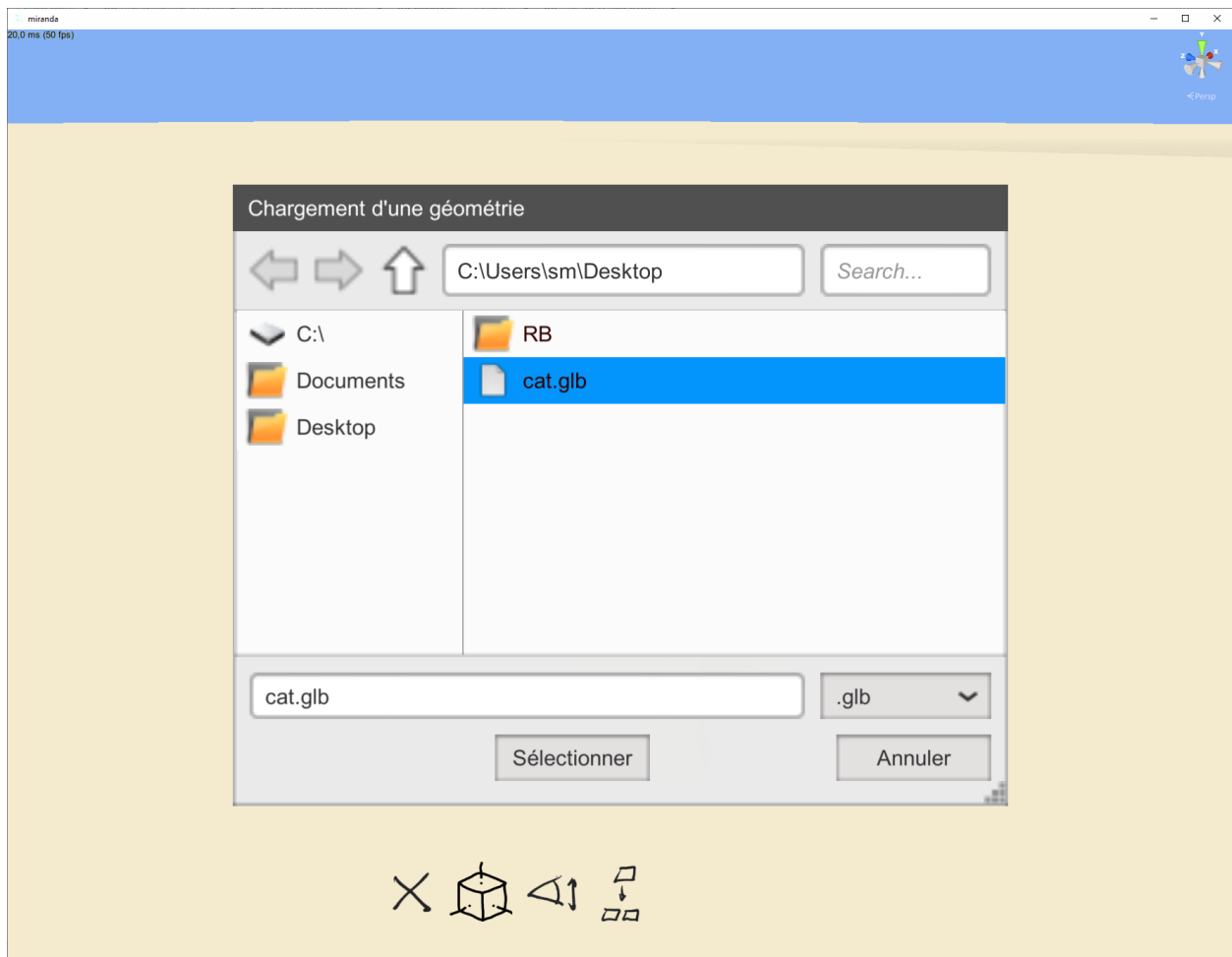
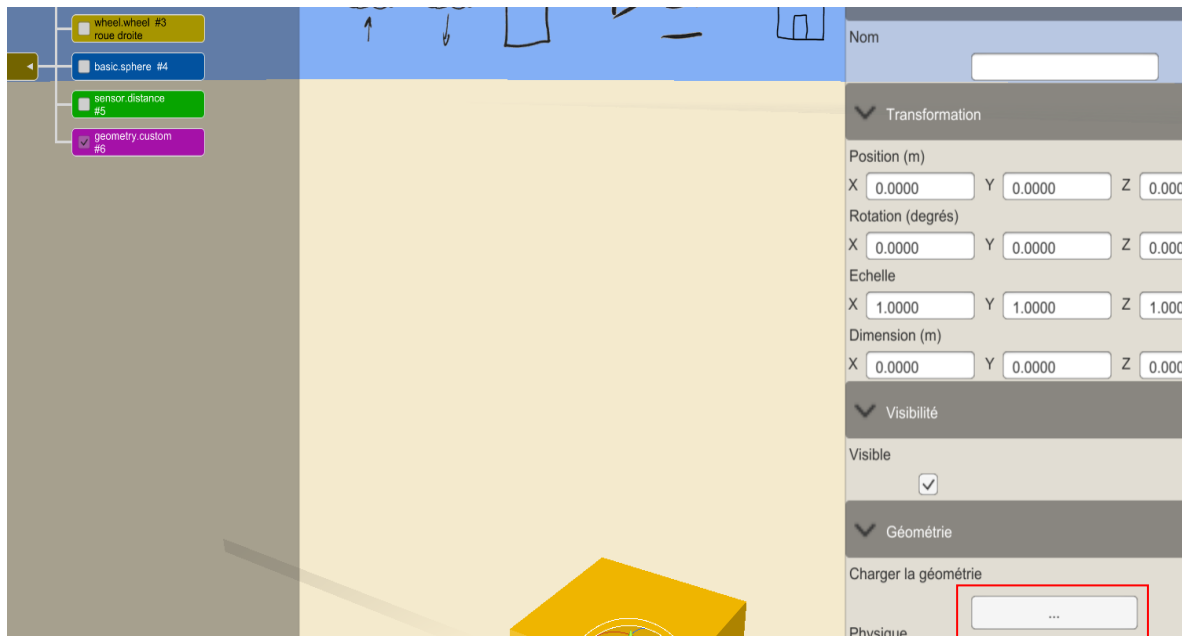


