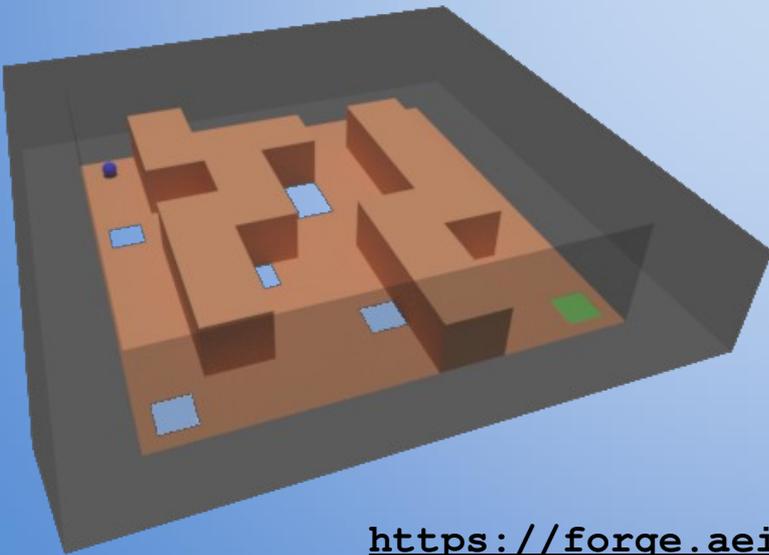


# Labyrinthe à bille

## Créer une scène 3D interactive

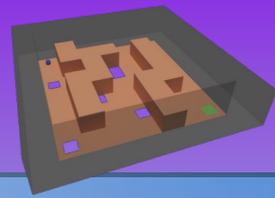
# Tutoriel 2

## 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**. La guidance de ce tutoriel a pour pré-requis la réalisation du tutoriel précédent (Tutoriel 1 - Ma première scène).

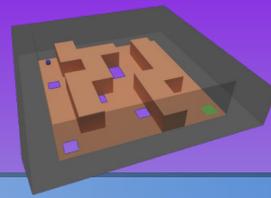
Le codage en Python permet

- 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



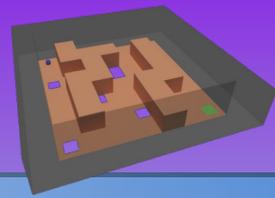
Le codage se fait par un éditeur de texte. Il en existe beaucoup et tout éditeur peut convenir. Pour ce tutoriel j'utilise **Emacs**, il est open source, très efficace et polyvalent mais peu intuitif. Le choix de l'éditeur est souvent très personnel, sans préférence je vous conseille **Spyder**, il est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

```
1-1-labyrinthe.py (~/home/phroy/Bureau/seriousgames/blender-edutech/git/blender-edutech-tuto/labyrinthe/1-scene3D/1-1-labyrinthe)
File Edit Options Buffers Tools Python Virtual Envs Elpy Flymake YASnippet Help
1 import bge # Bibliothèque Blender Game Engine (BGE)
2
3 #####
4 # labyrinthe.py
5 # @title: Commandes pour le tutoriel 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 scène : ", 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)
45
46     # Left
47     if (ACTIVATE == keyboard.events[bge.events.LEFTARROWKEY]):
48         obj.applyRotation((0,-resolution,0), False)
49
50     # Right
51     if (ACTIVATE == keyboard.events[bge.events.RIGHTARROWKEY]):
52         obj.applyRotation((0,resolution,0), False)
53
54     # Esc
55     if (ACTIVATE == keyboard.events[bge.events.ESCKEY]):
56         obj.applyRotation((0,0,0), False)
57
58     # Quit
59     if (ACTIVATE == keyboard.events[bge.events.EXIT]):
60         obj.applyRotation((0,0,0), False)
61
62     # End
63     return True
64
65 #####
66 # README.md
67 #####
```

```
1-1-labyrinthe.py (~/home/phroy/Bureau/seriousgames/blender-edutech/git/blender-edutech-tuto/labyrinthe/1-scene3D/1-1-labyrinthe.py)
File Edit Search Source Run Debug Consoles Projects Tools View Help
.../Bureau/seriousgames/blender-edutech/git/blender-edutech-tuto/labyrinthe/1-scene3D/1-1-labyrinthe.py
1-1-labyrinthe.py X
3 #####
4 # labyrinthe.py
5 # @title: Commandes pour le tutoriel 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 scène : ", 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)
45
46     # Left
47     if (ACTIVATE == keyboard.events[bge.events.LEFTARROWKEY]):
48         obj.applyRotation((0,-resolution,0), False)
49
50     # Right
51     if (ACTIVATE == keyboard.events[bge.events.RIGHTARROWKEY]):
52         obj.applyRotation((0,resolution,0), False)
53
54     # Esc
55     if (ACTIVATE == keyboard.events[bge.events.ESCKEY]):
56         obj.applyRotation((0,0,0), False)
57
58     # Quit
59     if (ACTIVATE == keyboard.events[bge.events.EXIT]):
60         obj.applyRotation((0,0,0), False)
61
62     # End
63     return True
64
65 #####
66 # README.md
67 #####
```

