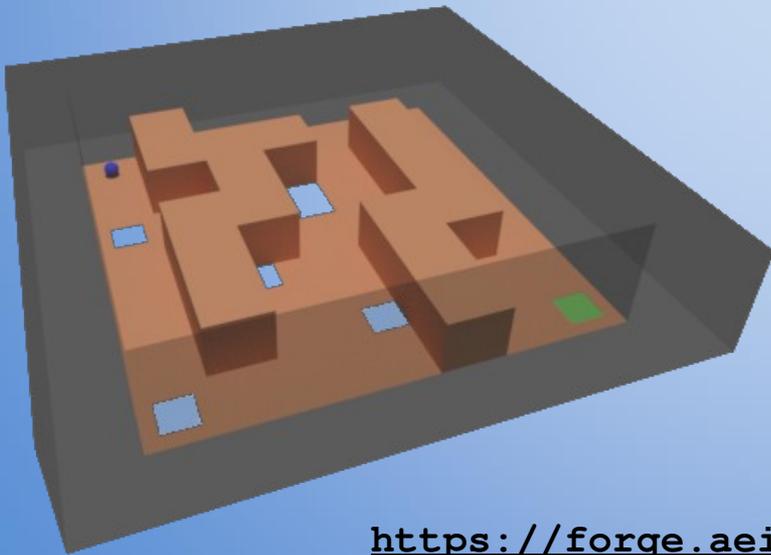


Labyrinthe à bille

Créer une scène 3D interactive

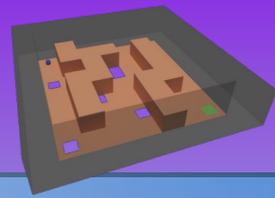
Tutoriel 2

Passage au Python



<https://forge.aeif.fr/blender-edutech/blender-edutech-tuto>

Objectif



L'objectif de ce tutoriel est de programmer le comportement des objets par des règles codées en Python. Nous allons donc reprendre le fichier blender du tutoriel précédent et remplacer les règles définies par les **briques logiques** avec **un module Python**.

Le codage en Python permet

- d'établir plus efficacement des règles plus complexes,
- d'accéder à des instructions plus précises,
- de séparer le fond (comportement) et la forme (objets 3D),
- de maintenir le code avec plus d'efficacité (gestion des versions).

Le tutoriel se décompose en 5 étapes :

- Installer l'éditeur de texte
- **Déplacer** le plateau
- **Définir le game play** (règles d'échec et de réussite)
- **Animer** la fenêtre de fin
- Fermer la fenêtre de fin par un **bouton cliquable**

1. Installation de l'éditeur de texte



Le codage se fait par un éditeur de texte. Il en existe beaucoup et tout éditeur peut convenir. Pour ce tutoriel j'utilise **Emacs**, il est open source, très efficace et polyvalent mais peu intuitif. Le choix de l'éditeur est souvent très personnel, sans préférence je vous conseille **Spyder**, il est open source et complet.

- **Emacs** ce trouve à cette adresse : <https://www.gnu.org/software/emacs>
- **Spyder** ce trouve à cette adresse : <https://www.spyder-ide.org>

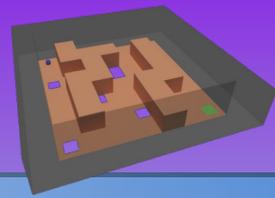
```
1 import bge # Bibliothèque Blender / Game Engine (BGE)
2
3 #####
4 # labyrinth.py
5 # @title: Commandes pour le tutotiel Labyrinthe
6 # @project: Blender-EduTech
7 # @lang: fr
8 # @authors: Philippe Roy <philippe.roy@ac-grenoble.fr>
9 # @copyright: Copyright (C) 2021 Philippe Roy
10 # @license: GNU GPL
11 #
12 # Commandes déclenchées par UPBGE pour le tutoriel Labyrinthe
13 #
14 #####
15
16 # Récupérer la scène 3D
17 scene = bge.logic.getCurrentScene()
18 # print("Objets de la scene : ", scene.objects)
19
20 # Constantes
21
22 JUST_ACTIVATED = bge.logic.KX_INPUT_JUST_ACTIVATED
23 JUST_RELEASED = bge.logic.KX_INPUT_JUST_RELEASED
24 ACTIVATE = bge.logic.KX_INPUT_ACTIVE
25 JUST_DEACTIVATED = bge.logic.KX_SENSOR_JUST_DEACTIVATED
26
27 #####
28 # Gestion du clavier
29 #####
30
31 # Flèches pour tourner le plateau
32 def clavier(cont):
33     # obj = cont.owner
34     obj = scene.objects["Plateau"]
35     keyboard = bge.logic.keyboard
36     resolution = 0.01
37
38     # Up
39     if (ACTIVATE == keyboard.events[bge.events.UPARROWKEY]):
40         obj.applyRotation((-resolution,0,0), False)
41
42     # Down
43     if (ACTIVATE == keyboard.events[bge.events.DOWNARROWKEY]):
44         obj.applyRotation((resolution,0,0), False)
```