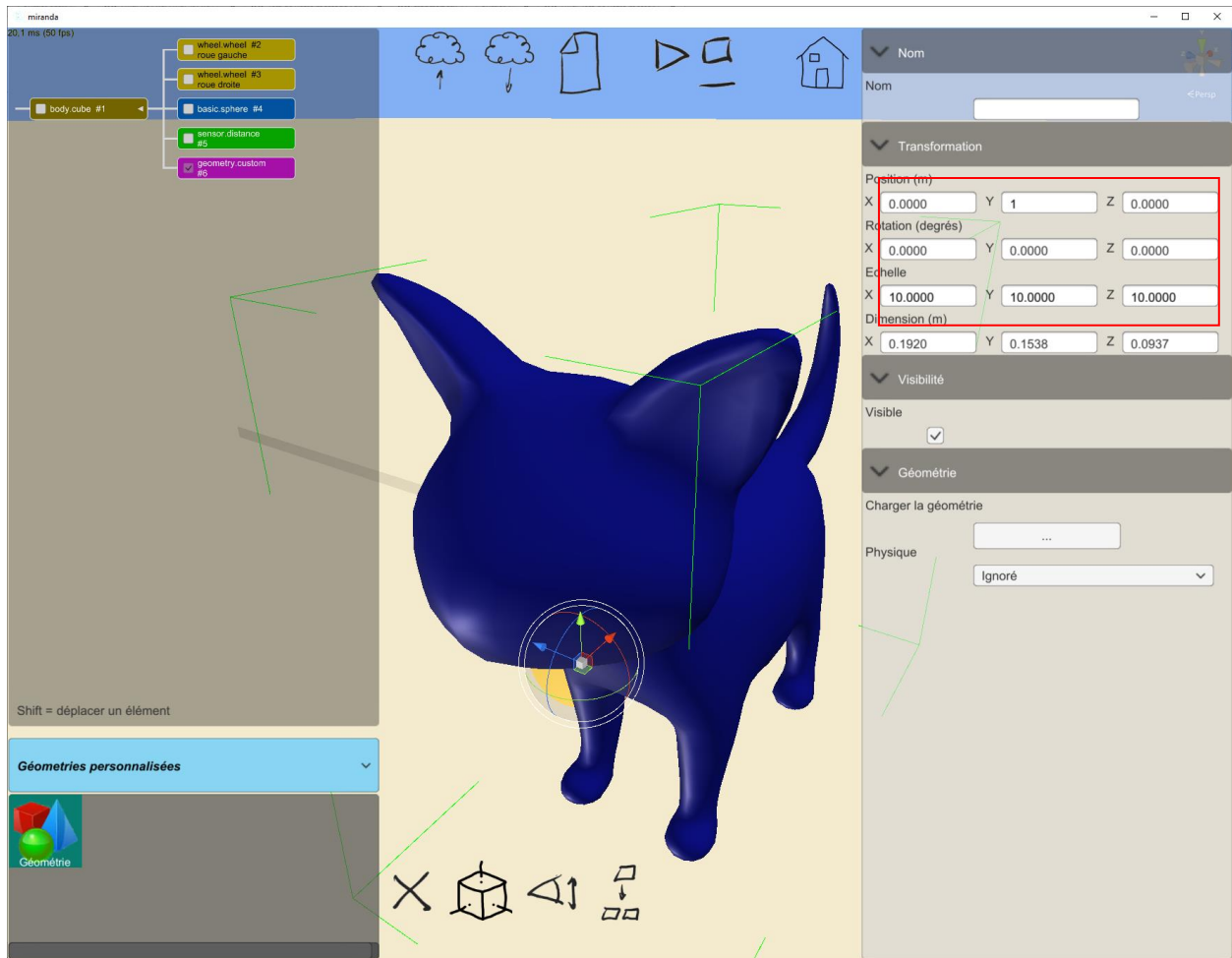
... le format du fichier 3d associé est « .glb », ces fichiers peuvent être créés avec Paint 3d (Windows 10) ou Blender...

