

Séquence 3

Algorithme et programmation



Simulateur 3D pour la programmation de système pluritechnique



Projet Blender-EduTech (Blender/UPBGE for Technology Education)

<https://gitlab.com/blender-edutech>

Présentation du simulateur 3D et de son environnement de programmation



Le simulateur 3D est une maquette numérique qui se commande grâce au langage **Python**. L'interface de programmation se décompose en 2 fenêtres : un éditeur de texte et le simulateur.

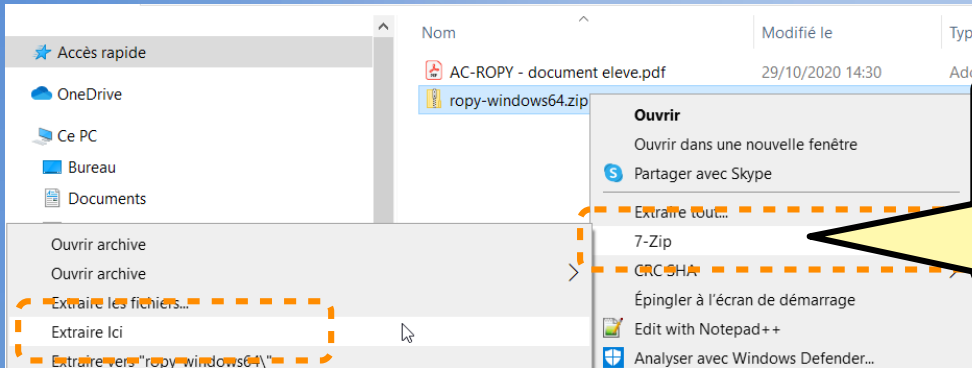


```
Spyder (Python 3.8)
Fichier  Édition  Recherche  Source  Exécution  Débugger  Console  Projets  Outils
...nts/blender-edutech/portail_couissant/programmation/portail_couissant-v1.0.0-linux64/p
porcou_cmd.py x
46 # - Temporisation en seconde : tempo(duree)
47 #
48 #####
49
50 def commandes():
51     scene.objects['Systeme']['thread_alive']=True # Tâche vivante
52
53
54     while True :
55         # Fermeture
56         if bp_int() == True :
57             print ("Fermeture : debut")
58             while fc f() ==False :
59                 gyr(True)
60                 mot f(True)
61             print ("Fermeture : fin")
62             mot f(False)
63             gyr(False)
64
65     scene.objects['Systeme']['thread_alive']=False # Tâche morte
66     print ("Thread ferme")
67
LSP Python: prêt  hon 3.8.8)  Line 50  ISO-8859-1  LF
```

Le simulateur permet de **visualiser l'évolution du système.**

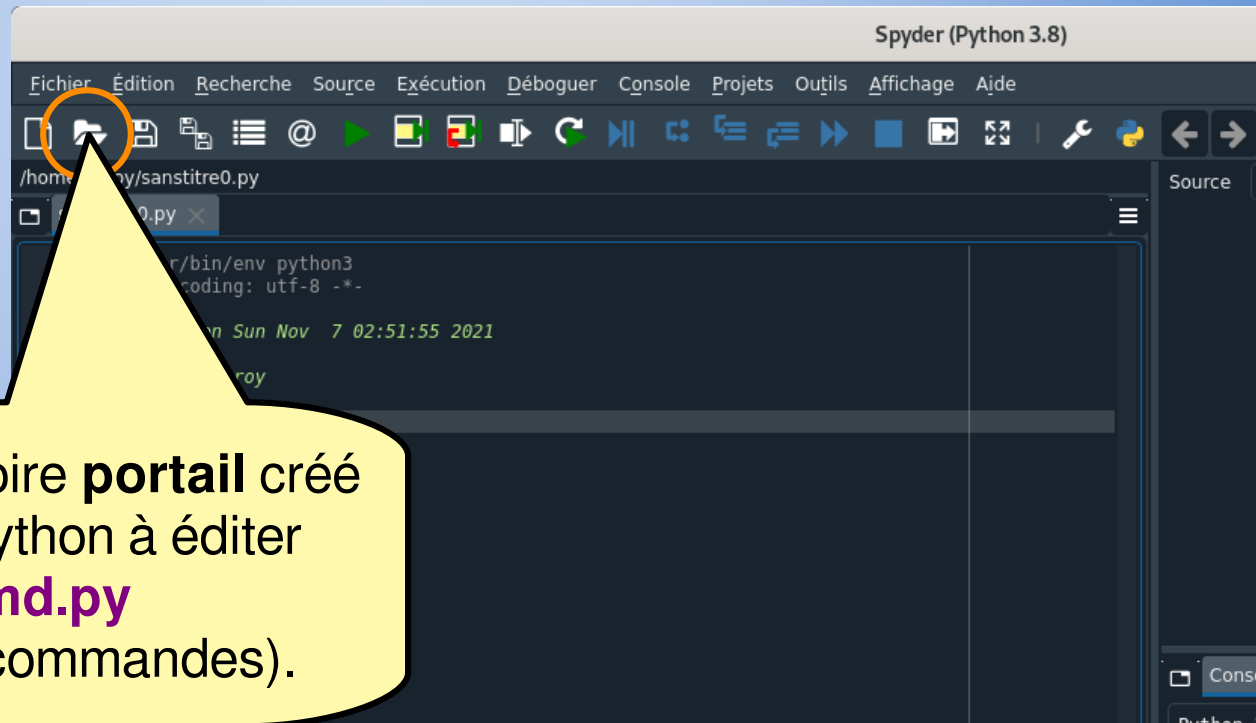
Un éditeur de texte (Notepad++, Spyder, Emacs, ...) pour **écrire le programme en Python.**