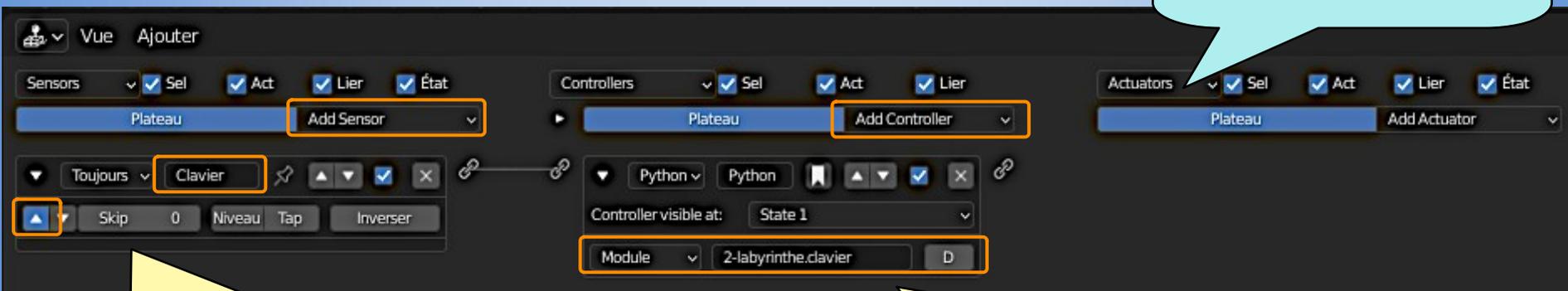
## 2. Déplacer le plateau



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 la 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 la fonction clavier**  
Copier-coller le code