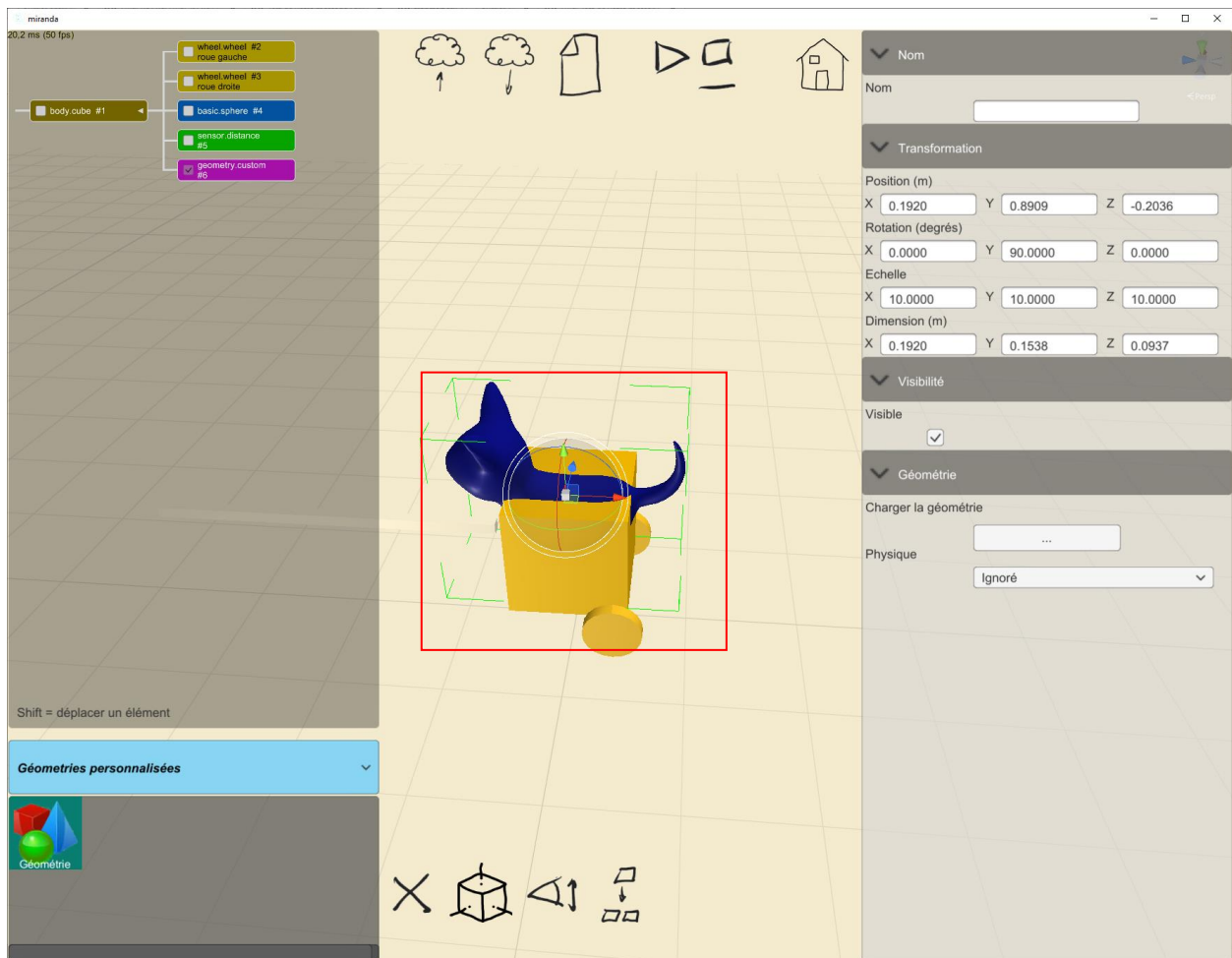




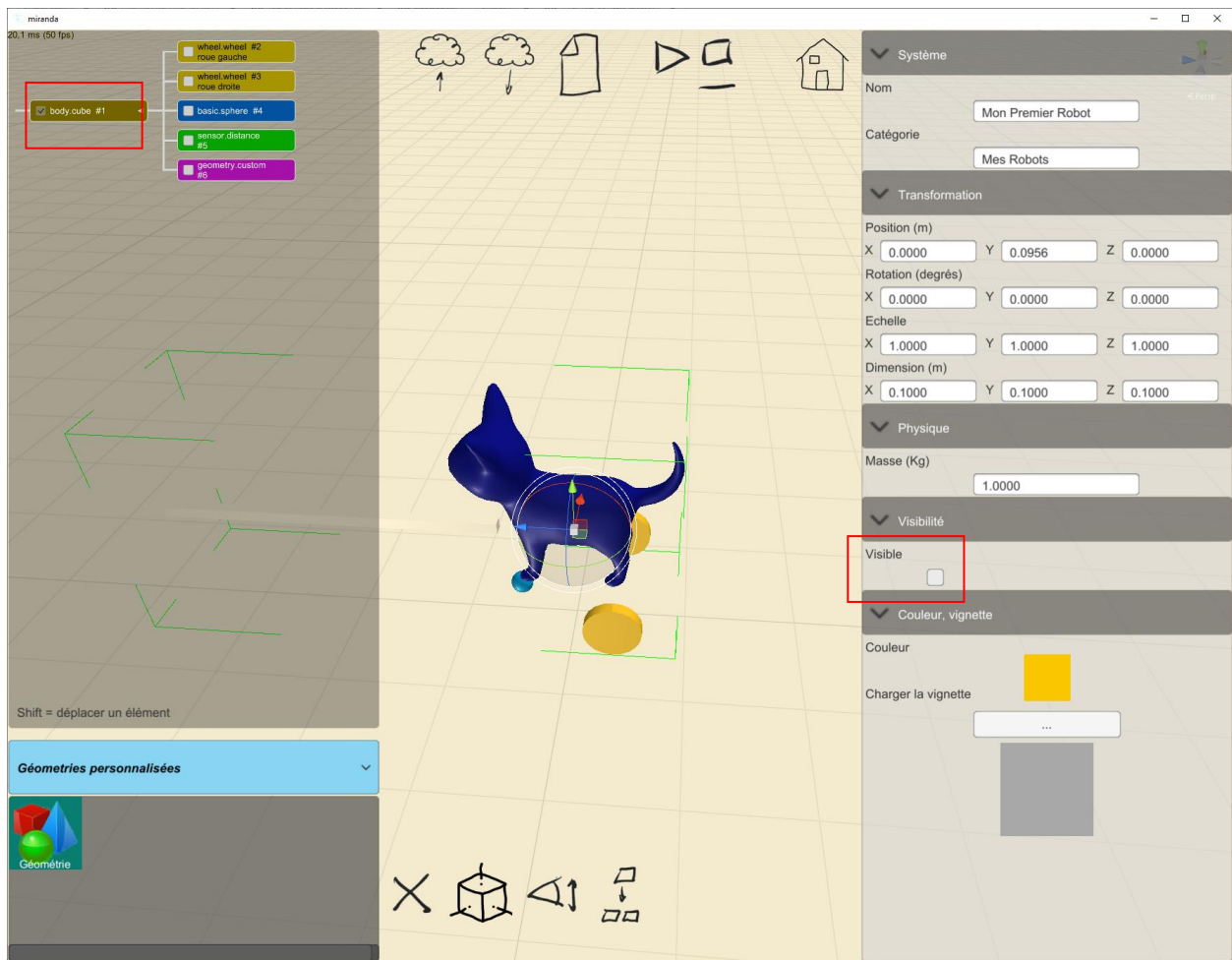








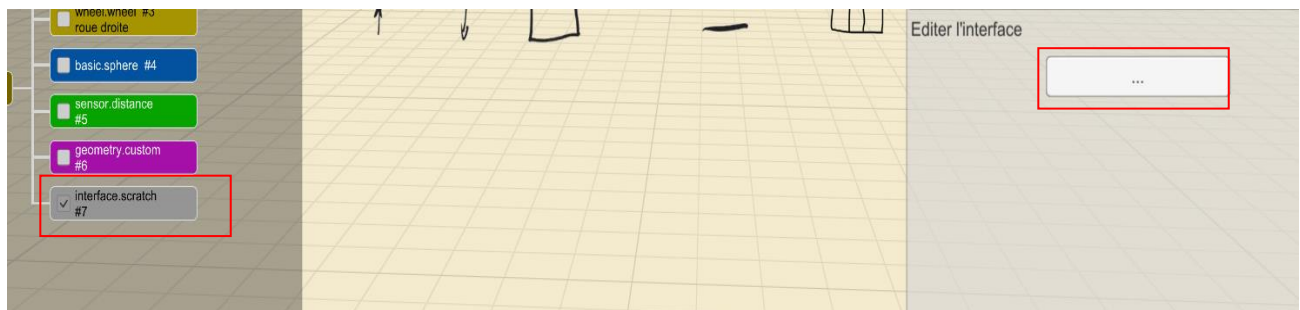
... les formes physiques peuvent être masquées pour ne laisser apparaître que les géométries personnalisées...



... personnalisons maintenant les blocs Scratches associés à notre robot...



... lorsque cet élément est ajouté à un système, les blocs Scratch par défaut sont entièrement remplacés par les blocs définis de la façon suivante...



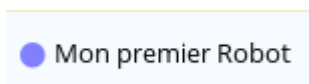
... cette partie est la plus complexe et nécessite d'écrire quelques lignes en langage Python pour définir l'aspect des blocs ainsi que leur fonctionnement interne. La première fonction Python à utiliser permet de créer une catégorie, en voici la syntaxe...

```
adb.addcat(<nom de la catégorie>,<couleur>)
```

... et un exemple ...

```
catid=adb.addcat("mon premier robot", "ff8080")
```

... affichera dans l'éditeur Scratch la catégorie comme ceci...