Éditer le programme avec Spyder



1 : Récupérer l'archive **portail-windows64.zip** et la décompresser avec **7-Zip** dans votre répertoire.

2 : Lancer le Logiciel **Spyder**.



3: Aller dans le répertoire **portail** créé et ouvrir le fichier Python à éditer **porcoul_cmd.py** (Portail coulissant commandes).

Éditer le programme avec Spyder



5 : **Sauvegarder** (Ctrl-s) le fichier

Attention !

Toujours sauvegarder le fichier avant son exécution avec le simulateur.

Le simulateur est le programme **portail_coulissant.exe**

```
46 # - Temporisation en seconde : tempo(duree)
47 #
48 #####
49
50 def commandes():
51     scene.objects['Systeme']['thread_alive']=True # Tâche vivante
52
53     while True :
54         # Fermeture
55         if bp_int() == True :
56             print ("Fermeture : debut")
57             while fc f() ==False :
58                 gyr(True)
59                 mot_f(True)
60             print ("Fermeture : fin")
61             mot_f(False)
62             gyr(False)
63
64     e.objects['Systeme']['thread_alive']=False # Tâche morte (
65     ("Thread ferme")
66
67
```

```
phroy@debian: ~/Documents/blender-edutech/portail_coulissant/prog
-----Details-----
Max texture units available. 4

GL_ARB_texture_env_combine supported? yes.
GL_ARB_draw_instanced supported? yes.
Thread start() ...
Fermeture : debut

```

Ce n'est pas obligatoire, mais l'exécution du simulateur à partir d'un **terminal** permet d'afficher les **messages** de la sortie standard (**print**) afin de faciliter de **débogage**.

4 : **Écrire** le code Python

Interface du simulateur



Le système
en fonctionnement

Exécuter le programme
puis Pause (touche **F5**)

Remise à zéro de
la vue (touche **début**)