Emacs

```
5 # labyrinth.py
6 # @title: Commandes pour le tutotiel Labyrinthe
7 # @project: Blender-EduTech
8 # @lang: fr
9 # @authors: Philippe Roy <philippe.roy@ac-grenoble.fr>
10 # @copyright: Copyright (C) 2021 Philippe Roy
11 # @license: GNU GPL
12 #
13 # Commandes déclenchées par UPBGE pour le tutoriel Labyrinthe
14 #
15 #####
16 # Récupérer la scène 3D
17 scene = bge.logic.getCurrentScene()
18 # print("Objets de la scene : ", scene.objects)
19
20 # Constantes
21
22 JUST_ACTIVATED = bge.logic.KX_INPUT_JUST_ACTIVATED
23 JUST_RELEASED = bge.logic.KX_INPUT_JUST_RELEASED
24 ACTIVATE = bge.logic.KX_INPUT_ACTIVE
25 JUST_DEACTIVATED = bge.logic.KX_SENSOR_JUST_DEACTIVATED
26
27 #####
28 # Gestion du clavier
29 #####
30
31 # Flèches pour tourner le plateau
32 def clavier(cont):
33     # obj = cont.owner
34     obj = scene.objects["Plateau"]
35     keyboard = bge.logic.keyboard
36     resolution = 0.01
37
38     # Up
39     if (ACTIVATE == keyboard.events[bge.events.UPARROWKEY]):
40         obj.applyRotation((-resolution,0,0), False)
41
42     # Down
43     if (ACTIVATE == keyboard.events[bge.events.DOWNARROWKEY]):
44         obj.applyRotation((resolution,0,0), False)
```