... si plusieurs catégories doivent être créées, plusieurs appels de cette fonction devront être réalisés. La fonction retourne un identifiant qui sera utilisé lors de l'ajout des blocs Scratches à la catégorie. Il y a plusieurs syntaxes pour l'ajout des blocs en fonction du type de bloc à ajouter. Pour un bloc d'action...

```
adb.addregularblk(<identifiant de catégorie>,<couleur du périmètre du bloc>,  
<couleur du fond du bloc>)
```

... un bloc retournant une valeur numérique ou texte...

```
adb.addvalueblk(<identifiant de catégorie>,<couleur du périmètre du bloc>,  
<couleur du fond du bloc>)
```

... un bloc retournant une valeur booléenne ...

```
adb.addbvalueblk(<identifiant de catégorie>,<couleur du périmètre du bloc>,  
<couleur du fond du bloc>)
```

... ces fonctions retournent un identifiant de bloc qui sera utilisé par les fonctions de création des éléments de blocs. Pour un texte...

```
adb.addstring(<identifiant de bloc>,<texte>)
```

... pour une zone de saisie de texte ou de valeur numérique...

```
adb.addvalue(<identifiant de bloc>,<valeur par défaut>)
```

... pour une zone booléenne...

```
adb.addbvalue(<identifiant de bloc>)
```

... pour une liste de choix...

```
adb.addoption(<identifiant de bloc>,<options séparées par ' ; '>,<valeur par défaut>)
```

... quelques exemples ...

```
id=adb.addregularblk(idcat,"000000","ff8000")  
adb.addstring(id,"roue gauche tourne à ")  
adb.addvalue(id,"100")  
adb.addstring(id," % de puissance, roue droite tourne à ")  
adb.addvalue(id,"100")  
adb.addstring(id,"% de puissance")
```



roue gauche tourne à 100 % de puissance, roue droite tourne à 100 % de puissance


```

id=adb.addregularblk(idcat,"000000","c000c0")
adb.addstring(id,"allumer la lumière ")
adb.addoption(id,"tout;gauche;droite","tout")
adb.addstring(id," avec la couleur rouge ")
adb.addvalue(id,"0")
adb.addstring(id," verte ")
adb.addvalue(id,"0")
adb.addstring(id," bleue ")
adb.addvalue(id,"0")