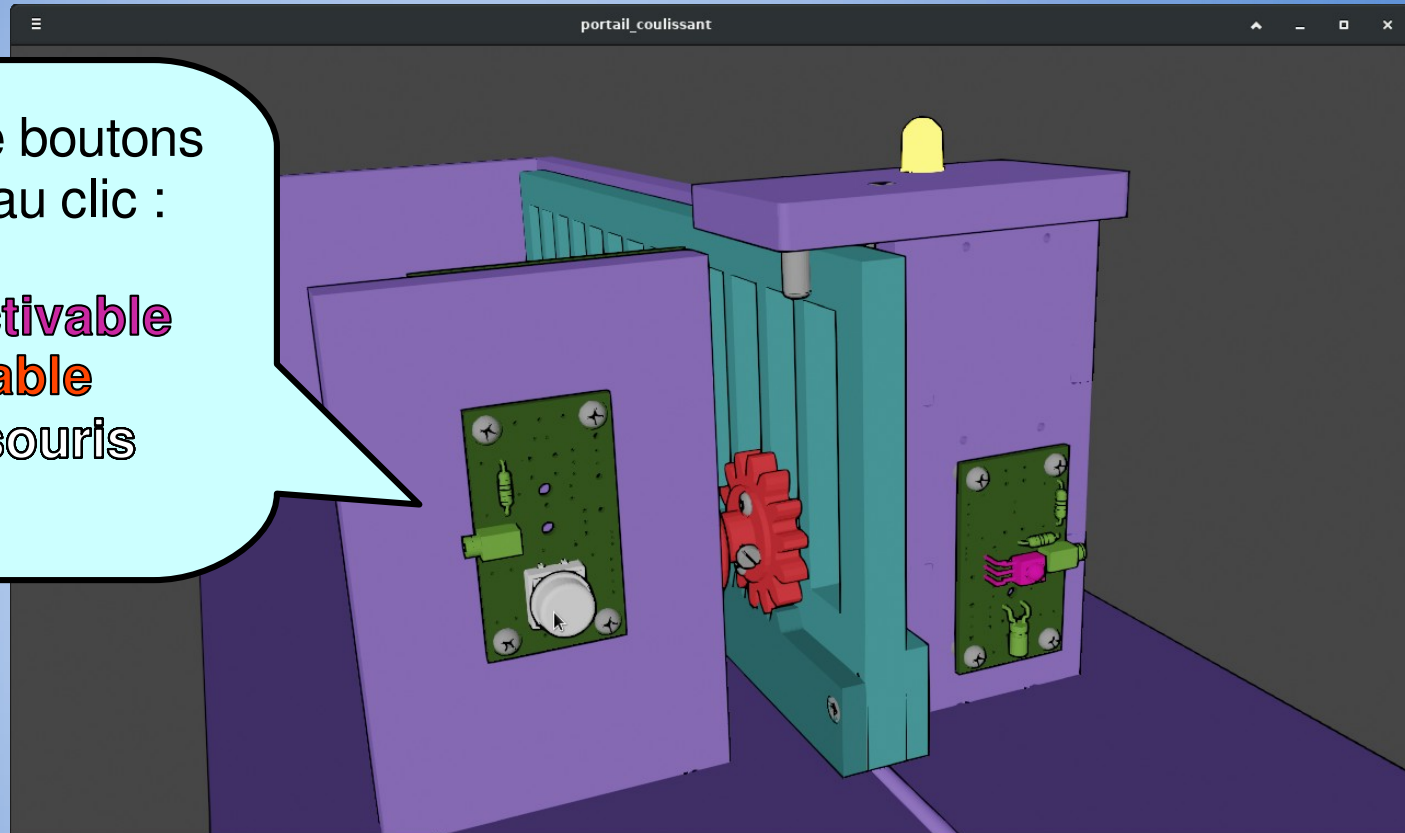
Page
d'aide

Manipulation de la maquette numérique



Les capteurs et le boutons sont sensibles au clic :

- **Magenta** : inactivable
- **Orange** : activable
- Blanc : focus souris
- **Jaune** : activé



Le **bouton du centre** sert à **manipuler** le modèle 3D :

- **Clic centre** : rotation du mécanisme (orbit)
- **Clic centre + Maj** : déplacement du mécanisme (pan)
- **Clic centre + Ctrl** : zoom
- **Molette** : zoom

Carte de référence du portail coulissant



Boutons :

- Bouton poussoir coté rue : `bp_ext()`
- Bouton poussoir coté cour : `bp_int()`

Capteur de fin de course :

- Capteur portail ouvert : `fc_o()`
- Capteur portail fermé : `fc_f()`

Moteur :

- Ouvrir le portail : `mo_o(ordre)`
- Fermer le portail : `mo_f(ordre)`

Gyrophare :

- Allumé/éteindre : `gyr(ordre)`

Capteur barrage :

- Activation de l'émetteur : `ir_emet(ordre)`
- État du récepteur (absence d'obstacle) : `ir_recep()`

Valeur retournée par les capteurs et les boutons

- `True` : actif
- `False` : inactif

Ordre pour les actionneurs

- `True` : activer
- `False` : désactiver