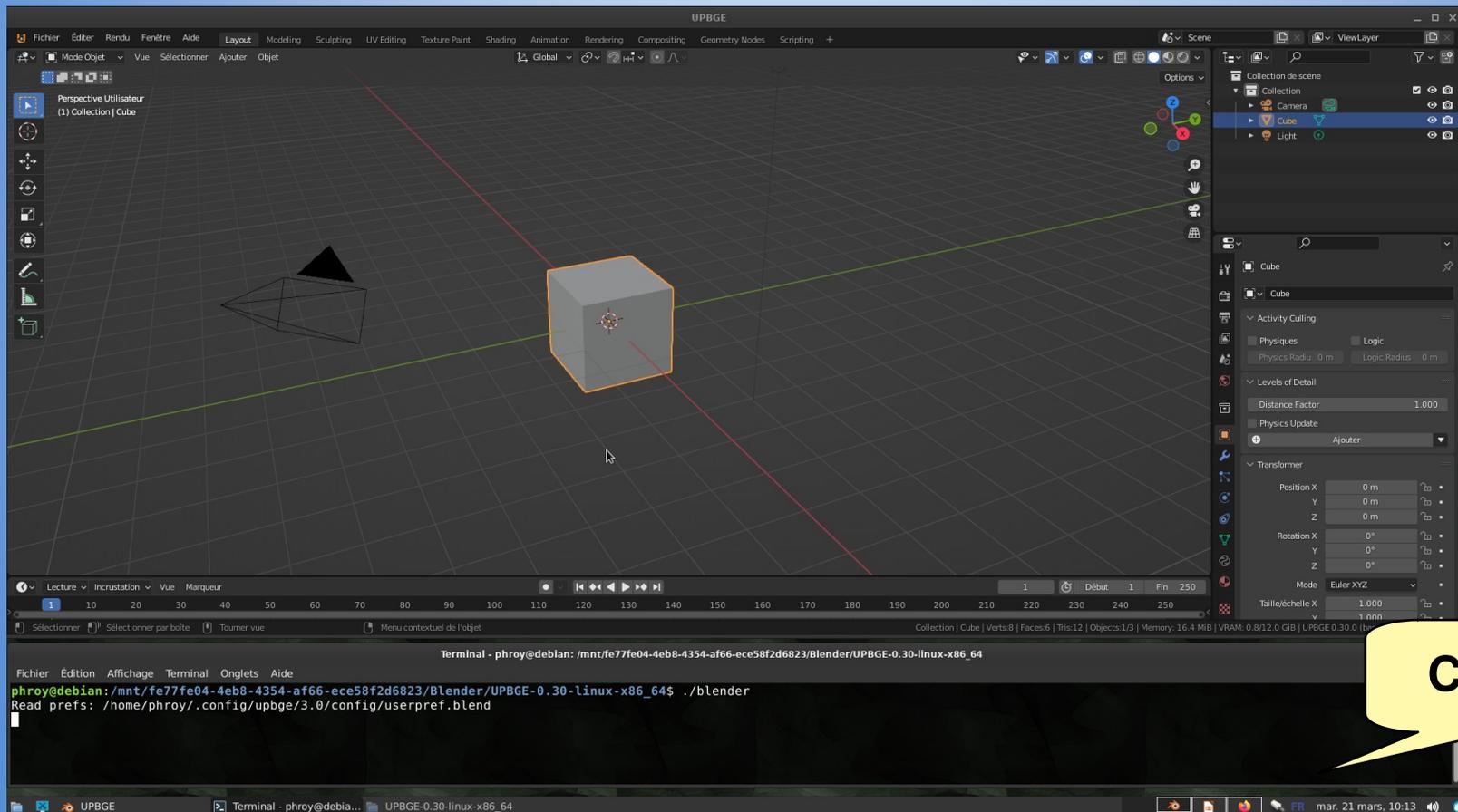
Spyder

2. Déplacer le plateau



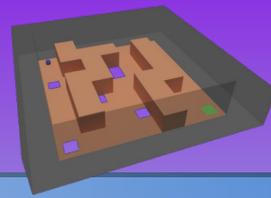
Le moteur de jeu **UPBGE** (**UP Blender Game Engine**) intègre déjà Blender, donc seule son installation suffit. Lancer UPBGE dans une **console** permet de visualiser les messages (erreurs, sortie standard, ...).