```



... comme dit précédemment, le script python défini à la fois les blocs et leur fonctionnement. Une variable nommée « mode » est documentée avec une valeur déterminant si le code doit définir les blocs (mode=0) ou si le code doit gérer le fonctionnement interne de chaque bloc (bloc=1 pour le fonctionnement du premier bloc, bloc=2 pour le deuxième bloc, etc...). Les blocs sont numérotés dans l'ordre de leur définition : le premier bloc défini porte le numéro 1, le deuxième le numéro 2, etc. Le code Python sera structuré par des test if...sur la variable mode. Si mode=0 alors définir les blocs, si mode=1 alors gérer l'exécution du premier bloc, etc. Le code Python doit utiliser une fonction nommée « adb.readvar("mode") » pour lire la valeur de la variable « mode ». Revenons à notre exemple et créons un bloc permettant de piloter les deux moteurs de notre robot...

#définition des blocs

```

if adb.readvar("mode")==0:
    idcat=adb.addcat("Mon premier Robot","ff8080")
    id=adb.addregularblk(idcat,"000000","ff8000")
    adb.addstring(id,"roue gauche tourne à ")
    adb.addvalue(id,"100")
    adb.addstring(id," % de puissance, roue droite tourne à ")
    adb.addvalue(id,"100")
    adb.addstring(id,"% de puissance")

```

#fonctionnement du bloc 1 : pilotage des moteurs

```
if adb.readvar("mode")==1:
```

```
    import monpremierrobot
```

```
    monpremierrobot.rouegauche.setpower(adb.readvar("arg2"))
```

```
    monpremierrobot.rouedroite.setpower(adb.readvar("arg4"))
```

... quelques explications sur la définition du fonctionnement du bloc sont nécessaires. « import monpremierrobot » est nécessaire pour pouvoir faire appel aux fonctions de notre robot. Les fonctions utilisables, dépendantes des éléments utilisés pour la création d'un système, sont automatiquement listées dans les commentaires en tête du script Python...

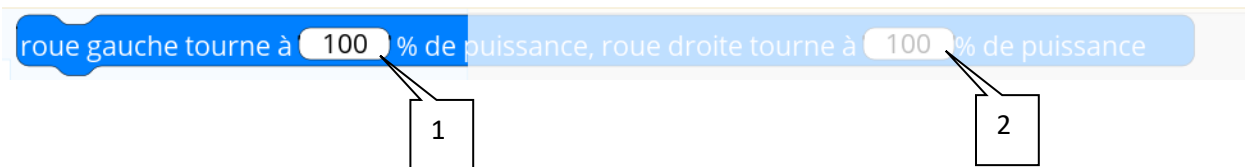
Définition des blocs

```
# monpremierrobot.rouegauche.setpower(<puissance (de -100 à 100)>)
```

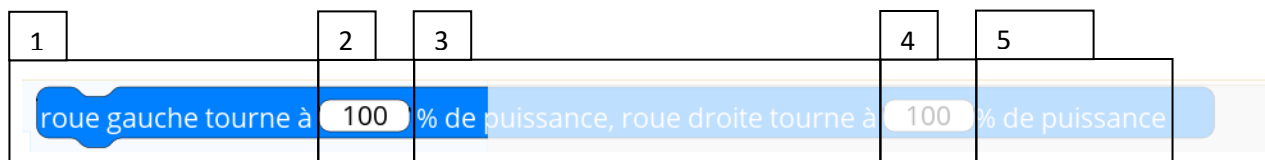
```
# monpremierrobot.rouedroite.setpower(<puissance (de -100 à 100)>)
```

```
# monpremierrobot.distancesensor()
```

... une autre explication est nécessaire sur la méthode utilisée pour référencer une valeur associée à un bloc. Notre bloc comporte deux valeurs...



... la position des valeurs est déterminée par leur ordre de création sur le bloc...



