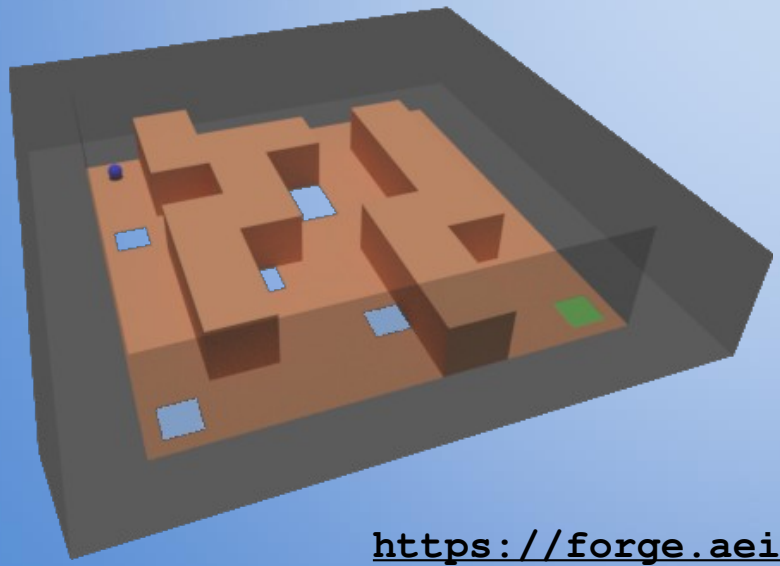
Labyrinthe à bille
Créer une scène 3D interactive

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 définis par les **briques logiques** avec **un module Python**. La guidance de ce tutoriel a pour pré-requis la réalisation du tutoriel précédent (Tutoriel 1 - Ma première scène).

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).

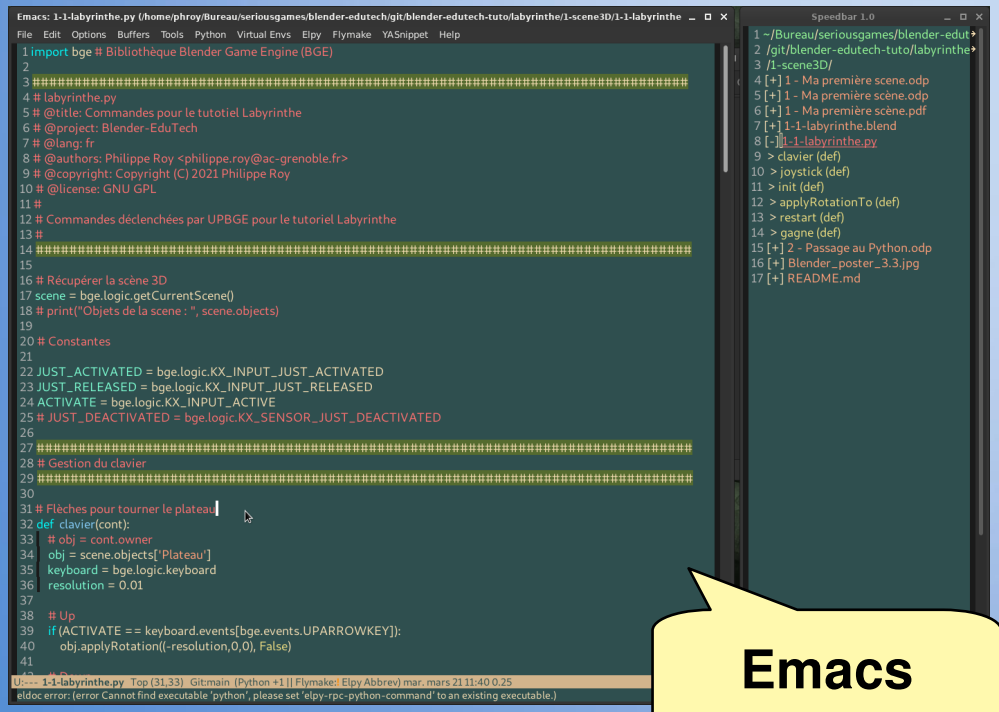
Le tutoriel se décompose en 5 étapes :