- **UPBGE** (version 0.3+) se trouve à cette adresse : <https://upbge.org>



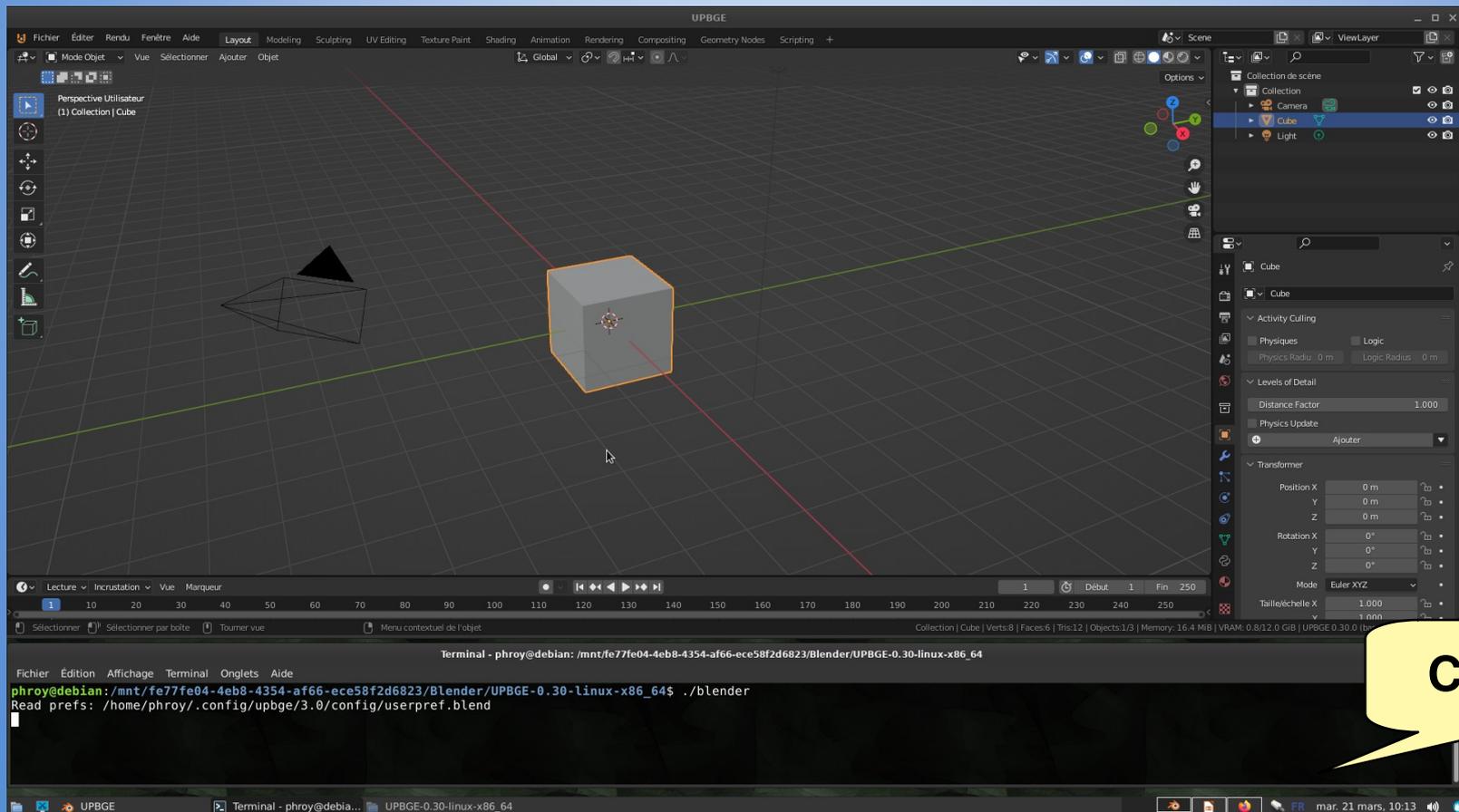
Console

3. Définir le game play (règles d'échec et de réussite)



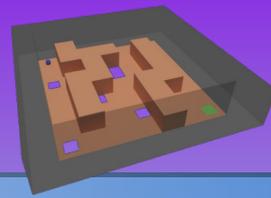
Le moteur de jeu **UPBGE** (**UP** Blender **G**ame **E**ngine) intègre déjà Blender, donc seule son installation suffit. Lancer UPBGE dans une **console** permet de visualiser les messages (erreurs, sortie standard, ...).