... les variables « arg<position> » permettent de faire référence aux variables du bloc par rapport à leur position. Pour notre exemple, « adb.readvar("arg2") » retourne la valeur pour la roue gauche et « adb.readvar("arg4") » la valeur pour la roue droite. Reste à passer ces valeurs aux fonctions correspondant à chaque roue : « monpremierrobot.rouegauche.setpower » et « monpremierrobot.rouedroite.setpower ». Poursuivons notre exemple en définissant un bloc qui retournera la distance mesurée par le capteur de distance en centimètres...

```
if adb.readvar("mode")==0:
```

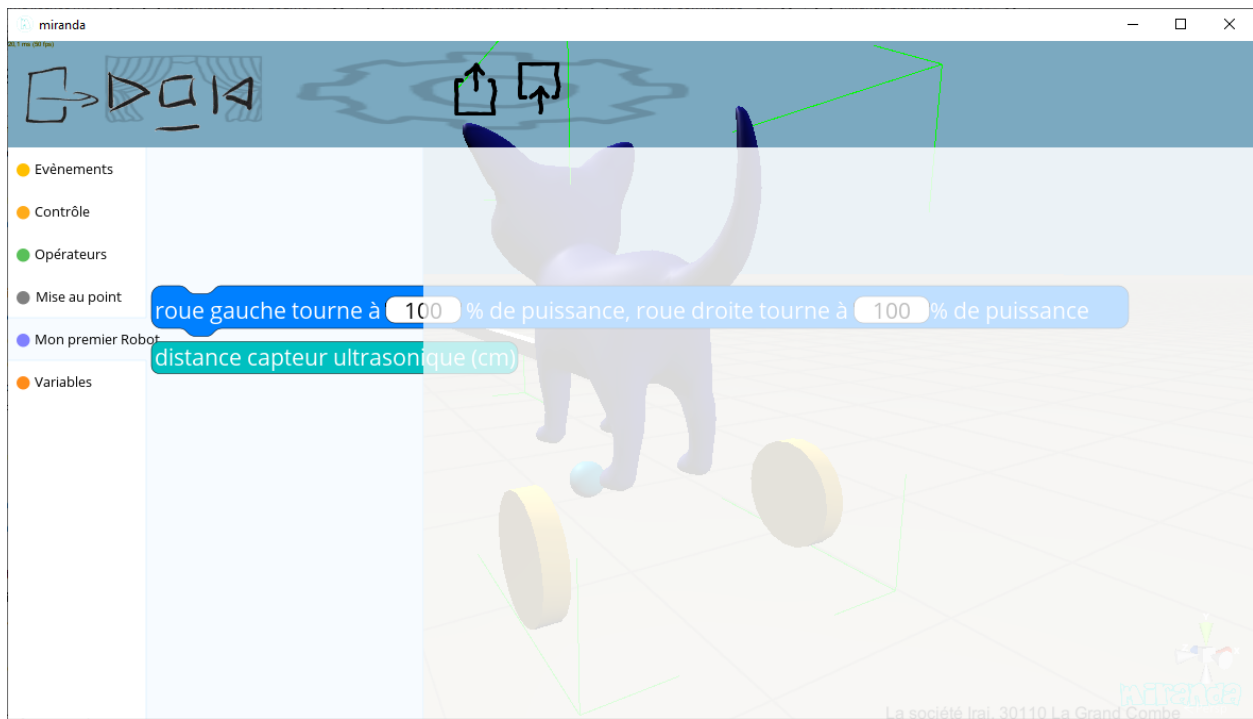
```
    idcat=adb.addcat("Mon premier Robot","ff8080")
```

```

id=adb.addregularblk(idcat,"000000","ff8000")
adb.addstring(id,"roue gauche tourne à ")
adb.addvalue(id,"100")
adb.addstring(id," % de puissance, roue droite tourne à ")
adb.addvalue(id,"100")
adb.addstring(id,"% de puissance")
id=adb.addvalueblk(idcat,"000000","c0c000")
adb.addstring(id,"distance capteur ultrasonique (cm)")
#fonctionnement du bloc 1 : pilotage des moteurs
if adb.readvar("mode")==1:
    import monpremierrobot
    monpremierrobot.rouegauche.setpower(adb.readvar("arg2"))
    monpremierrobot.rouedroite.setpower(adb.readvar("arg4"))
#fonctionnement du bloc 2 : capteur de distance en cm
if adb.readvar("mode")==2:
    import monpremierrobot
    adb.writevar("result",monpremierrobot.distancesensor()*30)

```

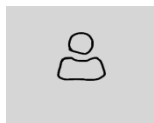
... quelques explications supplémentaires. Le bloc « capteur de distance » retourne une valeur. Ce retour de valeur est gérée par l'écriture depuis le Script Python d'une variable nommée « result » en utilisant la fonction « adb_writevar ». Le résultat est multiplié par 30, en effet, la fonction « monpremierrobot.distancesensor() » retourne une valeur comprise entre 0 et 1 en fonction de la distance mesurée et nous avons défini une zone de détection de 30cm de long. Nous en avons terminé avec cet exemple, après sauvegarde, l'interface de programmation de notre robot dans l'éditeur Scratch se présente ainsi...