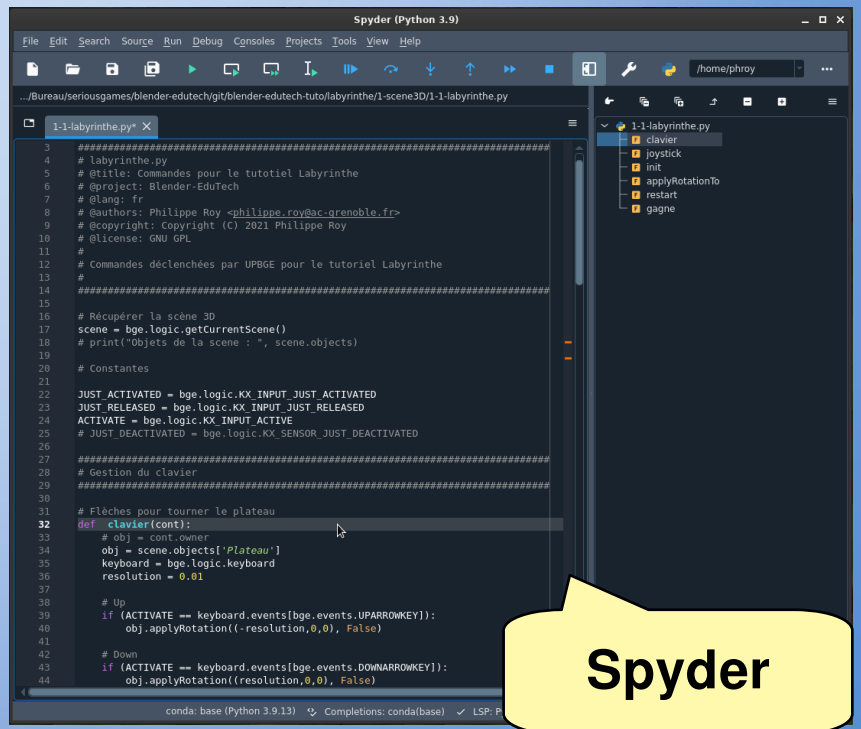
- 1. Installer l'**éditeur de texte**
- 2. **Déplacer** le plateau
- 3. **Définir le game play** (règles d'échec et de réussite)
- 4. **Animer** la fenêtre de fin
- 5. Fermer la fenêtre de fin par un **bouton cliquable**

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



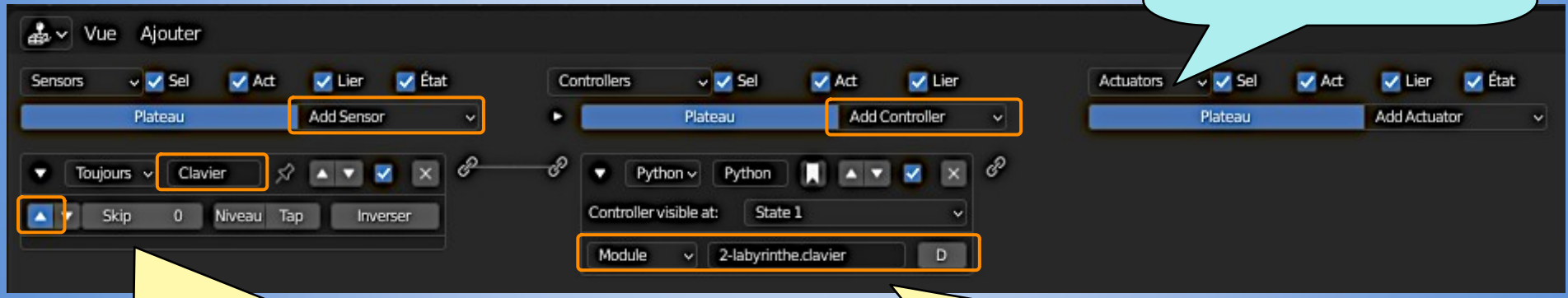


2. Déplacer le plateau

Le fichier Blender de départ est le fichier résultat du tutoriel 1 sans les briques logiques ni les propriétés. Il est disponible dans le répertoire du tutoriel sous le nom « **2-labyrinthe-debut.blend** ».