- **UPBGE** (version 0.3+) se trouve à cette adresse : <https://upbge.org>



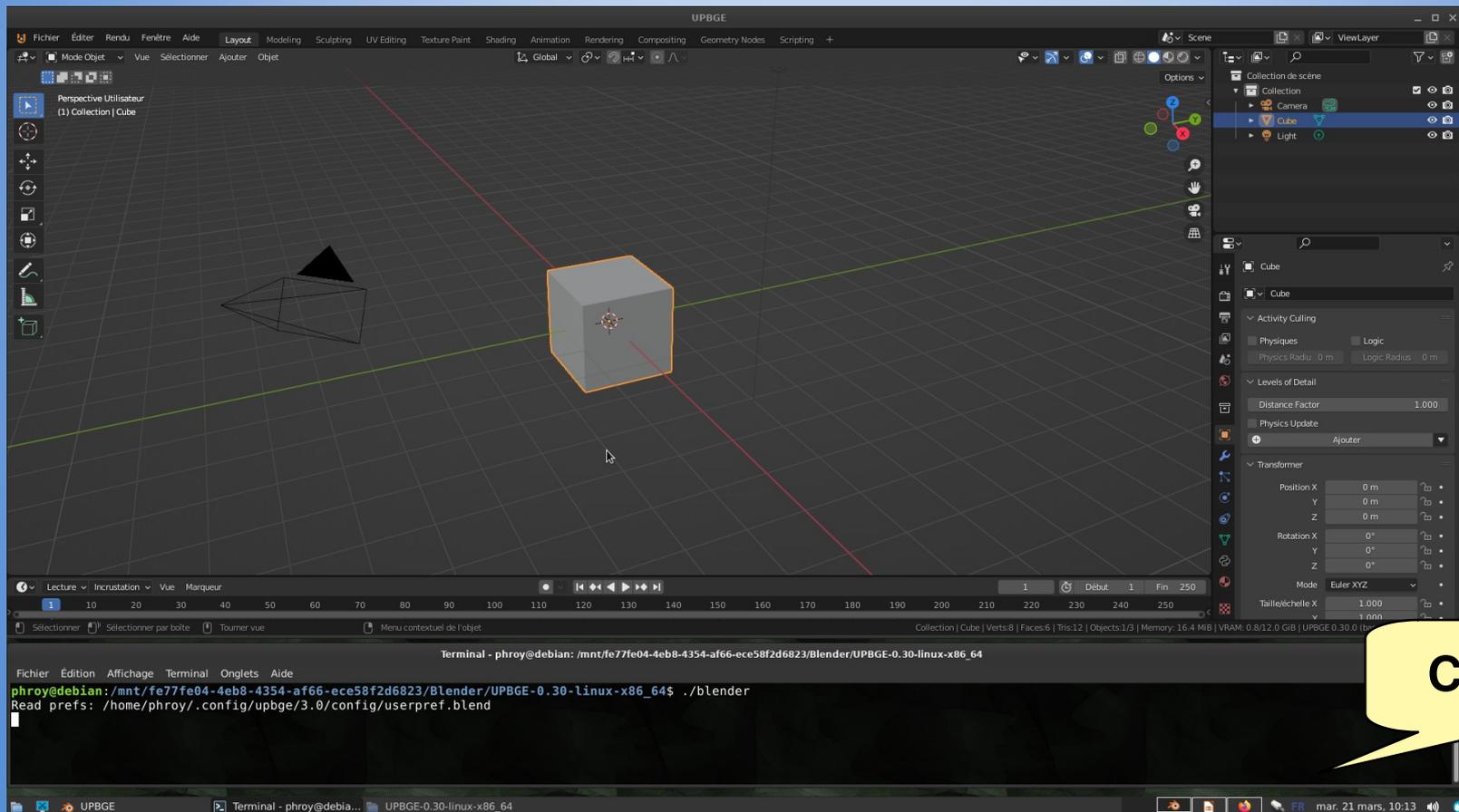
Console

4. Animer la fenêtre de fin

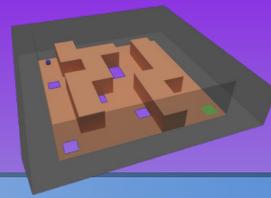


Le moteur de jeu **UPBGE** (**UP Blender Game Engine**) intègre déjà Blender, donc seule son installation suffit. Lancer UPBGE dans une **console** permet de visualiser les messages (erreurs, sortie standard, ...).