Gestionnaire d'utilisateurs

Les versions « établissement » de miranda permettent de gérer des utilisateurs, généralement associés à des élèves ou groupes d'élèves.

L'accès à la liste des utilisateurs...



Nom	Mot de passe	Défi	Dernier essai	Nombre d'essais	Niveau atteint
irai:sm	1515				
irai:toto	1234				

... la ligne du bas permet d'ajouter des utilisateurs.

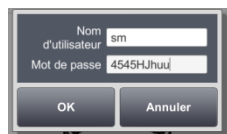
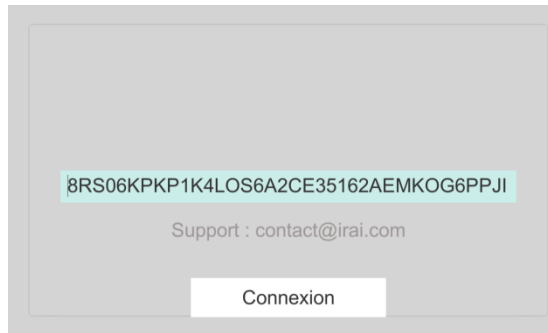
La zone « filtre »...

Nom	Mot de passe
irai:sm	1515

...permet d'afficher les utilisateurs dont le nom contient le filtre. Après modification du filtre, cliquer sur le bouton « Mettre à jour ».

Pour chaque utilisateur, les défis utilisés apparaissent avec le niveau d'avancement. Le bouton « Ouvrir » permet d'accéder au dernier programme créé pour chaque défi utilisé.

Le code affiché sur la ligne du haut permet aux utilisateurs de se connecter à miranda...



Le code peut être transmis au lancement de miranda. Ce code est toujours le même pour un même identifiant de connexion (code client) à miranda.

Pour la version s'exécutant dans un navigateur Web :

<url> ?<CODE>

Par exemple :

<http://irai2.com/mir?8RS06KPKP1K4LOS6A2CE35162AEZKOG6PPJIK898BQK2J058G9EA2I1>

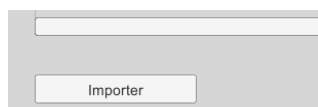
Pour la version exécutable :

<chemin d'accès à l'exécutable miranda> CODE=<CODE>

Par exemple :

"C:\program files\miranda\miranda.exe" CODE=8RS06KPKP1K4LOS6A2CE35162AEMKOG6PPJIK8Z8BQ32J058G9EA3I1

Les utilisateurs peuvent être importés (version exécutable de miranda uniquement) ...



... depuis un fichier texte.

Le fichier texte peut contenir, sur chaque ligne, un nom de l'utilisateur suivi, éventuellement d'un mot de passe.

Exemples de fichiers:

Elisa
Albert

Elisa,1234
Albert,9856

Elisa ;1234
Albert ;9856

Si le mot de passe n'est pas présent, un mot de passe est automatiquement généré.

Les utilisateurs affichés peuvent également être exportés.

La liste d'utilisateurs peut être supprimée.

Groupes d'utilisateurs

Le nom des utilisateurs peut contenir une entête suivi de « : » définissant l'appartenance à un groupe.

Par exemple...

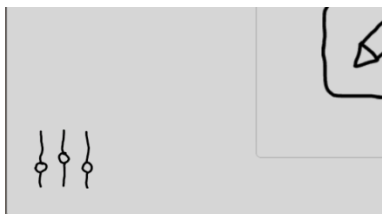
irai:sm

... utilisateur « sm » appartenant groupe « irai ».

Remarque : les utilisateurs doivent entrer le nom complet lors de la connexion. Par exemple :

irai:sm

L'accès à la liste des groupes...



Nom	Nom(s) des scènes	Editeur de scènes	Editeur de systèmes	
Irai	robot,thymio	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Supprimer
		<input type="checkbox"/>	<input type="checkbox"/>	

Mettre à jour

... pour chaque groupe, est défini : son nom, une liste des scènes accessibles, la possibilité d'utiliser ou non l'éditeur de scènes et l'éditeur de systèmes.

La liste de scènes accessibles peut contenir un ou plusieurs débuts de nom de scène séparés par « ; » .

Si cette liste est vide, le mode de fonctionnement par défaut est appliqué : seulement les scène dont le nom commence par « : ».