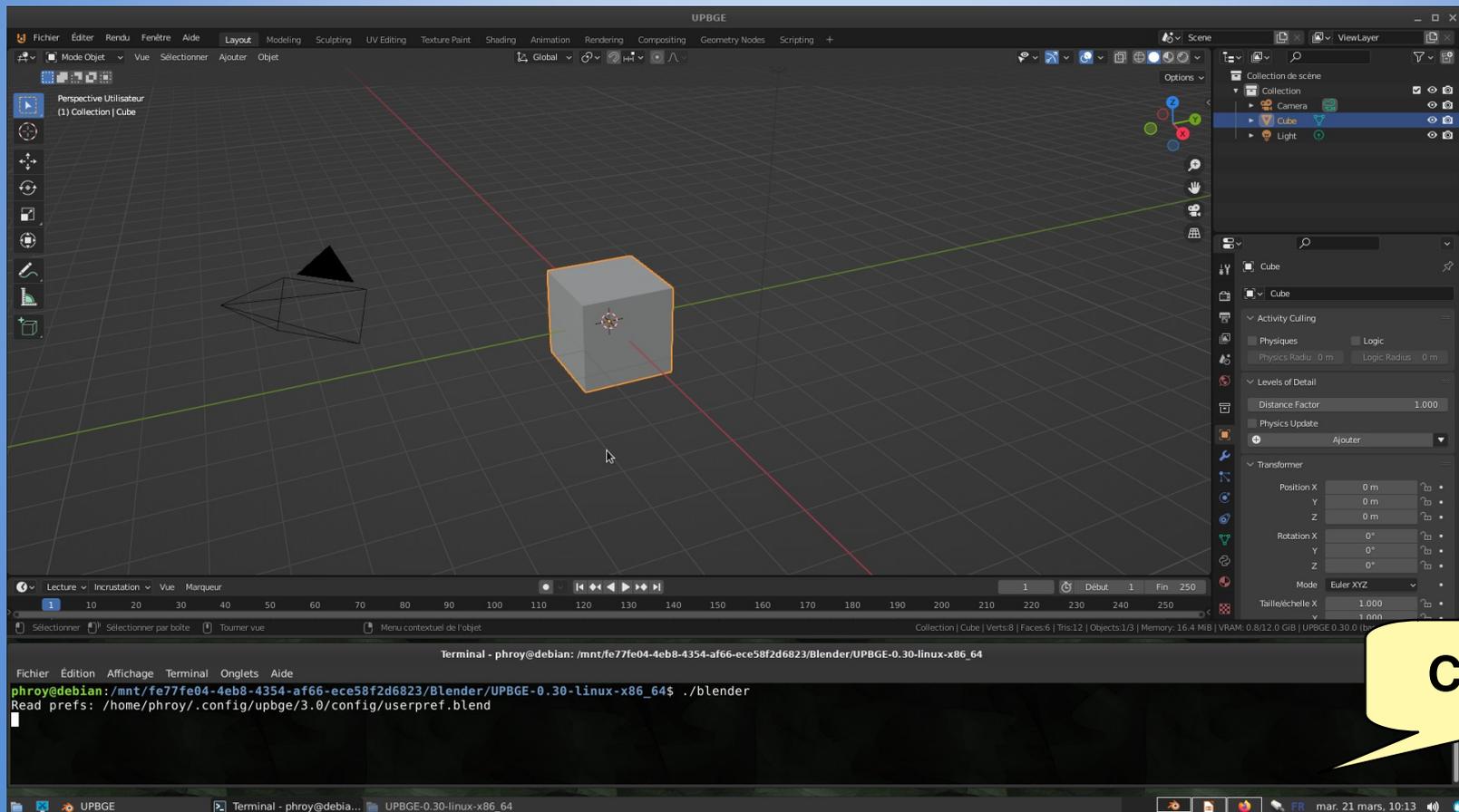
**5 : Tester la scène [P]**

# 4. Animer la fenêtre de fin



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>



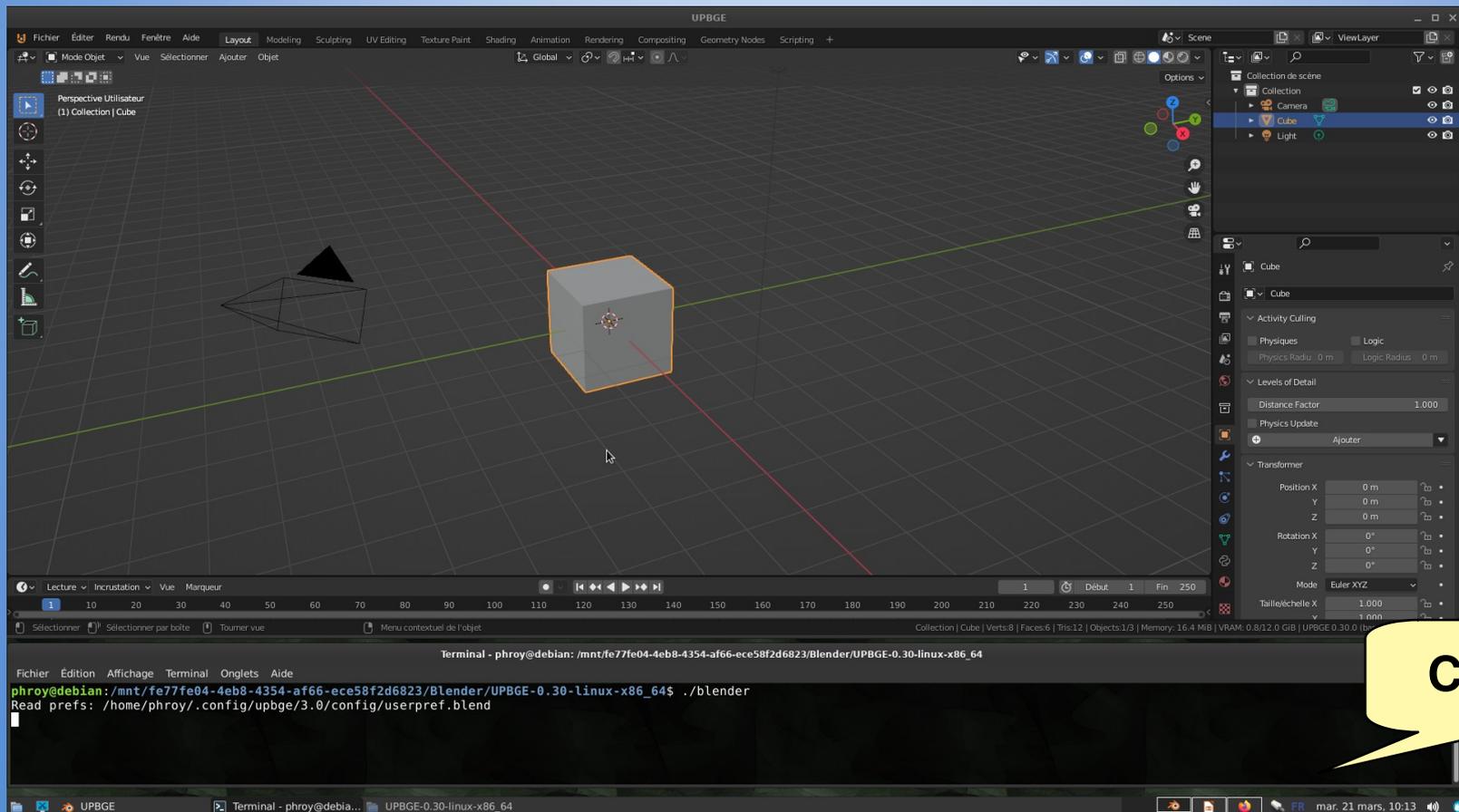
Console

# 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