- **UPBGE** (version 0.3+) se trouve à cette adresse : <https://upbge.org>

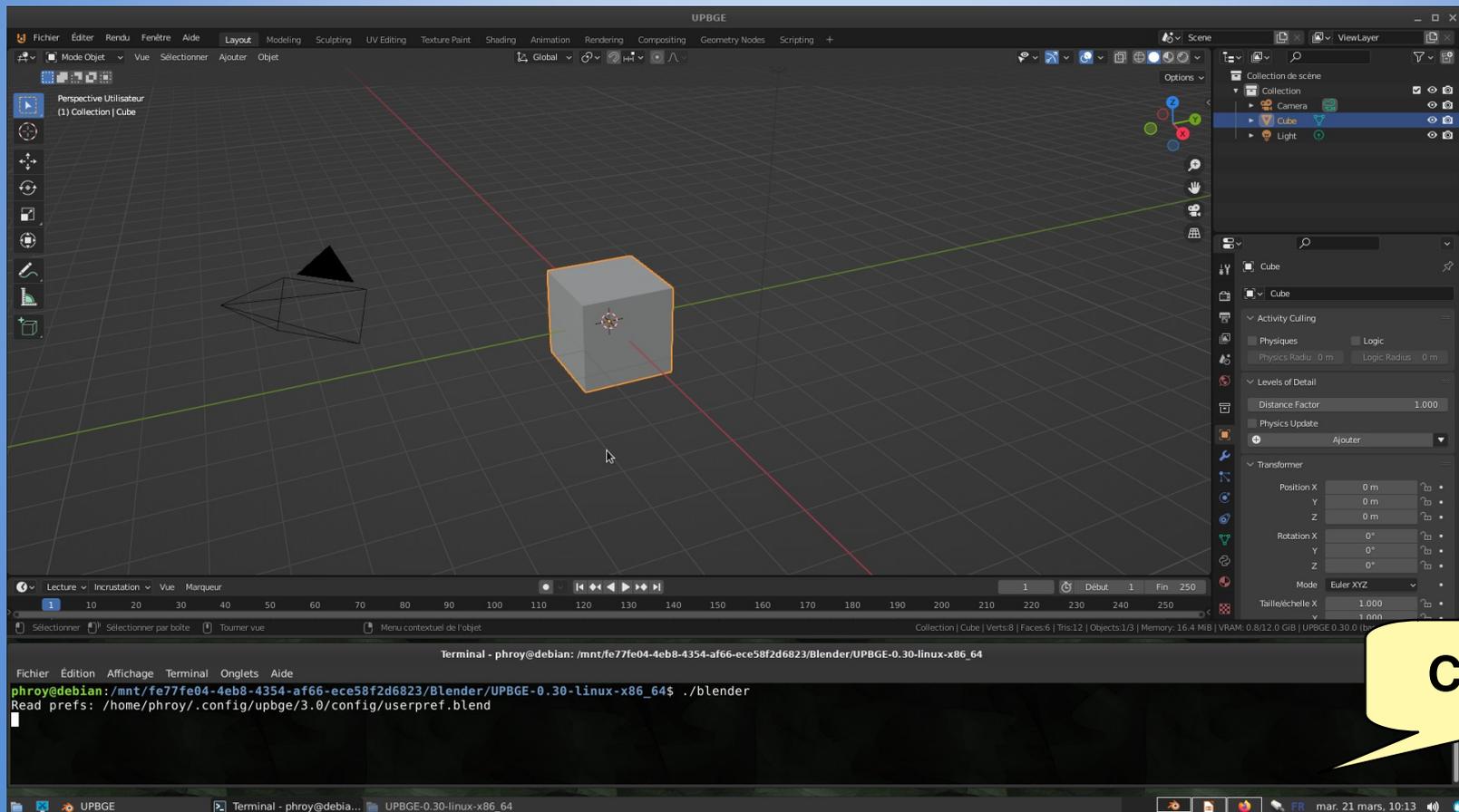


5. Fermer la fenêtre de fin par un bouton cliquable



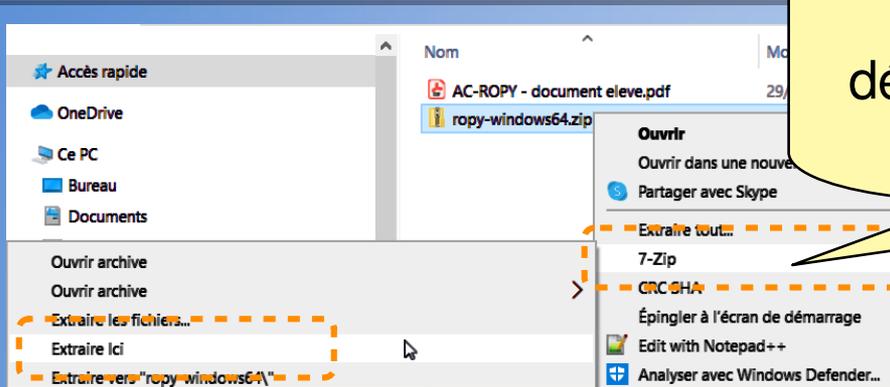
Le moteur de jeu **UPBGE** (**UP** Blender **G**ame **E**ngine) intègre déjà Blender, donc seule son installation suffit. Lancer UPBGE dans une **console** permet de visualiser les messages (erreurs, sortie standard, ...).