Pour la gestion du clavier, le principe est de créer un **boucle infinie** qui exécute la fonction **clavier** à chaque **pulsation logique**.

Briques logiques de **Plateau**



1 : Créer la boucle infinie

- **Ajouter** un **Capteur Toujours**
- **activer** le **Pulse True Level (▲)**
- **renommer** le capteur avec **Clavier**

2 : Appeler la fonction

- **Ajouter** un **Contrôleur Python**
- **définir** le **Module** avec la fonction **2-labyrinthe.clavier**

2. Déplacer le plateau

Le module Python est le fichier « **2-labyrinthe.py** ».

```
import bge # Bibliothèque Blender Game Engine (UPBGE

################################################################################
# 2-labyrinthe.py
################################################################################

# Récupérer la scène 3D
scene = bge.logic.getCurrentScene()

# Constantes
JUST_ACTIVATED = bge.logic.KX_INPUT_JUST_ACTIVATED
JUST_RELEASED = bge.logic.KX_INPUT_JUST_RELEASED
ACTIVATE = bge.logic.KX_INPUT_ACTIVE

################################################################################
# Gestion du clavier
################################################################################

# Flèches pour tourner le plateau
def clavier(cont):
    obj = scene.objects['Plateau'] # obj est l'objet 'Plateau' de la scène
    keyboard = bge.logic.keyboard
    resolution = 0.01

    # Flèche haut - Up arrow
    if (ACTIVATE == keyboard.events[bge.events.UPARROWKEY]):
        obj.applyRotation((-resolution,0,0), False)

    # Flèche bas - Down arrow
    if (ACTIVATE == keyboard.events[bge.events.DOWNARROWKEY]):
        obj.applyRotation((resolution,0,0), False)

    # Flèche gauche - Left arrow
    if (ACTIVATE == keyboard.events[bge.events.LEFTARROWKEY]):
        obj.applyRotation((0, -resolution,0), False)

    # Flèche droit - Right arrow
    if (ACTIVATE == keyboard.events[bge.events.RIGHTARROWKEY]):
        obj.applyRotation((0, resolution,0), False)
```

3 : Créer le fichier Python
Ouvrir votre éditeur et créer le fichier **2-labyrinthe.py**

4 : Créer le fonction clavier
Copier-coller le code

5 : Tester la scène [P]

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

Le module Python est le fichier « **2-labyrinthe.py** ».

```
import bge # Bibliothèque Blender Game Engine (UPBGE

################################################################################
# 2-labyrinthe.py
################################################################################

# Récupérer la scène 3D
scene = bge.logic.getCurrentScene()

# Constantes
JUST_ACTIVATED = bge.logic.KX_INPUT_JUST_ACTIVATED
JUST_RELEASED = bge.logic.KX_INPUT_JUST_RELEASED
ACTIVATE = bge.logic.KX_INPUT_ACTIVE

################################################################################
# Gestion du clavier
################################################################################

# Flèches pour tourner le plateau
def clavier(cont):
    obj = scene.objects['Plateau']
    keyboard = bge.logic.keyboard
    resolution = 0.01

    # Flèche haut - Up arrow
    if (ACTIVATE == keyboard.events[bge.events.UPARROWKEY]):
        obj.applyRotation((-resolution,0,0), False)

    # Flèche bas - Down arrow
    if (ACTIVATE == keyboard.events[bge.events.DOWNARROWKEY]):
        obj.applyRotation((resolution,0,0), False)

    # Flèche gauche - Left arrow
    if (ACTIVATE == keyboard.events[bge.events.LEFTARROWKEY]):
        obj.applyRotation((0, -resolution,0), False)

    # Flèche droit - Right arrow
    if (ACTIVATE == keyboard.events[bge.events.RIGHTARROWKEY]):
        obj.applyRotation((0, resolution,0), False)
```

3 : Créer le fichier Python
Ouvrir votre éditeur et créer le fichier **2-labyrinthe.py**