- **UPBGE** (version 0.3+) se trouve à cette adresse : <https://upbge.org>



Console

Mettre en place l'environnement de développement



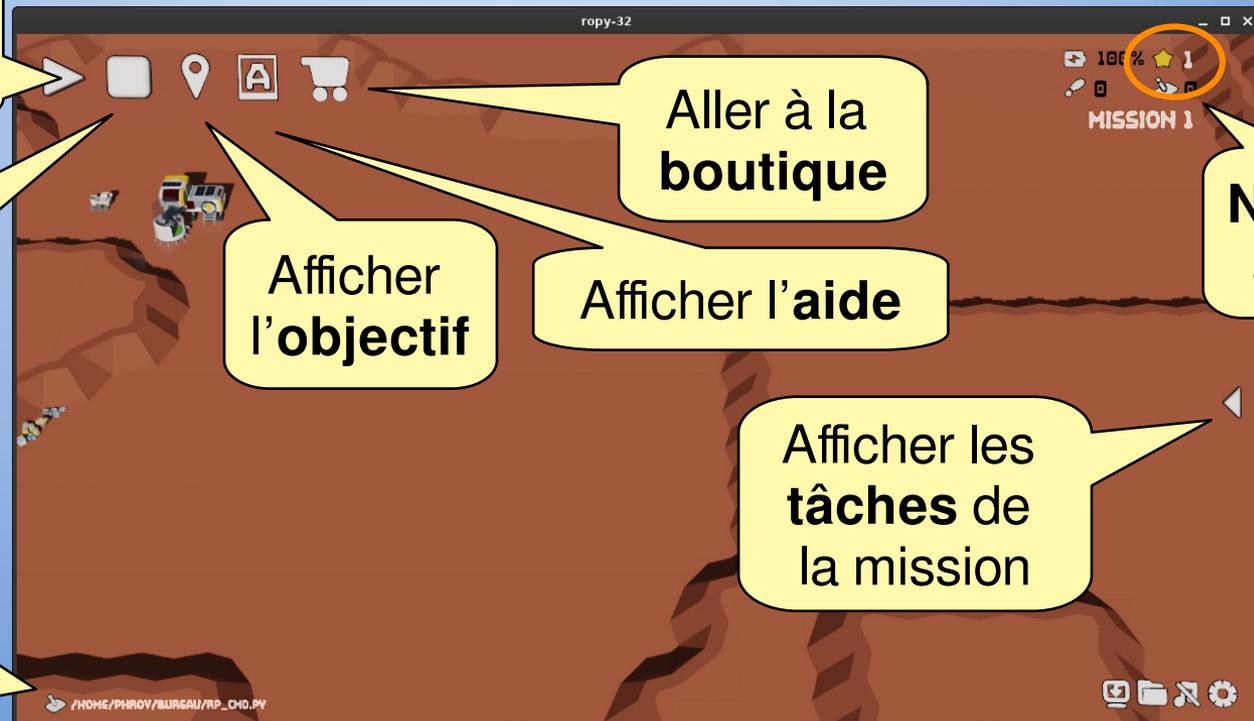
1 : Récupérer l'archive **ropy-windows64.zip** et la décompresser avec **7-Zip** sur le **bureau**. L'extraction va créer le répertoire ropy

Le **simulateur** et la **console** se lancent en même temps avec **ropy.bat** (situé dans le répertoire créé).

Exécuter le programme

Arrêter et réinitialiser

Fichier de commandes



Aller à la boutique

Afficher l'objectif

Afficher l'aide

Niveau actuel

Afficher les tâches de la mission

Mettre en place l'environnement de développement

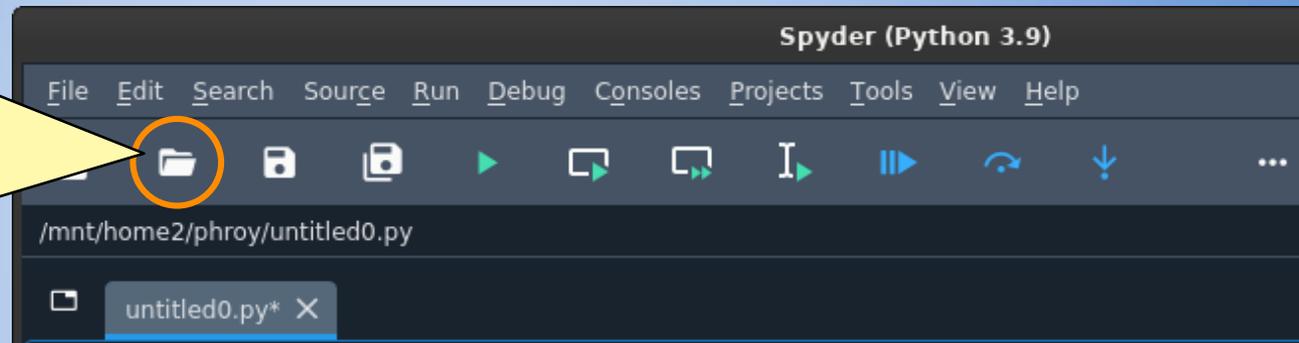


2 : Copier **dans votre répertoire** le fichier de commandes : **ropy_cmd.py** (ropy commandes).

3 : Lancer **Spyder**.



4 : Dans **Spyder** ouvrir le fichier de commandes qui a été précédemment copié dans votre répertoire.



6 : Le nom de votre fichier doit apparaître ici.

/HOME/PHROY/BUREAU/ROP_CMD.PY

5 : Dans le **simulateur**, définir votre fichier comme fichier de commandes.