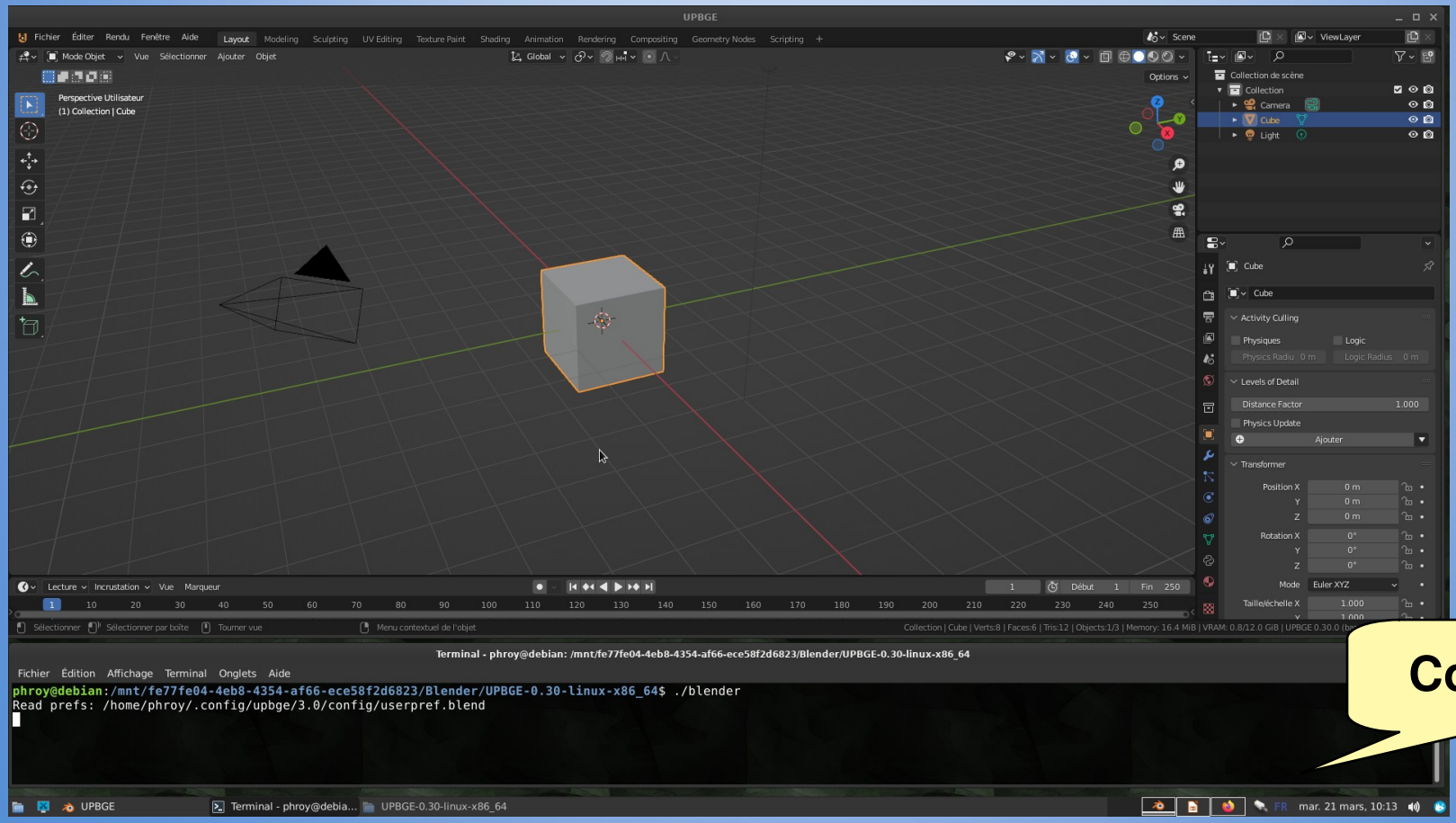
4 : Créer le fonction clavier
Copier-coller le code

5 : Tester la scène [P]

4. Animer la fenêtre de fin

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

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



5. Fermer la fenêtre de fin par un bouton cliquable

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

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