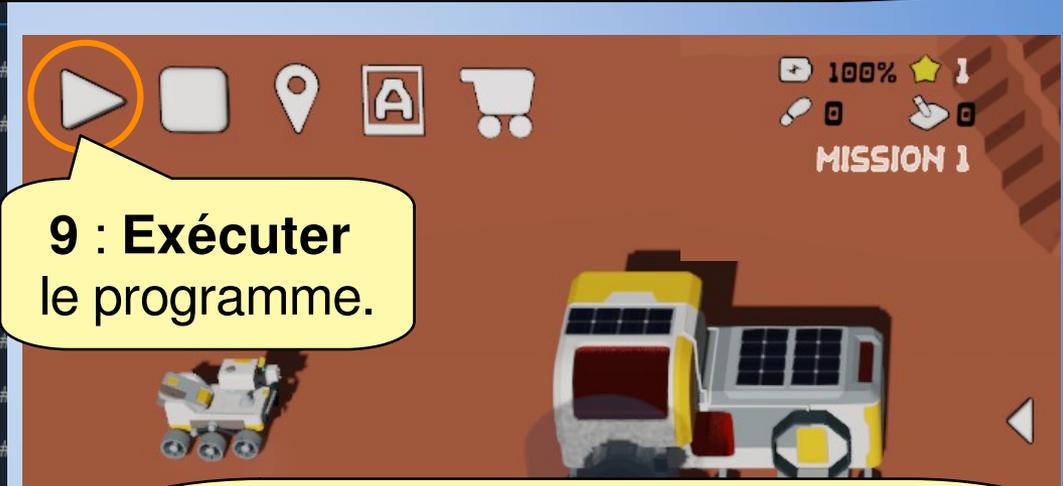


Mettre en place l'environnement de développement



```
File Edit Search Source Run Del
/home/phroy/Bureau/rp_cmd.py
rp_cmd.py X
7  # @project: Ropy (Blender-EduTech)
8  #####
9
10 #####
11 # Initialisation du niveau :
12 # Niveau 1 : Les premiers pas de Ropy
13 # Niveau 2 : Ma première fonction
14 # Niveau 3 : Sécuriser Ropy
15 # Niveau 4 : Partir au bout du monde
16 # Niveau 5 : Faire face à l'inconnu
17 # Niveau 6 : Se
18 #####
19
20 #####
21 # Fonctions
22 #####
23
24 #####
25 # Commandes
26 #####
27
28 def commandes():
29
30     rp_gauche()
31     rp_avancerr()
32     rp_avancer()
33     rp_avancer()
34     rp_avancer()
35
36     rp_fin() # A garder
37
38 #####
conda: base (Python 3.9.13) Comp
```

8 : Sauvegarder le fichier
Attention !
Toujours sauvegarder le fichier avant son exécution.



9 : Exécuter le programme.

7 : Écrire le code Python.

10 : Si votre code a une erreur,
la console indique où elle se trouve.
Ici la ligne 31 contient une commande indéfinie : **'rp_avancerr'** .

```
Fichier Édition Affichage Te
***** Module rp_cmd
rp_cmd.py:31: error (E0602, undefined-variable, commandes) Undefined variable 'rp_avancerr'

Your code has been rated at 4.44/10 (previous run: 10.00/10, -5.56)
```

Contenu du fichier rp_cmd.py



Le fichier `rp_cmd.py` comporte 4 sections.

```
import bge # Bibliothèque Blender Game Engine (UPBGE)
import time
from rp_lib import * # Bibliothèque Ropy

#####
# rp_cmd.py
# @title: Commandes pour le Rover Ropy
# @project: Ropy (Blender-EduTech)
#####

#####
# Initialisation du niveau :
# Niveau 1 : Les premiers pas de Ropy
# Niveau 2 : Ma première fonction
# Niveau 3 : Sécuriser Ropy
# Niveau 4 : Partir au bout du monde
# Niveau 5 : Faire face à l'inconnu
# Niveau 6 : Se rendre utile
#####

#####
# Fonctions
#####

#####
# Commandes
#####

def commandes():
    ➔ rp_gauche()
    rp_avancer()
    rp_avancer()
    rp_avancer()
    rp_avancer()

    rp_fin() # A garder

#####
# En: Externals calls << DONT CHANGE THIS SECTION >>
# Fr: Appels externes << NE PAS MODIFIER CETTE SECTION >>
#####

if __name__=='start':
    thread_cmd_start(commandes)
if __name__=='stop':
    thread_cmd_stop()
```

Le code doit être indenté (décalé sur la droite) avec la touche Tab

} **Import des bibliothèques**
Ne pas modifier cette section

} **Fonctions** : section pour le codage de **vos fonctions**

} **Commandes** : section pour le codage des commandes du robot

} **La commande `rp_fin()` est à conserver.**

} **Appels du simulateur**
(Blender Game Engine)
Ne pas modifier cette section